Kernel-based ARchitecture for safetY-critical cONtrol

KARYON

FP7-288195

D3.2–Final Report on Network Characteristics and Coordination Techniques

Work Package	WP3		
Due Date	24	Submission Date	2013-12-03
Main Author(s)	José Rufino (FFCUL), Jeferson Souza (FFCUL), Elad Michael Schiller (CUT)		
Contributors	Luís Marques (FFCUL), André Guerreiro (FFCUL), Rui P. Caldeira (FFCUL), Daniel Skarin (SP)		
Version	V1.0	Status	Final
Dissemination Level	Public	Nature	Report
Keywords	dependability, timeliness, real-time, fault tolerance, fault injection, network coordination, dependable adaptation, wireless communications, safety-critical control.		
Reviewers			

Part of the Seventh

Framework Programme

Funded by the EC - DG INFSO

Version history

Rev	Date	Author	Comments
V0.1	2013-07-18	José Rufino (FFCUL)	Initial contents definition
V0.2	2013-10-03	Jeferson Souza (FFCUL)	Adding FFCUL contributions
V0.3	2013-11-28	Jeferson Souza (FFCUL)	Adding Chalmers and SP contributions
V0.4	2013-12-02	Jeferson Souza (FFCUL)	Content and format edition
V1.0	2013-12-03	José Rufino (FFCUL) Jeferson Souza (FFCUL)	Final editing and review
V1.1	2013-12-03	António Casimiro (FFCUL)	Final review and delivery

Glossary of Acronyms

API	Application Programming Interface
BO	Beacon Order
CAP	Contention Access Period
CBR	Constant Bit Rate
CFP	Contention Free Period
COTS	Commercial Off-The-Shelf
CRC	Cyclic Redundancy Check
CTS	Clear To Send
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access / Collision Detection
DC	Diagnostic Coverage
D.C.	Direct Current
DFDR	Data Frame Delivery Ratio
DoW	Description of Work
Dx.y	Deliverable belonging to work package x, with serial number y
EDS	Electronic Data Sheet
FCS	Frame Check Sequence
FIFO	First In First Out
GPL	General Public License
GPS	Global Positioning System
GNSS	Global Navigation Satellite System
GTS	Guaranteed Time Slot
IEEE	Institute of Electrical and Electronics Engineers
IP	Inactive Period
IST	Information Society Technologies
KARYON	Kernel-based ARchitecture for safetY-critical cONtrol
LAN	Local Area Network
LIN	Local Interconnect Network
MAC	Medium Access Control
MIMO	Multiple Input Multiple Output
MLA	Mediator Layer
NS-2	Network Simulator 2
OEM	Original Equipment Manufacturer

OS	Operating System
OSI	Open Systems Interconnection
PC	Personal Computer
PCAP	Packet Capture
PER	Packet Error Rate
PHY	Physical Layer
P/S	Publisher/Subscriber
QoS	Quality of Service
R2T-MAC	Resilient Real-Time Medium Access Control
RCM	Ranging and Communication Module
RF	Radio Frequency
RPN	Risk Priority Number
RTE	RunTime Environment
RTS	Request To Send
SNR	Signal-to-Noise Ratio
SO	Superframe Order
TCP/IP	Transmission Control Protocol/Internet Protocol
TDMA	Time Division Multiple Access
TEDS	Transducer Electronic Datasheet
Tx.y	Task belonging to work package x, with serial number y
UDP	UserDatagramProtocol
UID	UniqueIdentifier
UWB	Ultra-Wideband
V2I	Vehicle-to-Infrastructure
V2R	Vehicle-to-Roadside
V2V	Vehice-to-Vehicle
VANET	Vehicular Ad Hoc Network
WnS	Wireless Network Segment
WLAN	Wireless Local Area Network
WP	Work Package
WPx	Work Package with serial number x
WSAN	Wireless Sensor and Actuator Network
WSN	Wireless Sensor Networks
XML	Extensible Markup Language

Executive Summary

The objectives of WP3 concentrate on supporting technologies to enable dependable control, interaction, monitoring and fault management in the cooperative distributed system. WP3 translates the hybrid architectural safety concepts developed in WP2 to system components, interfaces and services. The primary focus is on maintaining a high functional level in spite of environment uncertainties and faults of the communication and the computational systems. In that sense, there is also a strong relation with WP4, since the different components must be engaged in cooperation with the safety kernel for adaptation.

Thus, WP3 establishes a bridge between the conceptual work in WP2 and WP4 and the practical aspects of implementing a KARYON proof of concept demonstrator in WP5. Thus, the *Final Report in Network Characteristics and Coordination Techniques*, will include: the conceptual framework, the models, the methods and the tools required to deal with and evaluate all the uncertainty aspects of communications. This report will provide an outline of the services and structure of the supporting technologies. Taking into account the specified in the DoW, this report would be preferably compiled from papers accepted at relevant conferences. In particular, the report will encompass:

- a detailed analysis of network inaccessibility models, aiming to allow the evaluation of all the periods of network inaccessibility using the IEEE 802.15.4 as a use-case.
- the protocols and the protocol layers required to control uncertainties in communications, thus providing reliable and timely communication services despite the presence of errors;
- propose protocols that deal with uncertainties of networks. Particular emphasis is on providing abstractions to handle faults and support environment perception;
- provide a relevant set of tools for the analysis and verification of previous models. This includes model verification by fault injection techniques.
- a specification of the structure and of the service interface of the different components intended to cope uncertainty;
- reporting advances with respect to the reliable cooperation between mobile nodes and how they can set up vehicle coordination using the safety kernel.

This deliverable mainly describes the activities developed and the achievements reached in the execution of task T3.1, between M12 and M24. Task T3.1, now completed, was specifically related with fundamental aspects of predictability and resilience in embedded networks.

Given the work developed in the context of Task T3.1 has produced a set of analytical and experimental tools, Deliverable D3.2 is also, at least in part, related with Deliverable D3.3.

Table of Contents

List of Figures

List of Tables

1. Introduction

The objectives of WP3 concentrate on supporting technologies to enable dependable control, interaction, monitoring and fault management in the cooperative distributed system. WP3 translates the hybrid architectural safety concepts developed in WP2 to system components, interfaces and services. The primary focus is on maintaining a high functional level in spite of environment uncertainties and faults of the communication and the computational systems. In that sense, there is also a strong relation with WP4, since the different components must be engaged in cooperation with the safety kernel for adaptation.

Deliverable D3.2 – *The Final Report in Network Characteristics and Coordination Techniques*, includes: the conceptual framework, the models, the methods and the tools required to deal with and evaluate all the uncertainty aspects of communications. This report provides an outline of the services and structure of the supporting technologies. In particular, the report encompasses:

- a detailed analysis of network inaccessibility models, aiming to allow the evaluation of all the periods of network inaccessibility using the IEEE 802.15.4 as a use-case.
- the protocols and the protocol layers required to control uncertainties in communications, thus providing reliable and timely communication services despite the presence of errors;
- propose protocols that deal with uncertainties of networks. Particular emphasis is on providing abstractions to handle faults and support environment perception;
- provide a relevant set of tools for the analysis and verification of previous models. This includes model verification by fault injection techniques.
- a specification of the structure and of the service interface of the different components intended to cope uncertainty;
- reporting advances with respect to the reliable cooperation between mobile nodes and how they can set up vehicle coordination using the safety kernel.

The present document in essence describes the activities developed and the achievements reached in the execution of task T3.1, between M12 and M24. Task T3.1, which is now completed, was specifically related with fundamental aspects of predictability and resilience in embedded networks.

Taking into account the specified in the DoW, this report would be preferably compiled from papers accepted at relevant conferences or from technical reports. Thus, when applicable the reader is referred to a set of published papers and technical reports, included in Annex A.

Beyond the technical work of more formal and theoretical nature, a set of tools was developed within the context of the work developed in Task T3.1, namely:

- IEEE 802.15.4 Network Inaccessibility Evaluation Tool (OpenOffice)
- NS-2 Simulator module with GTS (Guaranteed Time Slots) support for frame transmissions on IEEE 802.15.4 wireless networks
- NS-2 Simulator extension module for fault-injection and timeliness evaluation of IEEE 802.15.4 wireless networks under error conditions
- Wireshark extension module for fault-injection, monitoring and evaluation of IEEE 802.15.4 wireless networks

These software packages already are (or will be very soon) publically available at the project website. In this sense, Task T3.1 during the reported period is also strongly related with Deliverable 3.3 - Working prototype of adaptive middleware.

2. Wireless Networks for Advanced Safety-Critical Control

This section addresses the characterisation of wireless network communications relevant for advanced safety-critical control, analysing the problems and presenting solutions associated with the provision of dependability and timeliness properties.

2.1 Impairments to Real-Time Wireless Communications

Wireless technologies are not mature enough to provide a trustful communication service in terrestrial and aerial environments addressed by the **KARYON** project, where communications between vehicles may have real-time requirements associated to safety-critical operation [10] . Inter-vehicular communications should be predictable, reliable, and time-bounded.

However, the use of wireless technologies introduces different source of interferences, which may generate disturbances, and because of them, the occurrence of communication errors. These communication errors place difficulties in the establishment and provision of a wireless real-time communication service, since the effects may vary from unnoticed loss of sporadic data frames to temporary and/or permanent break of the networked services.

Communication errors may have different sources, which are divided in three categories: **electromagnetic interferences**, **obstacles on the communication path**, and **mutual node interference**. Mobility could also be considered as an impairment to communications, however as it also introduces advantages to wireless communications, we present it separately.

It is important to remark that all of those communication impairments may occur accidentally or intentionally. Despite of the importance to deal with communication errors caused intentionally by a malicious entity, this document focuses its analysis only in the presence of accidental communication errors.

2.1.1 Electromagnetic interferences

The presence of noise emitted in the same radio channel(s) utilised by the network, characterises a phenomenon dubbed **electromagnetic interferences**. The presence of **electromagnetic interferences** may overlap, interrupt, or obstruct the network traffic, which may imply the loss of transmitted frames. Such kind of interference is characterised by the existence of fixed or transitory entities external to the network, being electrical equipment (e.g., electrical motors) the most common example of a source of electromagnetic interferences.

The common metric utilised to indicate the presence of undesirable radio frequencies (RF) is the signal-to-noise ratio (SNR). SNR helps to measure and indicate the quality of a communication channel, and during the signal reception, it is characterised by the ratio among the power of the desired signal (e.g., network traffic) and the power of the noise (undesirable signal). A good communication channel is the one that has the power of desired signal much higher than the power of noise within such relation.

Designed modulation and transmission techniques improve the protection against interferences on wireless communications, mainly originated by accidental disturbances. Spread spectrum modulation techniques [36] enable the coexistence of different networks with different standards, operating in the same frequency band with reduced mutual interference in the same geographical operating space.

2.1.2 Obstacles on the communication path

Obstacles on communication path may difficult the propagation of signals on wireless networks. The effects derived from the presence of obstacles may cause different interference phenomenon such as signal block, reflection, refraction, or diffraction, implying in partial or total loss of the transmitted frames.

The material and the superficies of each object may induce one or multiple of the aforementioned interference phenomena, which may also cause a known wireless communication problem dubbed ***fading problem***. A common ***fading problem*** is the multipath fading where a transmitted frame reaches its receivers in different paths [16] . Some of these paths may change the phase of the signal that carries the frame, which may result in a destructive interference that weak the received signal and therefore may cause the inability to recognize the incoming frame.

2.1.3 Mutual node interference

The presence of both electromagnetic interferences and obstacles on communication path may contribute to the occurrence of other type of communication impairment dubbed **mutual node interference**. Two of the most famous **mutual node interference** problems are the hidden and exposed node problems [1] . These problems are the most relevant examples of disturbances caused by **mutual node interference**, which may disrupt communication services on wireless networks.

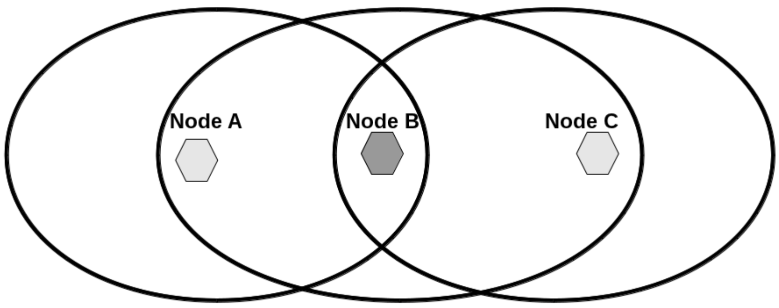


Figure 1 - Hidden terminal problem

The hidden node problem occurs when two or more nodes cannot sense one another, because the distance among nodes or even obstacles on the communication path, as illustrated in Figure 1. In the example of Figure 1, the ellipses denote the coverage region of each node. Note that node *A* and *C* are outside of their respective coverage regions, and therefore cannot sense each other. Frames transmitted by these nodes may be overlapped at some point into the space comprising the intersection of the coverage regions of both *A* and *C* nodes, where node *B* is located.

In case of exposed node problem, transmissions from neighbour nodes may prevent other nodes to send their frames as illustrated in Figure 2. Again, each ellipse in Figure 2 represents the coverage region of a node. In this case, nodes *B* and *C* are within the coverage region of one another, being node *A* only in the coverage region of node *B* and node *D* only in the coverage region of node *C*. Thus, when node *B* is transmitting a frame to node *A*, node *C* senses activity on the network and cannot transmit its own frames to node *D* (see Figure 2), even with node *D* outside of the coverage region of the node *B*.

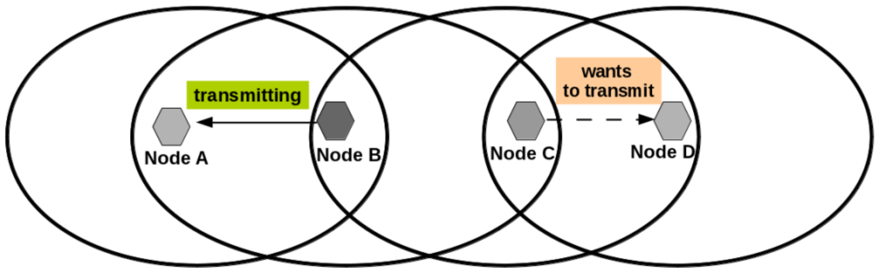


Figure 2 - Exposed terminal problem

2.1.4 Mobility

Node mobility is an optional but remarkable feature of wireless nodes, which is characterised by the capability of a node to change its position without a hard dependency of a physical infrastructure. However, mobility is also a difficult problem to deal with.

When a node moves out of range from other nodes that are part of the same network, communications cannot be established and performed, and therefore communication errors may occur. Unpredictable environments are created if nodes within a network move freely throughout the environment, and to outside of the communication range of others nodes.

2.2 Characterising Wireless Networking Communications

The occurrence of electromagnetic interferences, the presence of obstacles on the communication path, the hidden and exposed terminal problems, or even node mobility (in some cases) cannot be prevented.

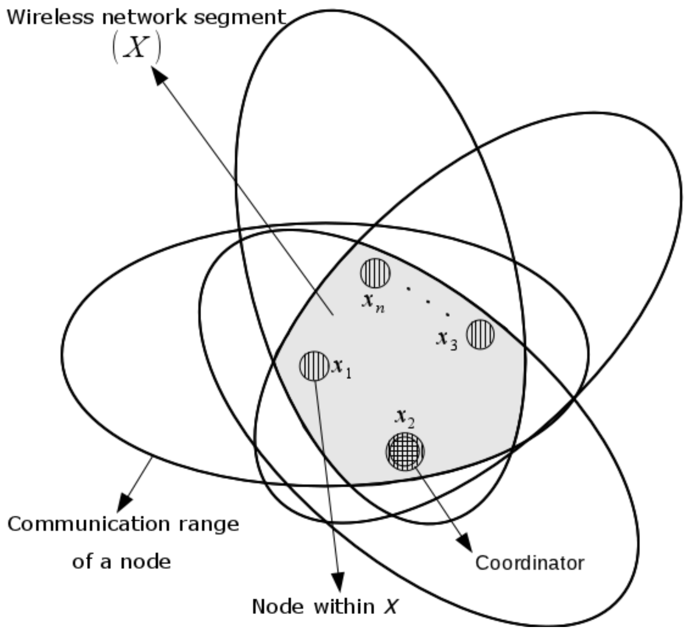


Figure 3 - Wireless network segment

The rigorous definition of a system model, considering communication faults, is a crucial step to understand and characterise the fundamental aspects of wireless communications and wireless network operation. The system model utilised by the **KARYON** project is formed by a set of wireless nodes$X = \{x_1, x_2, x_3, \cdots, x_n\}$, being $1 < n \leq \#A$, where A is the set of all wireless nodes using the same communication channel(s). A wireless node is a networking device capable to communicate with other nodes. In the rest of the document, we use the terms wireless node and node interchangeably.

The set X itself represents a networking entity dubbed wireless network segment (WnS), as depicted in Figure 3. A WnS establishes a wireless network where each node can sense one another within one-hop of distance, being complex networks composed by more than one WnS. For simplification purposes, our analysis assumes a network with one WnS, being its behaviour supported in the following assumptions:

1. The communication range of X, i.e. its broadcast domain, is given by: $B_X = \bigcap_{j=1}^{n} B_D(x),\ \forall x\ \in X$, where $B_D(x)$represents the communication range of a node x;
2. $\forall x\ \in A, x\ \in X \Leftrightarrow B_X\ \subseteq B_D(x)$ or, as a consequence of node mobility, $x \notin X \Leftrightarrow B_X\ \nsubseteq B_D(x)$;
3. $\forall x\ \in X$ can sense the transmissions of one another;
4. $\exists x\ \in X$ which is the coordinator, being unique and with responsibility to manage the set;
5. A network component (e.g. a node $x\ \in X$) either behaves correctly or crashes upon exceeding a given number of consecutive omissions (the component's *omission degree*), f_o, in a time interval of reference[1];
6. Failure bursts never affect more than f_o transmissions in a time interval of reference, T_{rd};
7. Omission failures may be inconsistent (i.e., not observed by all recipients).

For a given WnS, assumptions 1, 2, and 3 define the physical relationship between nodes, assumption 4 defines the existence of a coordinator, and assumptions 5, 6, and 7 define how the occurrence of communication errors are modelled and handled within the WnS. All communications and relations between nodes are established at Medium Access Control (MAC) level, which are reinforced by assumption 3. As a consequence of mobility, nodes may be driven away of or enter a given WnS (assumption 2). All communication errors within WnS are transformed into omissions (assumption 5), and in the context of network components an omission is an error that destroys a data or control frame.

Establishing a bound for the *omission degree* of individual components provides a general method for the detection of failed components. If each omission (frame loss) is detected and accounted for, the component fails once it exceeds the *omission degree bound,* f_o. The omission degree is thus a general measure of the reliability of network components with respect to transient errors.

[1]For instance, the duration of a given protocol execution. Note that this assumption is concerned with the total number of failures of possibly different nodes.

2.2.1 Wireless MAC-level properties

A relevant set of properties, presented in Table 1, are defined for the MAC sublayer and hold for the WnS. In wired communications, it has been proven that those properties are extremely useful for enforcing dependability and timeliness at higher layers [43] [38] . Thus, we are applying those techniques to the realm of wireless communications [40] .

Wireless MAC-level properties
WMAC1 – *Broadcast*: correct nodes on a WnS, receiving an uncorrupted frame transmission, receive the same frame.
WMAC2 – *Error Detection*: correct nodes detect any corruption done during frame transmissions in a locally received frame.
WMAC3 – *Frame Order*: any two frames received at any two correct nodes on a WnS are received in the same order at both nodes.
WMAC4 – *Local Full-Duplex*: a correct node may receive, on request, local frame transmissions.
WMAC5 – *Bounded Omission Degree*: on a WnS, within a known time interval $\boldsymbol{T_{rd}}$, omission failures may occur in at most *k* transmissions.
WMAC6 – *Bounded Inaccessibility*: in a known time interval $\boldsymbol{T_{rd}}$, a WnS may be inaccessible at most *i* times, with a total duration of at most $\boldsymbol{T_{ina}}$.
WMAC7 – *Bounded Transmission Delay*: any frame transmission request is transmitted on the WnS, within a bounded delay $\boldsymbol{T_{td}} + \boldsymbol{T_{ina}}$.
WMAC8 – *Tightness*: correct nodes on a WnS receiving an uncorrupted frame transmission, receive it at real time instants that differ, at most, by a known small time value $\Delta\boldsymbol{T_{tight}}$.

Table 1 - General Wireless MAC-level properties

Properties **WMAC1** and **WMAC2** impose correctness in the value domain. Property **WMAC1** (***Broadcast***) formalises that it is physically impossible to send conflicting information to different nodes, in the same broadcast on a WnS [7] . Property **WMAC2** (***Error Detection***) derives directly from frame protection through a CRC[2] polynomial, as provided by the MAC level. The MAC controller itself usually discards frames affected by errors. This means, frame errors are transformed into omissions. The residual probability of undetected frame errors is negligible [19] [17]

The extension of property **WMAC2** to include the signalling of frame discard actions to other protocol entities may significantly contribute to enhance the liveness properties at MAC protocol level. The provisioning of such unconventional primitive can be enabled by emerging controller technology, such as reprogrammable and/or open core MAC level solutions. No modifications are needed to the wireless MAC standards such as IEEE 802.15.4 [26] and IEEE 802.11p [25] .

[2]CRC - Cyclic Redundancy Check.

Properties **WMAC3** and **WMAC4** are common in networking technologies, wireless technologies included. Property **WMAC3** (***Frame Order***) is imposed by the communication medium within a WnS, and results directly from the serialisation of frame transmissions on the shared transmission medium. Property**WMAC4** (***Local Full-Duplex***) specifies that the sender itself is also included in that ordering property, as a recipient. Property **WMAC4** should be secured at the lowest level, preferably, by the MAC sublayer itself.

Property **WMAC5** (***Bounded Omission Degree***) formalises for the communication channel the failure semantics introduced earlier, being $k \geq f_o$. This property is crucial to implement protocols yielding bounded termination times. The channel omission degree bound, k, on a WnS can be established given the error characteristics of the corresponding wireless communication environment [17] [35] [41] .

The ***Bounded Omission Degree*** property is one of the most complex properties to secure in wireless communications. Securing this property with optimal values and with a high degree of dependability coverage will require the use of multiple channels [40] . Although an innovative solution to this problem needs to be further investigated, it may also provide an effective defence against a class of malicious physical layer (PHY) attacks, such as radio jamming [48] [29] [40] .

The behaviour of a WnS in the time domain is described by the remaining properties. Property **WMAC7** (***Bounded Transmission Delay***) specifies a maximum frame transmission delay, which is T_{td} in the absence of faults. The value of T_{td} includes the medium access and transmission delays and it depends on message latency class and overall offered load bounds [37] [21] . The value of T_{td} does not include the effects of omission errors. In particular, T_{td} does not account for possible frame retransmissions, such as those foreseen at the MAC level of the IEEE 802.15.4 specification [26] . However, T_{td} may include the extra delays resulting from longer medium access delays, resulting from subtle side effects caused by the occurrence of periods of network inaccessibility.

A period of network inaccessibility is a disturbance that may be induced by external sources, such as electromagnetic interference or mobility, or by glitches in the MAC protocol operation, such as those resulting from the omission of a MAC control frame (e.g., beacon). Hence, nodes may experience a loss of connectivity within a WnS. However, the WnS cannot be considered failed. In fact, the WnS only enters into a temporary state where the communication service is not provided to some or all of the nodes. Therefore, the bounded transmission delay within a WnS includesT_{ina}, a corrective term, which accounts for the worst-case duration of inaccessibility glitches, given the bounds specified by property **WMAC6** (***Bounded Inaccessibility***). The inaccessibility characteristics depend and can be predicted by the analysis of the MAC protocol [41] .

Finally, property **WMAC8** (***Tightness***) defines the maximum interval between frame reception instants in different nodes. The value of T_{tight} depends on the medium propagation delay variance. The value of T_{tight} is also a function of the variance of the latency of the particular mechanism used to establish the frame reception instant. The use of emerging MAC controller technology permits to bound such latency to a reasonably small value (e.g., using an interrupt driven methodology) or can even be made negligible (e.g., using open core MAC built-in timestamping facilities). This property is particularly important for the implementation of synchronisation services, such as synchronised clocks.

2.2.2 Extending Dependability and Timeliness Properties to Multiple Wireless Segments

In this document, we are concentrated in the characterisation of wireless communications within the scope of a single WnS, which is more suitable to advanced safety-critical control applications.

However, we also describe strategies to extend dependability and timeliness properties to multiple WnS, which are derived from an analysis of the entire wireless protocol stack and its capabilities.

Multiple WnS are utilised to extend the coverage of the network, allowing the distribution and mobility of nodes within a wide geographical area. Nodes distributed in multiple WnS may not be able to sense each other, which request the establishment of communication routes to forward data and to support their communication.

The use of routing operations breaks properties of wireless communications offered natively within a single WnS such as ***Broadcast*** and ***Frame Order***. The extension of the ***Broadcast*** property requires additional dissemination protocols to support data transmission to multiple WnS. Extending the ***Frame Order*** property to multiple WnS is a more outstanding problem that requires the use of agreement protocols to establish a common order to deliver the received frames. The extension of ***Error Detection*** and ***Local Full-Duplex*** are MAC-level properties applied locally.

The extension of the dependability and timeliness properties ***Bounded Omission Degree***, ***Bounded Inaccessibility***, ***Bounded Transmission Delay*** and ***Tightness*** depends on the characteristics of each WnS included in the "path" between the farthest WnSs.

2.3 Enhancing Dependability and Timeliness in Wireless Communications

Existent wireless standards such as the IEEE 802.15.4 [26] , IEEE 802.11p [25] and WirelessHART [46] are not completely prepared to provide a timely and dependable real-time communication service, which is capable to address temporal requirements needed by real-time protocols and applications present within safety-critical environments [6] . Attempts to enhance situation described in the literature does not solve the problem completely [2] [10] [11] [30] [45] . This evidences the necessity of enhancements to enable an effective use of wireless communication technologies within environments addressed by the **KARYON** project.

The Resilient Real-Time Medium Access Control (**R2T-MAC)** architecture is utilised by the **KARYON** project team to establish a solid foundation in the provision of an efficient wireless real-time communication service, with enhanced dependability and timeliness characteristics. An overview of the **R2T-MAC** architecture is presented next.

2.3.1 An Overview of the Resilient Real-Time Medium Access Control (R2T-MAC) Architecture

R2T-MAC is an innovative solution to support the provision of real-time communication services on wireless networks**.** Extensible components constitute the core of this architecture, wrapping the MAC sublayer with a set of components to add temporal guarantees to its data transmission service. These components were designed to enhance the resilience of the MAC sublayer against disturbances in the network, which cause negative effects on wireless communications. As a consequence, **R2T-MAC** increases the timeliness and dependability of the MAC sublayer, being in compliance with wireless standards, such as IEEE 802.15.4 and IEEE 802.11p.

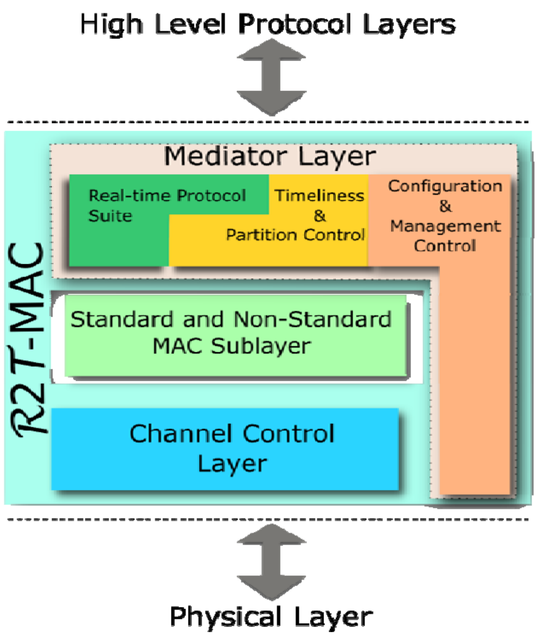

High Level Protocol Layers
R2T-MAC
Mediator Layer
Real-time Protocol Suite
Timeliness & Partition Control
Configuration & Management Control
Standard and Non-Standard MAC Sublayer
Channel Control Layer
Physical Layer

Figure 4 - R2T-MAC Architecture and its components

R2T-MAC provides the following characteristics: (a) resilience against disturbances in network; (b) supporting of data transmission with temporal restrictions; (c) flexibility and adaptability to environment restrictions and applications requirements; and (d) complete standard compliance.

Those characteristics are supported by the design and specification of two different layers: the Channel Control Layer and the Mediator Layer. Both are part of the R2T-MAC architecture, which wraps the standard MAC sublayer (and possibly proprietary non-standard MAC sublayers), as depicted in the diagram of

Figure 4.

2.3.1.1 Channel Control Layer

The *Channel Control Layer* was designed to enhance the monitoring and controlling operations over the RF circuitry, isolating the complexity and details of the hardware architecture from the remaining layers and communication protocols.

The *Channel Control Layer* is a thin layer designed to be implemented directly in hardware. The design principles of the *Channel Control Layer* allows the use of different MAC sublayers with different types of RF circuitries, which support the utilisation of a Multiple Input Multiple Output (MIMO) system and its advantages, e.g., spatial multiplexing, without the necessity of a specialised MAC sublayer to deal with the presence of multiple radios.

2.3.1.2 Integration of Selected Standard and Non-standard MAC sublayers

One of the main advantages of the **R2T-MAC** architecture is the possibility to integrate different MAC sublayers, which have potential to be utilised within the safety-critical environments addressed by the **KARYON** project. In this section, we will provide a brief overview of three different MAC sublayers: the IEEE 802.15.4, the IEEE 802.11p, and a non-standard TDMA-based MAC designed within the **KARYON** project.

IEEE 802.15.4 Standard: The IEEE 802.15.4 [26] is a wireless standard suitable to create wireless sensor and actuator networks (WSANs), which have potential utilisation on vehicular, industrial, and aero-spatial communications. Each IEEE 802.15.4 network must contain a coordinator that defines the network parameters and characteristics such as addressing, supported channels, and operation mode.

There are two operation modes defined in the standard: nonbeacon enabled and beacon enabled. The nonbeacon enabled mode uses a non-slotted version of the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol to control the network access. This control is decentralised, lacking a native support for communications with temporal restrictions.

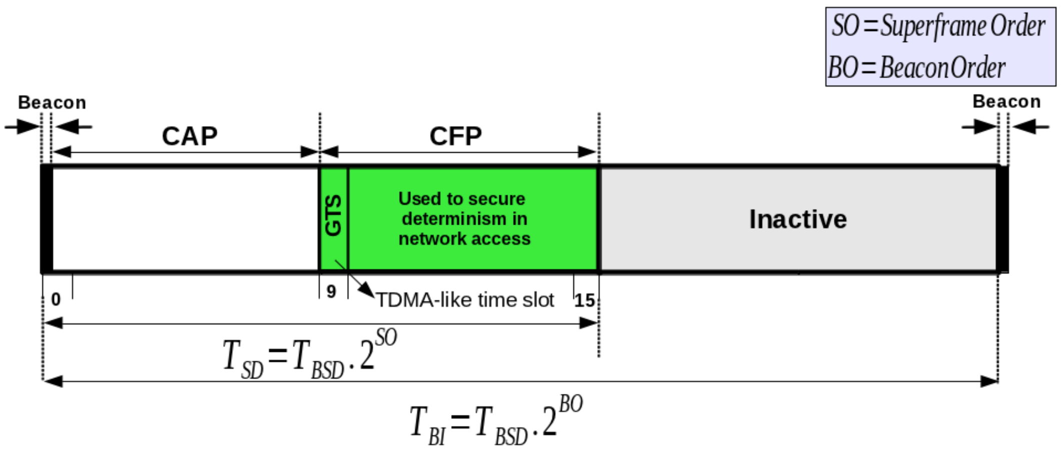

SO = Superframe Order
BO = Beacon Order
Beacon
Beacon
CAP
CFP
GTS
Used to secure determinism in network access
Inactive
0
9
TDMA-like time slot
15
$T_{SD} = T_{BSD} . 2^{SO}$
$T_{BI} = T_{BSD} . 2^{BO}$

Figure 5 - The IEEE 802.15.4 Superframe structure

Conversely, beacon enabled mode has a native specification for supporting communications with temporal restrictions. A coordinator controls the medium access using the superframe structure represented in Figure 5. The contention access period (CAP), contention free period (CFP), and the optional inactive period are the subdivisions of such structure, bounded by the transmission of a beacon frame used to synchronise nodes to access the network for communication purposes. In CAP all nodes compete equally for such access, using a slotted version of CSMA/CA protocol. Details of this protocol can be found in [26] [37] [28] .

On the other hand, the access to the network within CFP is supported by the allocation of reserved slots, which are called guaranteed time slots (GTS). This allocation "guarantees" that the network is free and nodes can transmit frames without use the CSMA/CA protocol. CFP was designed to support the execution of applications with real-time traffic restrictions [21] .

The inactive period may be utilised to reduce the energy consumption of nodes, during each transmission cycle denoted by superframe duration. The duration of a superframe is controlled by MAC attributes *macBeaconOrder* (BO) and *macSuperFrameOrder* (SO), where $0 \leq SO \leq BO \leq 14$. When *SO* and *BO* have the same value, the inactive period does not exist.

IEEE 802.11p Standard: The IEEE 802.11p [25] is an amendment of the traditional 802.11 wireless specification for local area networks, being proposed to address needs and requirements of networking communications on vehicular environments. The IEEE 802.11p operates in frequency bands around 5GHz to allow transmission over distances up to 1000 m, providing a coverage range useful for vehicle-to-vehicle (V2V), vehicle-to-roadside (V2R), and vehicle-to-infrastructure (V2I) communication patterns.

Particularly, the IEEE 802.11p provides a better support for dynamic environments where handoffs, disconnections, and communications problems caused by mobility of nodes may occur very often. The IEEE 802.11p also uses the CSMA/CA protocol to control the access to the network, being the characterisation of the transmission service similar with the IEEE 802.15.4. The main difference is the prioritisation of the network access by special control mechanisms defined in the IEEE 802.11 standards. These mechanisms establish strategies to provide support of traffic differentiation, prioritising the transmissions of some nodes instead of others (e.g., real-time traffic prioritisation).

Non-Standard MAC sublayers: As an example of integration of non-standard MAC sublayers in the R2T-MAC architecture, it is described an approach specifically suitable for providing enhanced timeliness and dependability characteristics, designed within the scope of the KARYON Project. Although the MAC protocol is non-standard, we focus on implementations that consider basic radio and clock settings.

Self-stabilizing communication, synchronization and ranging without external reference in large-scale multi-hop ad hoc networks

During the first two years, we have focused on the communication fundamentals that are related to medium access control for data communication and ranging devices without external reference. We present a prototype implementation that aims at increasing the degree predictability and resilience of wireless ad hoc and embedded networks. These components are fundamental for supporting the entire system, as Figure 6 illustrates. In this figure, we highlight in black and bold the technologies on which we focused during the first two years, namely (1) algorithms for network medium access control and ranging (localization) without external reference, and (2) algorithms for increasing synchrony among nearby transmitting stations. This description also considers the preliminary integration of these algorithms in large-scale multi-hop networks. The work during year 1 on items (1) and (2) considered these items separately, and thus during year 2 we worked on their integration as one algorithm as well as their implementation in two systems. The first system is a sensor network with ZigBee radio units and the second is a network of ranging devices that facilitate localization. The result includes a prototype implementation. Specifically, we have used the ZigBee radio system for showing that ad hoc networks can have a greater degree of predictability than the existing implementations for ad hoc networks. Moreover, this pilot implementation shows a very low pack drop rates. This is an important input for cooperative systems and for localization systems that uses ultra wideband ranging (UWB), e.g., ranging and communications module (RCM) radio devices for localizing the position of the vehicles. In the area of localization, we also study the trade-off between accuracy and delay in cooperative UWB localization. In addition, several reports given in the annex and submitted publications are pointed out. Next, we detail the algorithmic developments during year 2 (for which matching implementation will be presented in WP5), before discussing its extensions to large scale multi-hop networks.

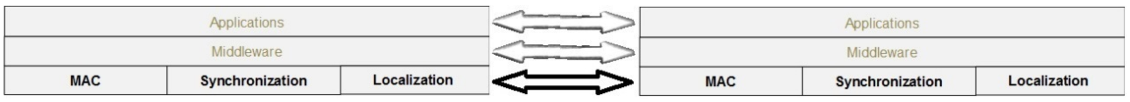


Figure 6 - The system structure of the supporting technology perspective.

Autonomous and cooperative systems will ultimately carry out risk-related tasks, such as piloting driverless cars, and liberate humankind from mundane labour, such as factory and production work. Note that the implementation of these cooperative systems implies the use of wireless ad hoc networks and their critical component – the medium access control (MAC) layer. Since cooperative systems operate in the presence of people, their safety requirements include the provision of real-time guarantees, such as constant communication delay. Infrastructure-based wireless networks successfully provide high bandwidth utilization and constant communication delay. They divide the radio into timeslots of uniform size, ξ, that are then combined into frames of uniform size, τ. Base-stations, access points or wireless network coordinators can schedule the frame in a way that enables each node to transmit during its own timeslot, and arbitrate between nearby nodes that wish to communicate concurrently. We strive to provide the needed MAC protocol properties, using limited radio and clock settings, i.e., no external reference for collision detection, time or position. For these settings, we demonstrate that there is no solution for the studied problem when $\tau<\max((2 - e)\delta, \chi 2)$, where $e > 0$, δ is a bound on the node degree, and $\chi 2$ is the chromatic number for distance-2 vertex colouring. Note that $\chi 2 = \delta + 1$ and $\chi 2 = 5\delta/3 + O(1)$ for the cases of tree, and respectively, planar graphs [33] .

The main result is the existence of probabilistic collision-free self-stabilizing TDMA algorithms that have constant communication delay of $\tau>\max(4\delta, X_2)+1$, where $X_2 \geq \chi 2$ is a number that depends on the colouring algorithm in use. In the context of self-stabilization systems that have no external reference, we are the first to study this problem (to the best of our knowledge).

Wireless ad hoc networks have a dynamic nature that is difficult to predict. This gives rise to many fault-tolerance issues and requires efficient solutions. These networks are also subject to transient faults due to temporal malfunctions in hardware, software and other short-lived violations of the assumed system settings, such as changes to the communication graph topology. We focus on fault-tolerant systems that recover after the occurrence of transient faults, which can cause an arbitrary corruption of the system state (so long as the program's code is still intact). These self-stabilizing [15] design criteria simplify the task of the application designer when dealing with low-level complications, and provide an essential level of abstraction. Consequently, the application design can easily focus on its task – and knowledge-driven aspects.

ALOHAnet protocols [1] are pioneering MAC algorithms that let each node select one timeslot per TDMA frame at random. In the Pure Aloha protocol, nodes may transmit at any point in time, whereas in the Slotted Aloha version, the transmissions start at the timeslot beginning. The latter protocol has a shorter exposure period during which packets may collide, because each transmission can collide only with transmissions that occur within its timeslot, rather than with two consecutive timeslots as in the Pure Aloha case. Note that the random access approach of ALOHAnet cannot provide constant communication delay. Distinguished nodes are often used when the application requires bounded communication delays, e.g., IEEE 802.15.4 and deterministic self-stabilizing TDMA [4] [31] . Without such external references, the TDMA algorithms have to align the timeslots while allocating them. Existing algorithms [12] circumvent this challenge by assuming that $\tau /(\Delta + 1) \geq 2$. This guarantees zero exposure period with respect to at least one timeslot, s, and all transmissions from transmitters that are at most two hops away, where Δ is a bound on their number. However, the $\tau /(\Delta + 1) \geq 2$ assumption implies bandwidth utilization that is up to $O(\delta)$ times lower than the proposed algorithm, because $\Delta \in O(\delta^2)$ (also for planar graphs [3]).

KARYON Contribution: As a basic result, we show that $\tau /\delta \geq 2$, and as a complement to this lower bound, we focus on considering the case of $\tau/\delta \geq 4$. We present a probabilistic collision-free self-stabilizing TDMA algorithm that has constant communication delay of τ, which is equivalent to $\mathcal{T}_{\text{td}}$, following the MAC-level properties specified in section 2.2.1. We show that it is sufficient to guarantee zero exposure period with respect to a single timeslot, s, and a single receiver, rather than all neighbours. This narrow opportunity window allows control packet exchange, and timeslot alignment. After convergence, there are no collisions of any kind, and each frame includes at most one control packet.

Related Work: The first proposal for a self-stabilizing TDMA algorithm for wireless ad hoc networks [22] considers external time reference for dividing the radio time. Simulations are used for evaluating the heuristics of MS-ALOHA [39] for dealing with timeslot exhaustion by adjusting the nodes' individual trans- mission signal strength. We provide analytical proofs and consider basic radio settings. The MAC algorithms in [44] [49] assumes the accessibility of an external time or geographical references or the node trajectories, e.g., Global Navigation Satellite System (GNSS). We instead base the TDMA timeslot alignment on self-stabilizing clock synchronization algorithms for wireless ad hoc networks [23] .

We consider minimal transmission duration, uniform packet priority, and a predefined frame size, τ, as in [32] . This assumption holds whenever the node degree is bounded by a known constant, the node distribution is sparse, or when nodes can adjust their transmission power in overpopulated areas. The system designer may decide to abandon real-time guarantees, and prefer the participation of all nodes over a uniform communication delay. Such design alternatives are beyond the scope of this work.

Extensions: The studied TDMA algorithm considers the reception of packets from all neighbouring nodes in every round and allows the transmission of a constant number of packets. This raises contention-related questions when considering multi-hop networks. As an extension, we consider the use of existing network coding techniques and designed the needed self-organizing algorithms for using them.

In existing computer networks, information is transmitted from the sender to the receiver through a set of intermediary nodes, which are responsible for forwarding the data in order to deliver it to the final destination. Normally, information received by intermediary nodes is stored and forwarded, this method known as store-and-forward. In general, computer networks rely on routing schemes, which allow the nodes in the network to select the right destination when a packet needs to be sent or forwarded. Wireless networks allow the transmission of one packet at a time that is broadcasted to all. Therefore, we aim at developing forwarding algorithms it is an advantage this property. We focus on network coding. These techniques break with the traditional paradigm of routing, in the sense that the packets are no longer needed to be treated as untouchable atomic packets since network coding permits the packets to be encoded using algebraic combinations with the aim of saving bandwidth. Therefore, when using network coding, the intermediary nodes have a more important task than merely acting as switches that receive information from an input link and then relay that information to an output link or set of output links.

One of the algorithms [8] [9] that uses network coding solves the problem of k-token dissemination in dynamic networks by means of network coding. The problem is defined as follows: initially there are a total number of k tokens in the network, some/all of these tokens are hold by one or more nodes. The main goal is that at the end of the execution of the algorithm all participating (correct) nodes must possess the same set of k tokens. As an extension to our work, we propose in [20] how to use Haeupler's algorithm [8] [9] . Namely, each message is broken down to packets such that each packet represents a token. The algorithm in [20] proposes how to do that when the network is dynamic, message can be dropped and nodes can fail.

Cooperative localization and the trade-off between accuracy and delay: We have implemented a system in which vehicles can use external references and vehicle to vehicle ranging for cooperatively finding their location. We base our system on ultra-wide bandwidth (UWB) ranging devices that allow for accurate positioning in environments where global navigation satellite systems may fail, especially when complemented with cooperative processing. While cooperative UWB has led to centimetre-level accuracies, the communication overhead is often neglected. We quantify how accuracy and delay trade off in a wide variety of operation conditions. We also derive the asymptotic scaling of accuracy and delay, indicating that in some conditions standard cooperation offers the worst possible trade-off. Both avenues lead to the same conclusion: indiscriminately targeting increased accuracy incurs a significant delay penalty. Simple countermeasures can be taken to reduce this penalty and obtain a meaningful trade-off between accuracy and delay, see the Annex A.1.1 for more details.

In annex:
(1) reprint of the paper "***Self-stabilizing TDMA Algorithms for Wireless Ad-Hoc Networks without External Reference***". T. Petig, E. M. Schiller, and P. Tsigas. In Stabilization, Safety, and Security of Distributed Systems: Proceedings of 15th International Symposium, SSS 2013. Lecture Notes in Computer Science. November 2013, Osaka, Japan.
(2) preprint of the technical report: "**On the Trade-off Between Accuracy and Delay in Cooperative UWB Localization: Performance Bounds and Scaling Laws**" Gabriel E. Garcia, Student Member, L. Srikar Muppirisetty, Elad M. Schiller, and Henk Wymeersch (submitted for publication)
(3) preprint of the technical report: "**Design of a Session Management Algorithm for Solving K-Token Dissemination Problem Using Network Coding**" Guillermo Barredo García and Iosif Salem

Conclusions: This work considers fault-tolerant systems that have basic radio and clock settings without access to external reference for collision detection, time or position, and yet require constant communication delay. We study collision-free TDMA algorithms that have uniform frame size and uniform times- lots and require convergence to a data packet schedule that does not change. Our analysis considers the timeslot allocation aspects of the studied problem, together with transmission timing aspects. Interestingly, we show that the existence of the problem's solution depends on convergence criteria that include the ratio, τ/δ, between the frame size and the node degree. We establish that $\tau/\delta \geq 2$ as a general convergence criterion, and prove the existence of collision-free TDMA algorithms for which $\tau/\delta \leq 4$. Unfortunately, our result implies that, for our systems settings, there is no distributed mechanism for asserting the convergence criteria within a constant time. For distributed systems that do not require constant communication delay, we propose to explore such criteria assertion mechanisms as future work. Moreover, as an extension we propose propagation methods that take the advantage of broadcasting in wireless networks.

2.3.1.3 Mediator Layer

Mediator Layer is an extensible component layer, which initially provides features such as reliable and real-time frame transmissions, controlling of temporary partitions, and control and management of MAC sublayer configurations. This layer is immediately above the MAC sublayer, and is responsible to enhance the MAC sublayer services provided to high level protocols and applications. The foregoing features are provided by the initial set of components formed by the *Real-Time Protocol Suite*, *Timeliness & Partition Control*, and *Configuration & Management Control*.

The *Real-Time Protocol Suite* is responsible for handling data transmissions. This component enhances the frame transmission service provided by the MAC sublayer, establishing a foundation to offer a set of different service guarantees, with respect to reliability and timeliness, such as temporal bounds for frame transmissions.

The *Real-Time Protocol Suite* can be utilised to handle and support different protocols, which can serve requests with different types of requisites, augmenting the applicability of MAC sublayers on different areas with different requirements, namely on those with strict real-time demands, such real-time control and monitoring. Everything related with data frame transmissions are managed by this component, which may include the interaction with the other *Mediator Layer* components to adjust specific parameters required by the transmission protocol, and to perform temporal control of the data transmission.

The *Timeliness and Partition Control* deals with the temporal aspects related to the data transmission service, controlling and monitoring the timing of the actions within the *Mediator Layer* using an internal *Temporal Watchdog*, which also controls the temporal execution of the transmission protocols utilised by the *Real-Time Protocol Suite* component. The *Timeliness and Partition Control* component also monitors the MAC sublayer actions, using a Partition Handler to detect the occurrence and to be aware of any partitioning incidents. The integration between the *Temporal Watchdog* and the *Partition Handler* Timer allows the use of optimal timeout values even in the presence of periods of network inaccessibility. Timeout values are automatically extended in this case, thus avoiding premature and equivocal error propagation to other components and to higher layers.

The *Configuration and Management Control* controls the configuration of all internal components of the **R2T-MAC** architecture, including also the controlling of the MAC sublayer configurations. This controlled configurations are stored within an information base, which is accessed directly by the *Configuration and Management Control* component through an internal Configuration Manager. The performed configurations have to be validated to respect realistic application requirements, resource limitations, and environment restrictions.

The *Configuration and Management Control* component is also responsible to manage the network configurations, providing a distributed service to establish a consistent membership view of the nodes present within the WnS. This component makes the *Mediator Layer* (self-)adaptive, and (self-)managed, allowing the possibility to change its internal state and its configuration profile according to the dynamics of environment conditions and temporal requirements of requested transmissions.

A description of a Mediator Layer architecture targeting IEEE 802.15.4 wireless networks is provided in Annex A.1.2.

In annex: reprint of the paper "***Towards Resilient Real-Time Wireless Communications***", J. L. R. Souza and J. Rufino, Proceedings of the 25th Euromicro Conference on Real-Time Systems (ECRTS-WiP 2013), July 2013.

2.3.2 R2T-MAC Service Interface

The **R2T-MAC** interacts with high layer protocols and applications though an exposed service interface, which allows the utilisation of the communication service to perform data transmissions.

The aforementioned exposed service interface, dubbed *Data Service Interface* provides three different primitives utilised to request and confirm a data transmission and to indicate a data delivering. The primitives of the *Data Service Interface* are presented in Table 2.

Data Service Interface	
***MLA**.Data.Request*	It allows above layers, its protocols and applications, to request a data transmission service to **R2T-MAC**. This service can be reliable/unreliable, where optional parameters passed to that primitive can be utilised to request the use of a specific protocol.
***MLA**.Data.Confirm*	It provides a local confirmation to signal that a previous request was transmitted successfully by the **R2T-MAC**. It is important to highlight that this confirmation does not have any dependency with operations performed by a remote entity (e.g., an indication of a successful data delivery to a remote wireless node).

***MLA**.Data.Indication*	It delivers data to above layers, which is the result of internal **R2T-MAC** data reception through the PHY interface.

Table 2 - Primitives of the R2T-MAC Data Service Interface

The data transmission services provided through the *Data Service Interface* are built on top of MAC sublayer primitives, enriching the unreliable data transmission service provided in that level. Table 3 presents the primitives present in the MAC sublayer data transmission service interface.

MAC data transmission service interface	
***MAC**.Data.Request*	It provides a way to request a data transmission to the MAC layer. **(Unreliable transmissions only)**
***MAC**.Data.confirm*	It provides a local confirmation that a frame has been sent to the medium. **(Does not provide any guarantee of delivery at the destination)**
***MAC**.Data.indication*	It provides notification about an arrived data frame.

Table 3 - Standard MAC layer primitives for data transmission

The introduction of the R2T-MAC services (Table 2) offers an enriched service interface when compared with the native MAC service interface (Table 3) since R2T-MAC incorporates the capability to cope with low-level issues. In particular, R2T-MAC deals with: the occurrence of disturbances in the medium and MAC protocols, including periods of network inaccessibility; runtime (re)configuration of the MAC sublayer parameters to adapt to different conditions.

2.3.3 Current Practical Results

The IEEE 802.15.4 standard was utilised as a case study in enhancing the dependability and timeliness of a wireless network operation. One key issue concerns the analysis of the network specification to characterise its temporal properties and the definition of effective policies to reduce the longest periods of network inaccessibility. The results from this work were applied in the design and development of an analytical tool able to calculate the worst (and the best) case durations for the periods of inaccessibility. Concerns related with the verification of the inaccessibility analysis model, have motivated the design and development of tools using both simulation and experimental evaluation approaches to analyse network operation and its behaviour under error conditions.

Analytical, simulation-based, and experimental tools are then utilised in a complementary manner, comparing and cross validating the results obtained by each one of these tools, separately.

2.3.3.1 Definition and Results of Inaccessibility Reduction Policies in IEEE 802.15.4

The study of IEEE 802.15.4 network inaccessibility in [41] has provided the worst (and best) case durations for the occurrence of inaccessibility events, as summarized in Figure 7, which presents the normalised worst-case durations for a comprehensive set of relevant scenarios. It is worthwhile noticing that in the worst-case scenarios (Orphan and Re-Association) the duration of inaccessibility events reach durations 70 times higher than the value of the network cycle, referenced as T_{BI} (the beacon interval). This huge difference evidences the need to reduce network inaccessibility, enhancing the temporal guarantees offered by IEEE 802.15.4 networks.

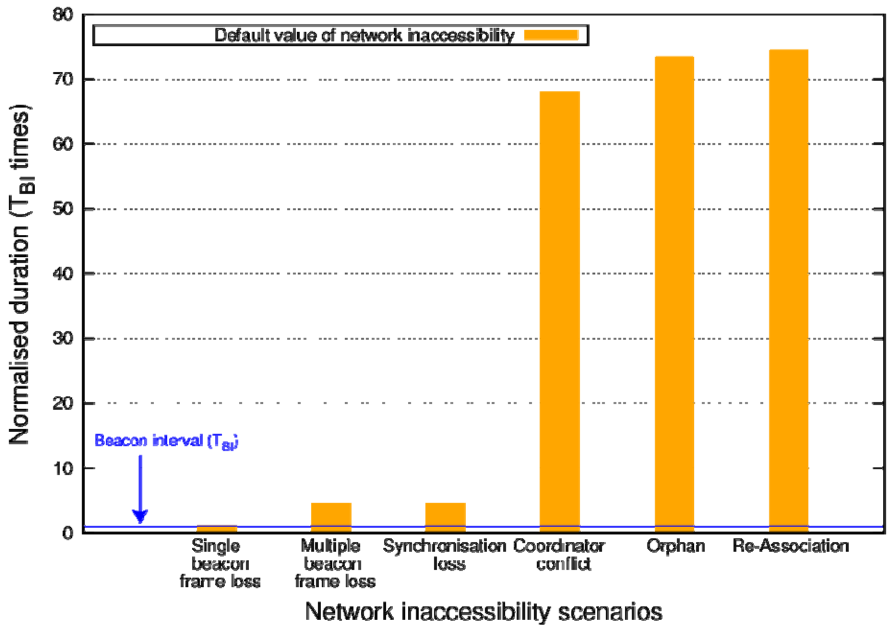


Figure 7 - Normalised duration of network inaccessibility on IEEE 802.15.4 networks

For self-containment purpose, a brief description of the causes of inaccessibility of the scenarios addressed in Figure 7 are presented in Table 4, concerning the relevant network inaccessibility scenarios right on target for reduction of their duration.

Scenario	Related Cause
Single and Multiple Beacon Frame Loss Synchronisation Loss	Loss of beacon frame(s).
Coordinator Conflict	Entrance of other coordinator within the WnS.
Orphan and Re-Association	Actions to re-establish the network operation after a synchronisation loss (including channel scan).

Table 4 – Brief description about relevant network inaccessibility scenarios

A summarised description of each reduction policy is presented in Table 5. The *Coordinator conflict policy* completely eliminates the network inaccessibility scenario. The use of a unique compound network identifier is a simple and quite efficient solution to such accomplishment.

Channel utilisation awareness policy restricts the number of channels that have to be scanned to search and find the network coordinator.

Network dependability awareness policy does not have a direct impact in the reduction of network inaccessibility, but it establishes a uniform way to use the local omission degree accounting, i.e., the number of consecutive omission failures detected by a node evaluate whether or not a given bound has been exceeded.

The *Logical channel diversity* policy uses the omission degree accounting mechanism — from the *Network dependability awareness* policy — to check if the communication channel has exceeded the omission degree bound, and then a switch operation to a new channel have to be done. This switch operation is fundamental for a fast re-establishment of networking communications, reducing then even more the duration of the Orphan and Re-Association scenarios.

Policy name	**Problem definition**	**Policy description**
Coordinator conflict avoidance	In the IEEE 802.15.4 standard different coordinators may transmit beacons with same *networkID*, creating a coordinator conflict.	Utilisation of a 2-Tuple (*networkID*,*coordinatorID*) as a unique compound network identifier.
Channel utilisation awareness	When a node loses synchronisation, a node always scans all available channels looking for the coordinator	The channel scan is restricted to channels where the likelihood of finding the coordinator is high
Network dependability awareness	Network dependability is not expressed	Channel omission degree is evaluated
Logical channel diversity	In the presence of faults, a logical channel may experience an increase on channel omission degree	If a logical channel exceeds the allowed omission degree, a node may switch to a different logical channel

Table 5 – A summary of reduction policies

The reduction policies achieve better results if applied using a cumulative approach. In such approach, the results obtained following the *Logical channel diversity* policy incorporate contributions from all policies applied before, as illustrated in Figure 8. Details regarding all aspects of the reduction policies can be found in Annex A.1.5.

In annex: reprint of the paper "***Analysing and Reducing Network Inaccessibility in IEEE 802.15.4 Wireless Communications***", J. L. R. Souza and J. Rufino. In Proceedings of the 38th IEEE Conference on Local Computer Networks (LCN 2013), October 2013.

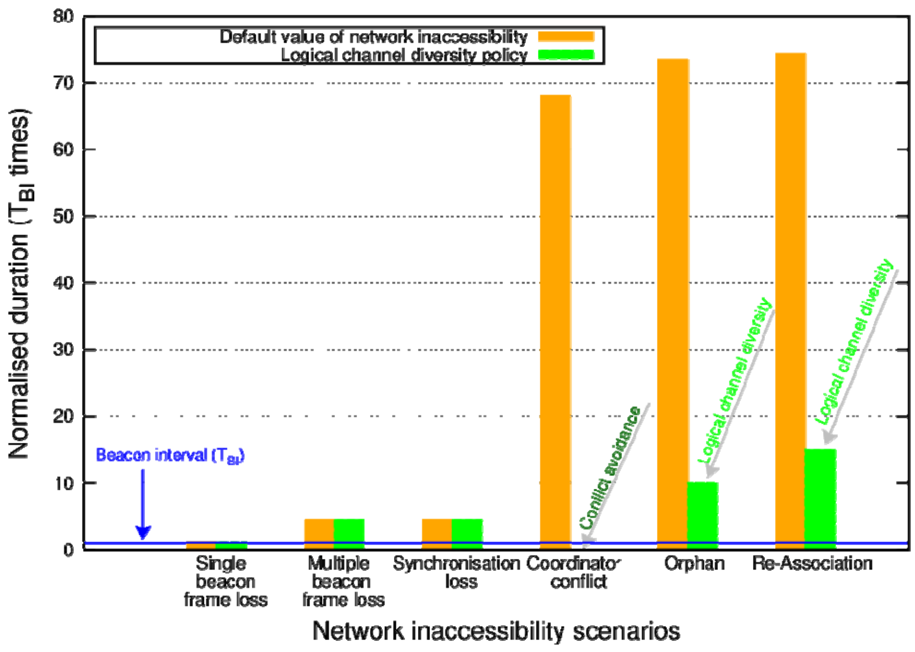


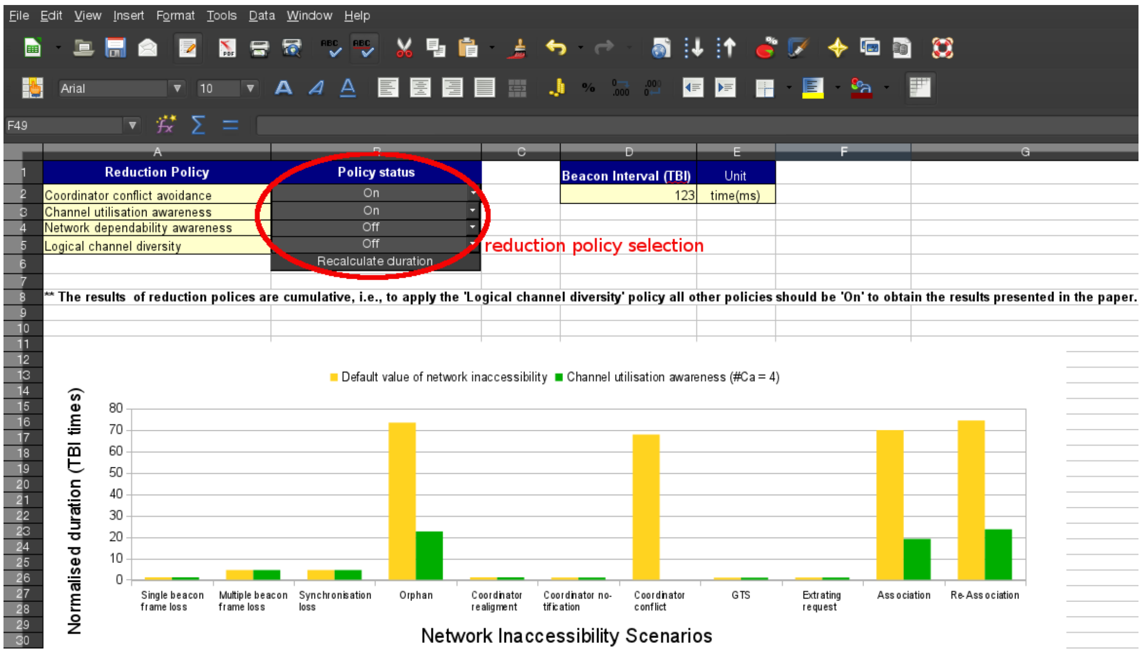
Figure 8 - Results from a cumulative approach in using reduction policies

2.3.3.2 IEEE 802.15.4 Network Inaccessibility Analysis Tool

We designed and developed an analytical tool to evaluate the duration of network inaccessibility scenarios as draw from the IEEE 802.15.4 standard. The analytical tool is a spreadsheet built-in, based on the LibreOffice suite (http://www.libreoffice.org/).

The tool helps the theoretical analysis of network inaccessibility for different network configurations of the IEEE 802.15.4 wireless standard. The tool, presented with more detail in the Deliverable D3.3 document is an open source tool available under a GNU General Public License (GPL) version 3, which can be downloaded at: http://www.karyon-project.eu/wp-content/uploads/2012/10/Inaccessibility_IEEE802.15.4_Beacon-enabled-Karyon.ods.

For completeness, Figure 9 illustrates a typical use of the tool. There are different tabs, each one designed to represent constants, parameters, and configurations that can be performed on a standard compliant IEEE 802.15.4 wireless network. The duration of network inaccessibility scenarios are evaluated and visualised as values in milliseconds (ms), or as normalised values by T_{BI} units of time.



Reduction Policy	Policy status
Coordinator conflict avoidance	On
Channel utilisation awareness	On
Network dependability awareness	Off
Logical channel diversity	Off
	Recalculate duration

Beacon Interval (TBI)	Unit
123	time(ms)

reduction policy selection

** The results of reduction polices are cumulative, i.e., to apply the 'Logical channel diversity' policy all other policies should be 'On' to obtain the results presented in the paper.

Figure 9 - Network inaccessibility evaluation tool.

2.3.3.3 Enhancement of the NS-2 Simulator Tool

Enhancements in the IEEE 802.15.4 NS-2 module were two-fold. On one hand, one aim to provide a better support for the emulation of networks with real-time requirements, through the incorporation of a functional contention free period (CFP) to the NS-2 IEEE 802.15.4 module, which uses a TDMA approach to control the access to the network. The original IEEE 802.15.4 distributed with the NS-2 simulator did not support these features. Management operations defined within the IEEE 802.15.4 standard, but not implemented on the original NS-2 IEEE 802.15.4 module, were also added to such module. Table 6 summarises the differences between the original NS-2 IEEE 802.15.4 module and its enhanced version created by the **KARYON** team.

MAC Management Action	IEEE 802.15.4 Standard Behaviour	NS-2 Original Module	NS-2 enhanced Module
Coordinator conflict	If two coordinators transmit beacons with the same network identifier, a conflict occurs	not implemented	Implemented
Orphan	A request is issued to start a channel scan action	not functional	Operational
Coordinator realignment	On orphan notification, the acknowledgement of a realignment is required	not functional	Operational
Channels available to scan	16 channels on 2.4Ghz	only scaned the first 3 channels	16 channels
Channel scan duration attribute	defines the wait period for detecting the coordinator	the definition was incorrect	in compliance with the standard
Network Information Base (NIB) attribute	Management Entity checks MAC/PHY layer attributes	no verification was performed	in compliance with the standard

Table 6 - NS-2 IEEE 802.15.4 Module behaviour comparison

A detailed description of the enhancements introduced in the NS-2 simulator can be found in the article reprint provided in Annex A.1.6.

In annex: reprint of the paper "**Improving NS-2 Network Simulator for IEEE 802.15.4 standard operation**". A. Guerreiro, J. L. R. Souza, J. Rufino, In 5th Simpósio de Informática (INFORUM), Évora, Portugal, September 2013.

Other important result achieved by the KARYON team was the design and incorporation of a fault injector and a temporal analysis component within the NS-2 simulator. The main goal of such tools is the analysis and evaluation of IEEE 802.15.4 wireless networks under error conditions. With this goal in mind, two components were added to the NS-2 platform to be used by IEEE 802.15.4 module: a temporal analysis tool and a fault injector. The integration of these components is illustrated in Figure 10.

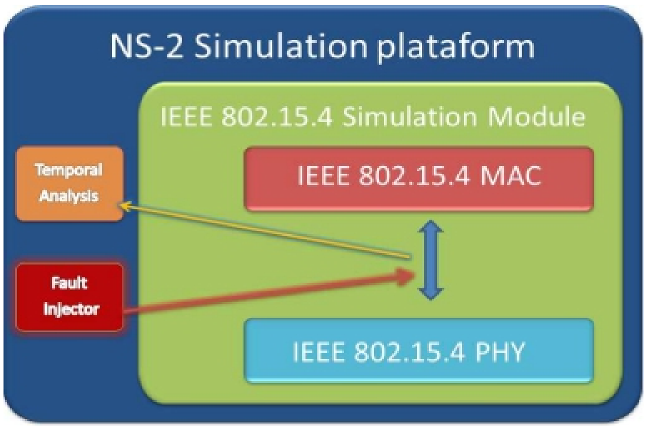


Figure 10 - New Features in IEEE 802.15.4 module.

The KARYON NS-2 fault injector is capable to use a fault pattern to introduce within networking communications, during the simulation. The criteria to define a fault pattern are totally configurable, allowing the definition of deterministic or probabilistic fault patterns. The fault injection scheme is depicted in Figure 11.

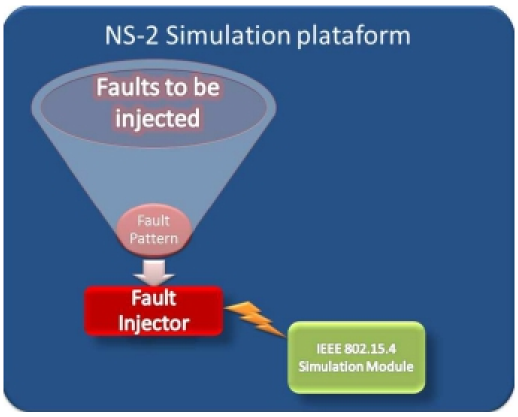


Figure 11 - Fault injector scheme.

The fault injector can be customised regarding the type of frame to be corrupted, the injection rate, and the duration of the fault injection campaign. Random noise or interferences are simulated according a random function implemented in the fault injector. A detailed description of the temporal analysis and fault injection tool used within the NS-2 simulator to perform the verification of network inaccessibility results is provided in Annex A.1.7.

In annex: reprint of the paper "***Improving NS-2 Network Simulator To Evaluate IEEE 802.15.4 Wireless Networks Under Error Conditions***", A. Guerreiro, J. L. R. Souza, J. Rufino. To appear in 3th International Conference on Sensor Networks (SENSORNETS), Lisbon, Portugal, Jan. 2014.

Next, we illustrate some results than can be achieved with the extensions introduced in the NS-2 simulator. First, we address the extensions concerning the incorporation of the contention free period. The following metrics are used for evaluation:

- *Data Frame Delivery Ratio (DFDR)*, which is the ratio between the total number of frames received in MAC sub-layer and the total number of data frame transmit requests during the simulation period. In our simulation, we consider the data frames transmit requests issued by all the nodes but the coordinator.

$$DFDR = \frac{Total\ Data\ frames\ received\ x\ 100}{Total\ Data\ frames\ transmitted}$$

- *Latency*, which represents the duration of a data frame transfer between nodes within the same WnS. For each individual data frame transfer, the frame transfer latency represents the interval between the instant when the data frame transmit request is issued (*TtxData*) and the instant of the corresponding data frame reception (*TrxData*). This metric includes the data frame processing and queuing time at the nodes, the data frame transmission time and the back off interval (if applicable). The average latency can then be calculated over all successful end-to-end transmissions within the simulation run.

$$AverageLatency = \frac{\sum_{allreceivedframes}(TrxData - TtxData)}{Total\ number\ of\ received\ frames}$$

On the other hand, the worst case value is given by:

$$WorstCaseLatency = max_{allreceivedframes}\ (TrxData - TtxData)$$

- *Energy*, the energy model present in NS-2 is used to calculate the amount of energy consumed by the nodes during the simulation time.

$$EnergyUsedPerNode = \frac{Total\ energy\ used\ on\ the\ simulation}{Number\ of\ nodes}$$

- *Throughput*, it measures the amount of data successfully received by the destination node within certain period of time.

$$Throughput = \frac{frames\_received\ x\ frame\_size}{Simulation\ Time}$$

We use the developed support for CFP and the defined metrics to evaluate the effectiveness of these mechanisms in support of deterministic network access times. Table 7 presents the setup of the parameters values utilised within the simulation.

Simulation Parameters	
NS-2 Version	2.35 updated with GTS features
Network topology	Star topology
Traffic	Constant Bit Rate (CBR)
Reception range	$15m$
Carrier Sense range	$15m$
Packet size	$70Bytes$
CAP transmission type	Direct, using CSMA/CA
CFP transmission type	GTS transmission
Transmission/Reception power	$30mW$
Beacon	Enabled
Beacon order (BO)	3
Superframe Order (SO)	3
Maximum CSMA/CA attempts	4
Simulation time	$600s$

Table 7 - Simulation parameters for analysis of CFP behaviour

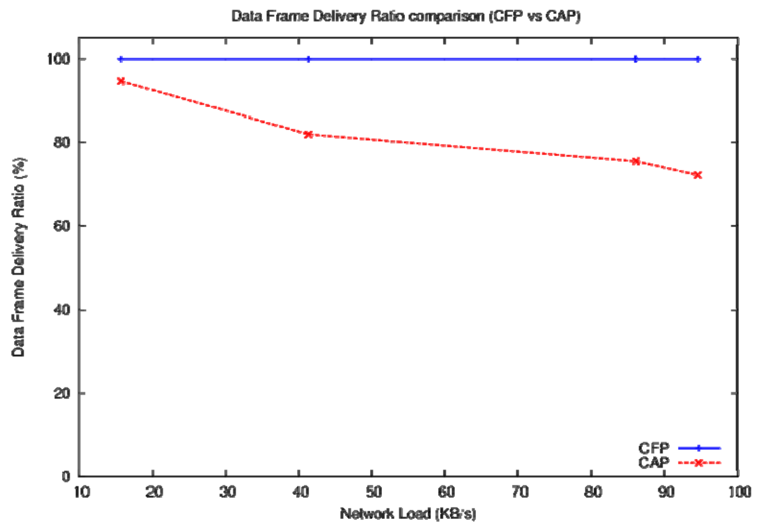


Figure 12 - Data Frame Delivery Ration comparison on transmissions during CAP and CFP

Figure 12 represents the delivery ratio on the network, providing a comparison of the results achieved for transmission requests issued during the CAP and CFP periods. During CFP, nodes use the allocated GTS and get direct and exclusive network access, which allows to achieve about 100% delivery rate. In CAP the delivery ratio drops in function of the increase in the network load. This is explained by the occurrence of collisions during CAP, or due to the number of nodes attempting to access the medium. In the CSMA/CA protocol a data frame transmit request is dropped, if the number of transmission attempts exceed a given threshold defined by the IEEE 802.15.4 standard. This value represents the maximum number of backoffs the CSMA/CA algorithm will attempt before declaring a channel access failure.

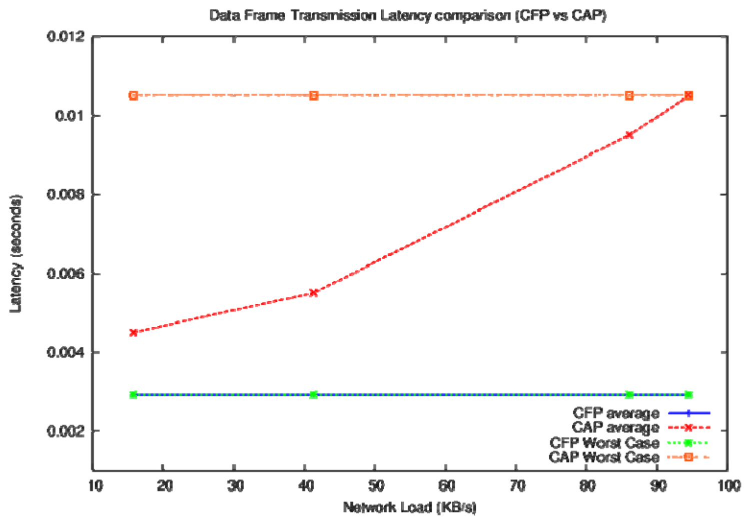


Figure 13 - Latency comparison on data frame transmissions performed within CAP and CFP

Figure 13 shows the latency comparison between a data frame transmission using CFP and CAP. While the latency remains almost constant when data frames are transmitted during CFP, using allocated GTS, the latency highly increases while using CAP. The constancy achieved in data frame transfer times during CFP is a sign of determinism and predictability and shows in the diagram of Figure 13 in two ways: an (almost) constant worst-case data frame transmission latency; the optimal value of this latency, which does not exceed 0.002936 seconds. This is due to nodes during CFP get exclusive network access, meaning nodes do not have to check if the media is idle and no collisions occur for those nodes. These results show the importance of the GTS mechanism in applications with real-time requirements on which deterministic data frame transmission times are mandatory. Additionally, the data frame transmission latency increases in CAP, up to the worst-case value of 0.010512 seconds, given the worst-case network load in the simulation setup.

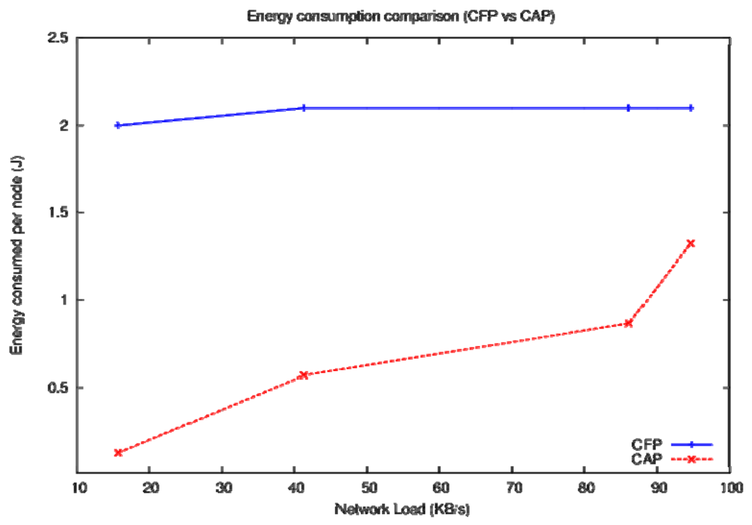


Figure 14 - Per node energy consumption for data transmissions within CAP and CFP

Finally, Figure 14 represents the average energy consumption by the nodes during the simulation period. It worth noticing that the energy consumption increases when using CFP in comparison with CAP as result of required beacon frame reception tracking by the node, an action that obliges the node to switch-on its transceiver during the active period of every superframe instance. Contention-based access is more efficient under light network loads, whereas contention-free access becomes preferable when the background network load increases.

Network simulation results under error conditions: After setting up the simulation environment, an simulation script was executed in order to achieve the duration of the best and worst cases for the inaccessibility scenarios, the best case is achieved if the network is able to recover in the first attempt and the worst case represents the contrary.

Table 8 represents a comparison between the worst-case values of network inaccessibility obtained through the theoretical analysis, with the worst-case values obtained from simulations.

Figure 15 presents the inaccessibility durations events in comparison with a normal data transmission. Thus, analysing the Figure 15 we observe the increasing of the inaccessibility duration values mainly for the beacon loss related scenarios. The Orphan Node is by far the event with the most impact on the network. The graphic show a very high value to meet the requirements of most hard real-time applications. If the beacon interval is reduced, the gap between normal network access times and the periods of network inaccessibility may become even higher and the overall system predictability, timeliness and dependability properties may be at risk.

Scenario	Theoretical	Simulated
Single Beacon Frame Loss	0.262	0.245
Multiple Beacon Frame Loss	1.045	0.737
Synchronisation Loss	1.045	0.737
Orphan node	9.527	8.148

Coordinator Realignment	0.150	0.017
GTS request	0.120	0.001
Coordinator Conflict Detection	0.126	0.001
Coordinator Conflict Resolution	8.354	8.052
Association	8.605	7.717
Re-Association	9.647	8.270

Table 8 - Theoretical vs. Simulated - Worst-case comparison (BO=SO=4, T_{BI}=0.240*s*).

The same network inaccessibility event has the same observed duration at all nodes, which experienced such event within a wireless network segment. Thus, the number of nodes within a simulation does not influence the accuracy and correctness of our results. All the simulated results are lower than the theoretical, this can be explained by deterministic behaviour of the network simulator, and the fact that the implementation is not fully compliant with the IEEE 802.15.4 standard.

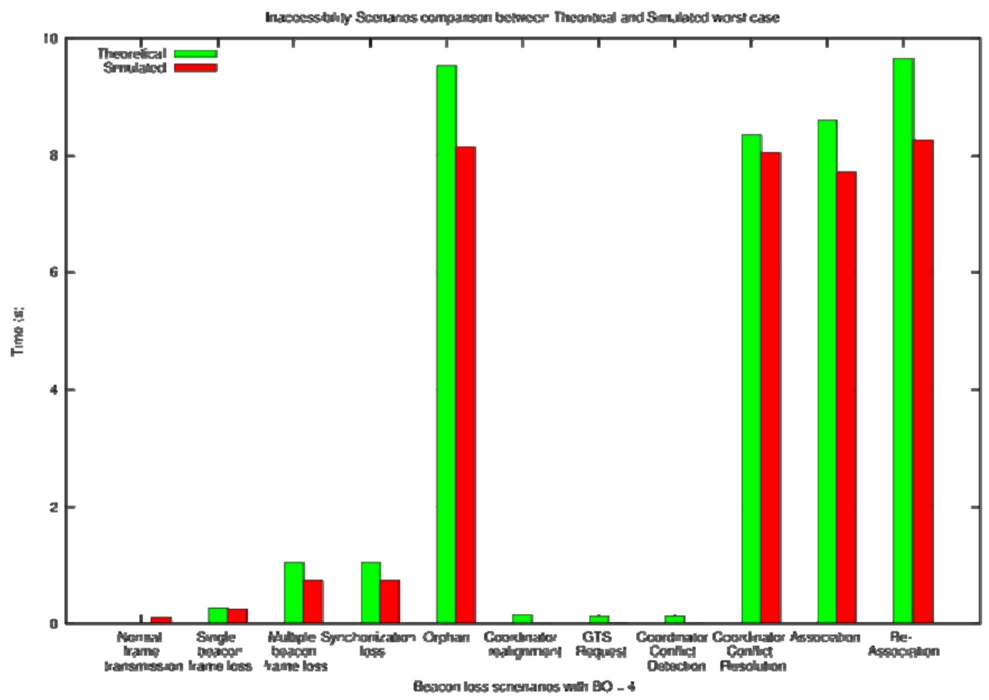


Figure 15 - Theoretical vs. Simulated – Worst-case comparison (BO=SO=4, T_{BI}=0.240*s*).

The engineering details of these extensions to the NS-2 IEEE 802.15.4 module can be found in the Deliverable D3.3. The source code of the IEEE 802.15.4 module with the GTS support can be downloaded at URL: http://www.karyon-project.eu/wp-content/uploads/2013/11/ns-2_v2_35_802_15_4_gts_extension.zip. The source code of the fault injector and the temporal analysis component can be downloaded at URL:http://www.karyon-project.eu/wp-content/uploads/2013/11/ns-2_v2_35_fault_injector_module.zip. Instructions about how to install the additional tools in the NS-2 simulator (version 2.35) are found in the available packages.

2.3.3.4 Traffic Analysis Tool

In order to analyze the traffic that runs through a network and study it, it became fundamental to build a traffic analysis tool that has the ability to report and log in a user-friendly way several aspects of network operation. To achieve this purpose we have extended the Wireshark Network Protocol Analyzer as follows (Figure 16): a traffic monitoring probe for IEEE 802.15.4 wireless networks is included for Wireshark use; an adapter application, translating the format of the frames received by the IEEE 802.15.4 frame monitor into the PCAP format understandable by the Wireshark tool; a fault injection unit aiming the corruption in the air of IEEE 802.15.4 frames.

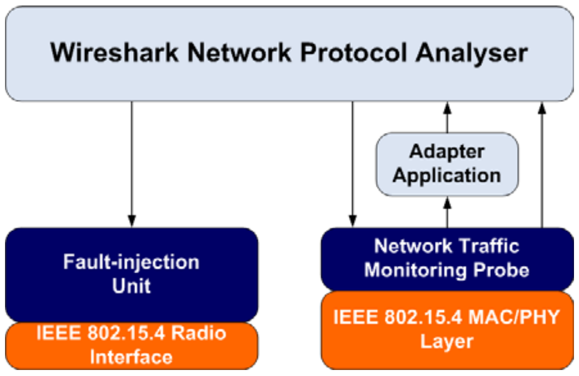

Wireshark Network Protocol Analyser
Adapter Application
Fault-injection Unit
IEEE 802.15.4 Radio Interface
Network Traffic Monitoring Probe
IEEE 802.15.4 MAC/PHY Layer

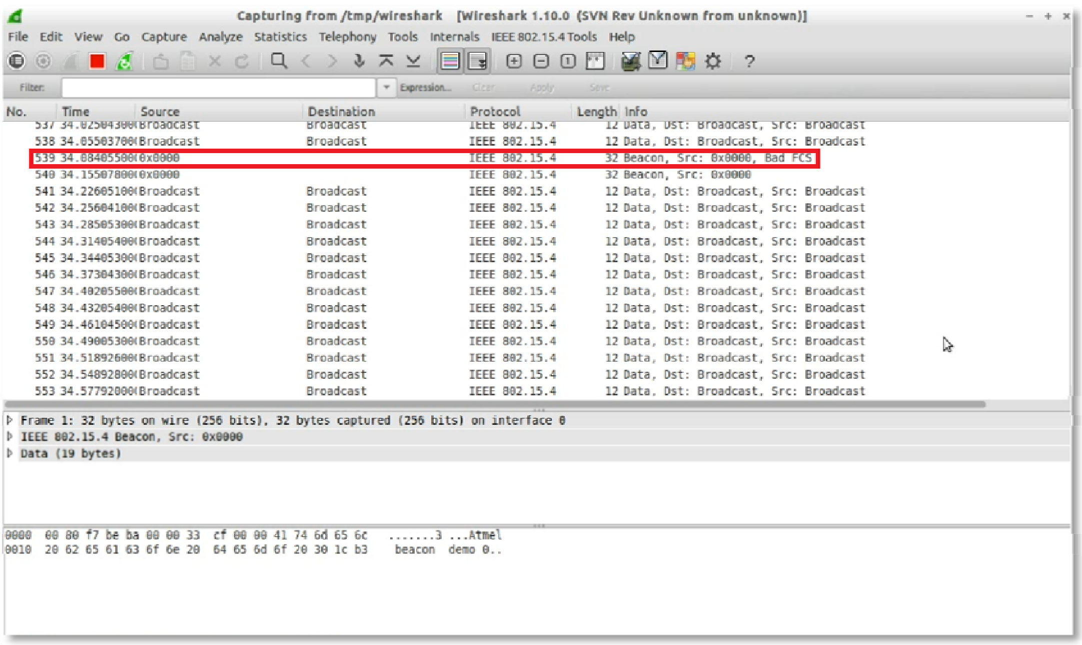
Figure 16 – Integration of developed modules with WireShark

Figure 17 - Atmel REB232ED-EK Evaluation Kit

Both the traffic monitoring tool and the fault injection unit are build using the Atmel REB232ED-EK Evaluation Kit boards, presented in Figure 17. The board used for IEEE 802.15.4 network traffic monitoring is configured as a promiscuous listener, with some low-level protocol extensions of frame check sequence (FCS) mechanisms for error notification; the Wireshark tool sees this probe as a device driver. The adapter application reads the probe and performs a format translation of the received frames to the PCAP format. On the other hand, a special-purpose configuration is set at the fault injection unit, directly from the Wireshark interface: both MAC and CSMA/CD protocols can be disabled, in order to allow board utilization as fault injector.

The fault injection unit can be configured for the injection in the network medium of several entities: noise; PHY layer symbol stream; MAC control/data formatted frame. Several options can be selected for fault injection timing characteristics: duration; burst size; minimum separation time; randomness.

The effectiveness of this approach is illustrated in Figure 18, which shows the monitoring of a frame received with an incorrect FCS, resulting from a previous fault injected by the Wireshark-based tool.


Capturing from /tmp/wireshark [Wireshark 1.10.0 (SVN Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals IEEE 802.15.4 Tools Help
Filter:
Expression... Clear Apply Save

```
No.  Time         Source      Destination   Protocol       Length Info
537 34.02504300(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
538 34.05503700(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
539 34.08405500(0x0000                      IEEE 802.15.4    32 Beacon, Src: 0x0000, Bad FCS
540 34.15507800(0x0000                      IEEE 802.15.4    32 Beacon, Src: 0x0000
541 34.22605100(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
542 34.25604100(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
543 34.28505300(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
544 34.31405400(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
545 34.34405300(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
546 34.37304300(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
547 34.40205500(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
548 34.43205400(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
549 34.46104500(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
550 34.49005300(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
551 34.51892600(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
552 34.54892800(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast
553 34.57792000(Broadcast     Broadcast     IEEE 802.15.4    12 Data, Dst: Broadcast, Src: Broadcast

▷ Frame 1: 32 bytes on wire (256 bits), 32 bytes captured (256 bits) on interface 0
▷ IEEE 802.15.4 Beacon, Src: 0x0000
▷ Data (19 bytes)

0000  00 80 f7 be ba 00 00 33  cf 00 00 41 74 6d 65 6c   .......3 ...Atmel
0010  20 62 65 61 63 6f 6e 20  64 65 6d 6f 20 30 1c b3    beacon  demo 0..
```

Figure 18 - Wireshark monitoring IEEE 802.15.4 traffic

The source code of the modules integrated into the WireShark tool can be downloaded from a protected repository (access to these tools has to be explicitly requested) at: https://bitbucket.org/rpcaldeira/karyon-wireshark/get/master.zip. The communication module responsible to connect WireShark with the ATMEL hardware is available in a separated repository, and can be downloaded at: https://bitbucket.org/rpcaldeira/karyon-adapter/get/master.zip. Instructions about how to compile those tools are found in the available packages.

2.3.3.5 Fault Injector for the automotive demonstrator

This section describes fault injection support for the automotive demonstrator in KARYON. We first describe failure modes that are applicable for communication, and then describe how the different failure modes can be emulated in the automotive demonstrator, which we refer to as Gulliver.

Failure modes: Part 5 of the ISO26262 international standard for functional safety [27] in road vehicles lists different failure modes for communication that needs to be analyzed. The failure modes are listed in Table D.1 in Part 5 of the standard, and are divided into on-chip communication and data transmission. The failure modes for the two types of communication are described next.

On-chip communication: Failures in the on-chip communication can be detected and handled using techniques such as parity bits, Hamming coding and by the use redundant channels (see, e.g., Table D.14 in [27]). Based on which diagnostic coverage (DC) that is claimed for a safety mechanism, Table D.1 in [27] lists which failure modes that must be analyzed. Next, we describe the failure modes, listed in italics, which are listed for on-chip communication.

Stuck-at failures are described as a continuous low or high signal at the pins of an element. They are applicable for elements which have a pin level interface for data, control, address or arbitration signals.

The *direct current (d.c.)* fault model extend stuck-at failures with stuck-open, open or high impedance outputs, and short circuits between signal lines. The analysis of the d.c. fault model is applicable for data, control, address and arbitration signals, but is mainly intended for main signals or on highly coupled interconnections.

Bus arbitration is used to determine which device that controls the bus, when several devices are connected to the bus. In [27] , *no arbitration* and *continuous arbitration* are mentioned as failure modes for the on-chip communication. *Time out* is mentioned in both IEC 61508 and ISO 26262 but none of the standards describe the failure mode. We interpret the time out failure mode as a delay in a signal.

Soft errors, which are also referred to as Single Event Upsets (SEU), are caused by ionizing particles such as alpha particles, heavy-ions, and high energy neutrons. When such particles hit an integrated circuit, they may alter the value that is stored in a memory element and thereby causing a bit-flip. According to [ISO], soft errors should be analyzed for sequential parts.

Failure mode	**Interpretation**
Message corruption	The received data of a message is corrupted.
Message delay	A message is received later than expected by all receivers.
Message loss	A message is lost by all receivers.
Unintended message repetition	Receivers obtain two messages with the same information instead of one message.
Resequencing	Messages are received with incorrect sequence numbering.
Insertion of message	Receivers obtain a message that they did not expect
Masquerading (or incorrect addressing)	A sender sends messages using an id of a different sender
Asymmetric information	This can be that information from a single sender is received differently by receivers. It can also be that information from a sender is only received by a subset of the receivers.
Blocking access to a communication channel	This is similar to a babbling idiot, and prevents nodes from accessing the communication channel

Table 9 - Failure modes for data transmission

<u>Data transmission:</u> Failures in the data transmission can be handled using redundancy in different forms, e.g., by using additional information or hardware. The types of failures that need to be analyzed according to Table D.1 in Part 5 and Annex D in Part 6 of the ISO26262 standard are summarized in Table 9.

<u>Emulating Failure modes for communication:</u> We emulate the previously mentioned failure modes for on-chip communication using the techniques illustrated in Figure 19 and Figure 20. The signal between two elements will pass via a fault injection module, which has the capability to modify the transmitted signal value. In most cases, such as for soft errors, a faulty signal only

relies on the non-faulty signal as shown in Figure 20. For short-circuit between signals, however, it might be easier to emulate the effects using both of the non-faulty signals as shown in Figure 19.

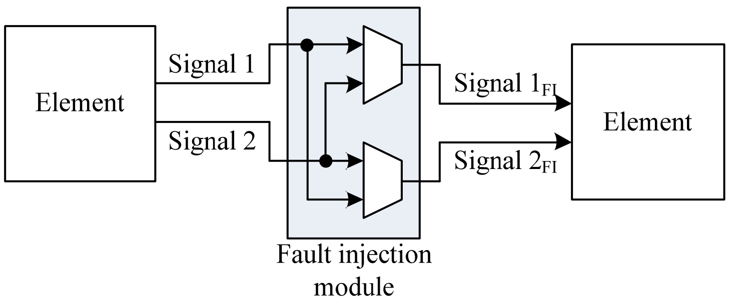


Figure 19 - Injection of short-circuit failures between two signals.

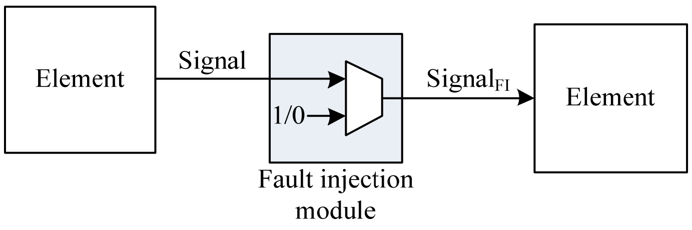


Figure 20 - Injection of stuck-at faults in a signal.

The failure modes for data communication will be emulated using jamming, packet injection and packet sniffing. Jamming (see, e.g., [13] and [47]) is used to prevent one or several nodes from receiving or sending packets. Packet injection is used to insert additional, duplicated and corrupted messages in the wireless network. Packet sniffing allows the fault injection module to listen on the wireless traffic in a non-intrusive manner, and this is used for logging but also for triggering the injection of different failure modes. Table 10 shows how the different failure modes are realized by jamming and packet injection. For example, to emulate the effects of a message delay, we will use jamming to prevent nodes from receiving the original message. We will then resend the original message with a delay. (This assumes that we have knowledge of the content of the message a priori.) Message losses are emulated by activating the jamming, e.g., when a specific message is transmitted.

Failure mode	**Jamming**	**Packet injection**
Message corruption	x	x
Message delay	x	x
Message loss	x	
Unintended message repetition		x
Resequencing		x

Insertion of message		x
Masquerading (or incorrect addressing)		x
Asymmetric information	x	x
Blocking access to a communication channel	x	

Table 10 - Emulation of different failure modes.

<u>Fault injection triggering:</u> Fault injection is activated using triggers which can be based on elapsed time, probability per received packet, sender or receiver of a packet, or data in a packet. As seen in Figure 21, different triggers can be combined to create a chain of events for starting or stopping the injection of a specific fault. Using this method, well-known packet loss models such as Bernoulli and Gilbert-Elliot can be supported, as well as simple triggers based on, e.g., the elapsed time.

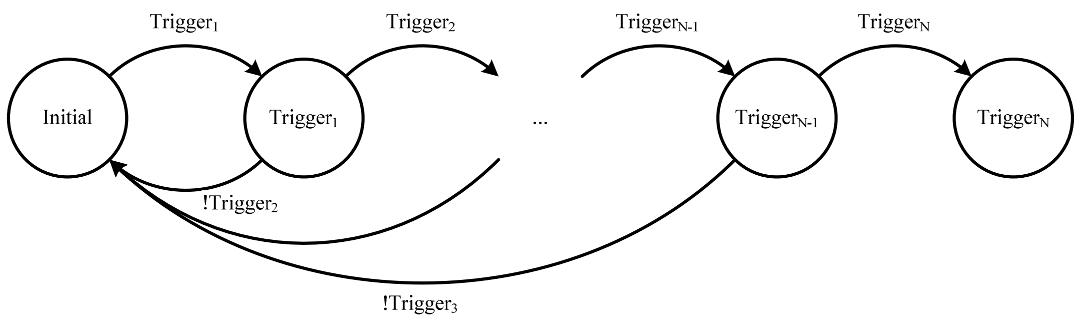


Figure 21 - Fault injection triggering.

Figure 22 shows the state machine in that we use to control the fault injection in the Gulliver vehicles. The idle state has an internal variable to keep track which trigger that is currently evaluated. When all triggers have been evaluated to true in the correct order, fault injection is activated in the "Start FI" state. We immediately after enter the FI state, which we stay in until all of the stop triggers have been fulfilled. We then either stop the fault injection completely, or return to the idle state to wait until the start triggers are fulfilled again.

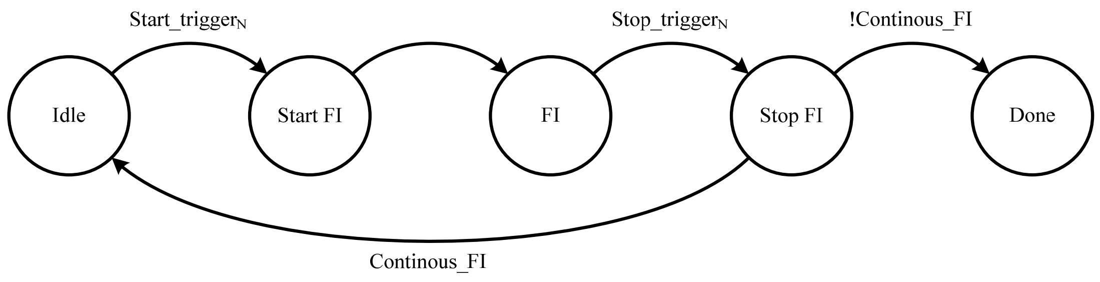


Figure 22. Fault injection state machine.

To inject failures in Gulliver, we will use the following triggers:

- Time – Trigger is enabled after a specified time has elapsed.

- Packet probability – Trigger is enabled with a specified probability for each received packet.
- Packet destination address – Trigger is enabled when a packet with a matching source address is received.
- Packet source address – Trigger is enabled when a packet with a matching destination address is received.
- Packet data – Trigger is enabled when the specified data matches the received data.

Triggers can be used in any combination, making it possible to define advanced sequences for enabling the fault injection.

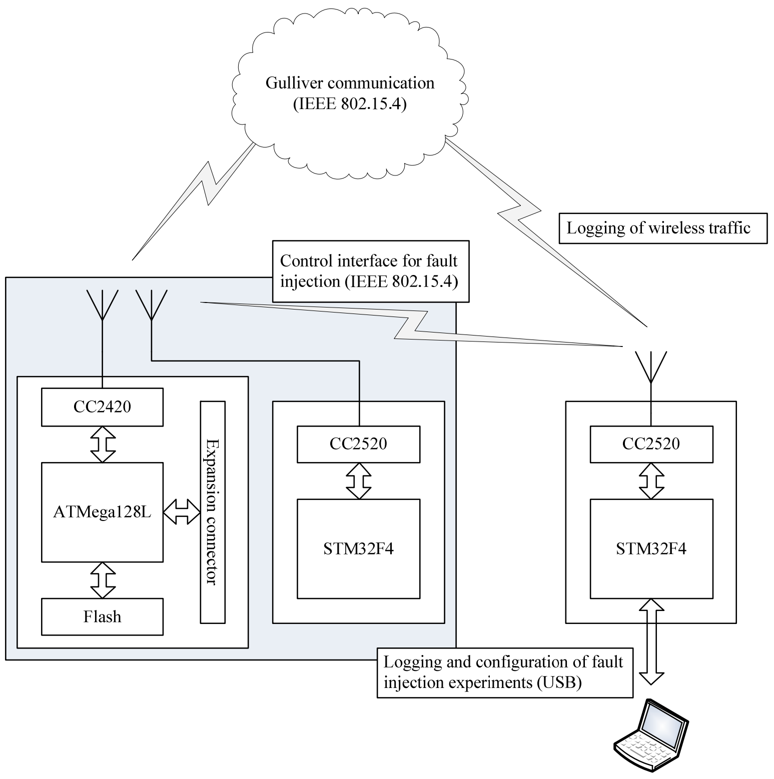

Gulliver communication
(IEEE 802.15.4)
Logging of wireless traffic
Control interface for fault injection (IEEE 802.15.4)
CC2420
ATMega128L
Expansion connector
Flash
CC2520
STM32F4
CC2520
STM32F4
Logging and configuration of fault injection experiments (USB)

Figure 23 - Fault injection for Gulliver

<u>Fault injection prototype:</u> Figure 23 shows a fault injection prototype for the Gulliver demonstrator. Each Gulliver vehicle is extended with an additional computer node for fault injection, which can be controlled via IEEE 802.15.4. The fault injection node is based on the STM32F4 microcontroller from ST and the CC2520 communication chip from Texas Instrument, see Figure 24. The node is built based on layout and hardware schematics which are freely available from [14] . The fault injection nodes on the Gulliver vehicles are controlled using a PC and a gateway node connected to the PC. The gateway node is also responsible for logging all of Gulliver's IEEE 802.15.4 traffic to be able to analyze how an injected error affects the system. The gateway node uses the same hardware as the fault injection nodes but is running different software.

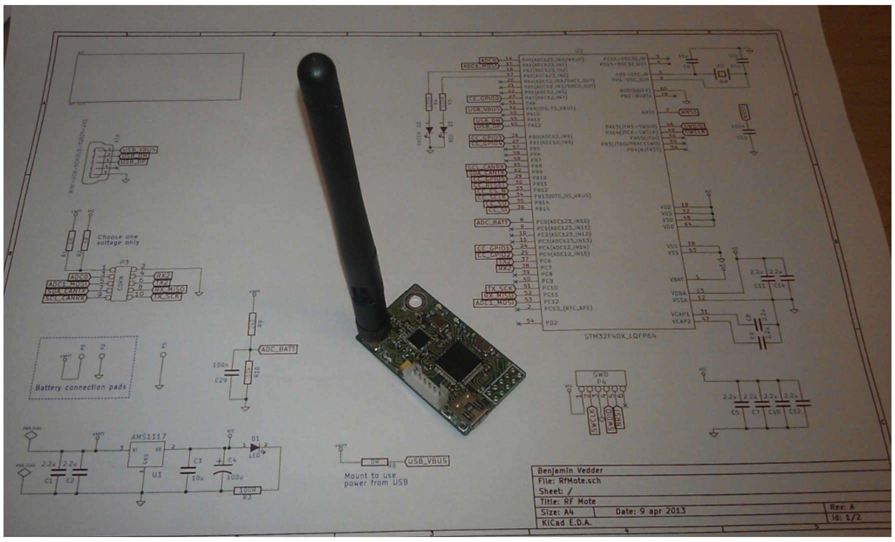


Figure 24 - Fault injection node based on STM32F4 and CC2520.

This page is intentionally left blank.

2.4 Predictability and Adaptation in Middleware for Advanced Safety-Critical Control

2.4.1 Dependable Adaptation of Real-Time Applications

Despite the various proposed techniques to improve the resilience of wireless networks, there may still be some cases where hard real-time control performed using wireless networking may not be the best solution. This can happen in cases where the cost (in terms of performance, or the monetary cost of equipment) of achieving the desired reliability necessary for hard real-time control makes this option less appealing than the alternatives, based on less strict real-time guarantees.

For instance, this may happen in applications such as road vehicles operating in environments where we cannot control the number and intensity of the sources of radio frequency interference, or the various radio-opaque obstacles encountered during the vehicles' movement. Nevertheless, even in those situations, wireless networking can be used as part of a broader safety-critical control solution, as long as the real-time guarantees of the control cycle are not compromised by timing faults relating to the wireless networking.

Such solutions require partitioning the system into separate domains, with different timeliness properties. One hard real-time domain must be responsible for timely processing sensory information and reacting to the environmental conditions, in such a way that safety is maintained. Another, separate domain, with less stringent real-time properties, can contribute to the overall process of acquiring sensory data, coordinating with other systems and contributing to actuation decisions, as long as this process does not jeopardize the safety of the former domain.

The hard real-time domain, being limited by a hard real-time model, will not be very flexible in its scheduling and other time-related properties, due to that model being based on an a priori determination of the worst-case timings, as well as other relevant properties impacting the control loop, such as the accuracy and validity of sensory data, or the characteristics of the actuators used.

The non-hard real-time domain, on the other hand, does not have to be limited by these constraints. For instance, it may have deadlines that should be met when possible, but that do not have to be absolutely guaranteed. An example of this pattern would be if we structured the system in such a way that the non-hard real-time domain should produce information to be used by the hard real-time domain; if this information is not available to the hard real-time domain within a given deadline then the hard real-time domain can detect this omission and still react appropriately, and on time.

Being free of the absolute guarantees of the hard real-time model, the non-hard real-time domain has the opportunity to use more efficient timeliness models, since it does not have to be designed based on worst-case conditions, as a hard real-time model would require. This can allow the overall safety-critical control loop to be more efficient. We still have the safety guarantees provided by the hard real-time subsystems, but we can also reap the benefits of more dynamic and flexible non-hard real-time subsystems.

Although the non-hard real-time domain does not have to absolutely guarantee that deadlines are respected, for this scheme to be truly effective it should still fulfill its functional requirements within the deadlines as much as possible, at least to a degree that is still in balance with its overall goal of being flexible and dynamic, and thus allows it to adapt to different scenarios and best take advantage of them.

We evaluated the feasibleness of using a probabilistic a probabilistic approach to attain such a trade-off between guaranteeing that deadlines are met by a non-hard real-time domain of the system, while still allowing such a domain to take best advantage of the actual runtime conditions experienced under a variety of scenarios.

2.4.1.1 Probabilistic Approach

We model the operating environment (the wireless network and the physical environment that can affect it) as a stochastic process, such that the QoS of the network is a random variable of that stochastic process. The driving assumption regarding the environment (and the associated stochastic process) is that it has limited dynamics; that is, it cannot repeatedly change substantially in a short amount of time. Either it changes slowly, or if it changes quickly it can only do so every so often, with a low probability. This assumption enables us to dependably monitor and adapt to the QoS offered by the operating environment.

The challenge behind this approach was to characterize whether the actual operating environments we are interested in (the combination of real physical environments, including the interferences they experience, with specific network technologies) actually follow this assumption, and thus are amenable to the probabilistic approach proposed.

To validate this assumption, we evaluated how well our previously proposed probabilistic technique was able to characterize the QoS of 802.15.4 networks (CAP traffic), including deriving bounds related to this QoS, to be used as deadlines for a probabilistic real-time approach. In particular, we focused on analysing the latency of the network, the bounds for which can be used as a deadline for the delivery of messages.

2.4.1.2 Evaluation Scenarios

We chose two types of evaluation scenarios: a simple linear network, to isolate the effects we were trying to study from any unwanted additional interactions, and a larger and more complex network, to evaluate more complex and realistic scenarios.

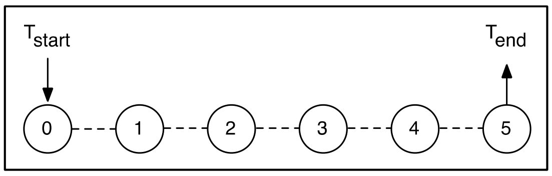


Figure 25 - Simulation Scenario 1 – Simple Linear Network, without Interferences

The simulation scenario 1 implemented a straightforward linear network, where messages are sent from a network node in one end and received by a node at the other end. The latency of this trip is measured.

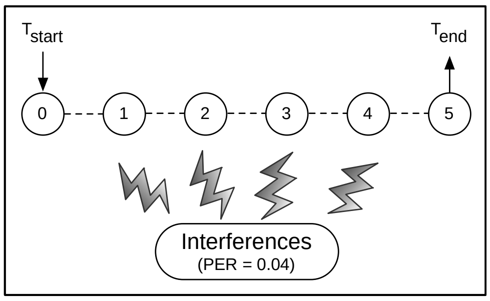


Figure 26 - Simulation Scenario 2 – Simple Linear Network, with Interferences (PER 4%)

The simulation scenario 2 is equal to scenario 1, except that a source of interferences was introduced, which produced an effective Packet Error Rate (PER) of approximately 4%.

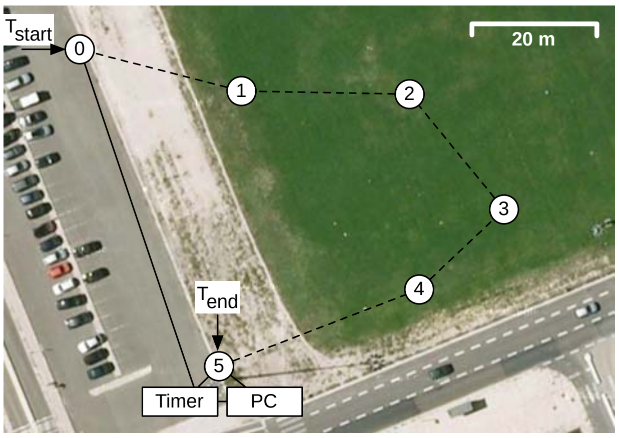


Figure 27 - Scenario 1 – Real Linear Network

A real network, built from 6 RCB128RFA1 V6.3.1 evaluation modules from dresdenelektronik, was also built, in such a way that replicated as closely as possible the linear simulation scenarios, given the constraints of the deployment target. These nodes used an 802.15.4 PHY and MAC, with the routing performed statically. For accuracy, the latency was measured with the help of additional hardware, through wires connected to both the source and destination nodes, and recorded to a PC.

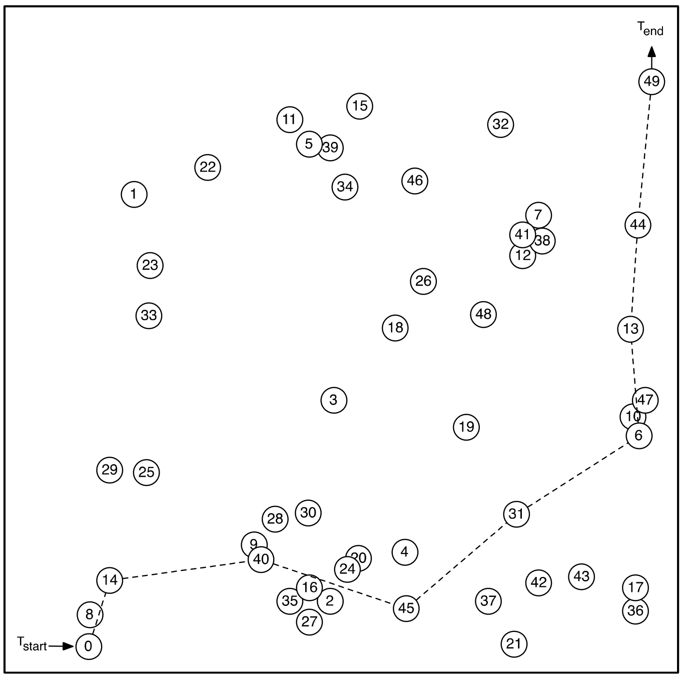


Figure 28 - Simulation Scenario 3

The simulation scenario 3 studied the more complex scenario previously mentioned. For all of these scenarios multiple variants were tested, to study the effects of various parameters. The following table summarizes the main tested scenarios and variants.

Scenario	# Nodes	Update Interval	Sample Size (n)
Real Network	6	Constant (0.05 s)	20, 30, 60, 100
1-A	6	Constant (0.05 s)	20, 30, 60, 100
1-B	6	Exponential ($\lambda^{-1} = 0.05$)	30
1-C	5-30	Constant (0.05 s)	30
2	6	Constant (0.05 s)	30, 100
3-A	50	Constant (0.05 s)	30
3-B	50	Exponential ($\lambda^{-1} = 0.05$)	30, 100
3-C (Movement)	50	Exponential ($\lambda^{-1} = 0.05$)	30
3-D (Movement, Competing message flows)	50	Exponential ($\lambda^{-1} = 0.05$)	30, 100

Table 11 - List of Main Evaluation Scenarios and Variants

2.4.1.3 Evaluation Results

In terms of the measured latencies, we observed two important results. One is that, overall, the latencies of the simulation scenarios and the real network mostly agree, as can be seen in Figure 29 - Latencies: Real Network vs. Simulation Scenario 1–A. The only noticeable difference is that in the real network a few messages had increased latency (+50 ms, or +100 ms) due to one (+50 ms) or two (+100 ms) transmission retries. This similarity between simulation and empirical experiment validates the other simulation scenarios, corroborating the more complex scenarios, which were not physically implemented.

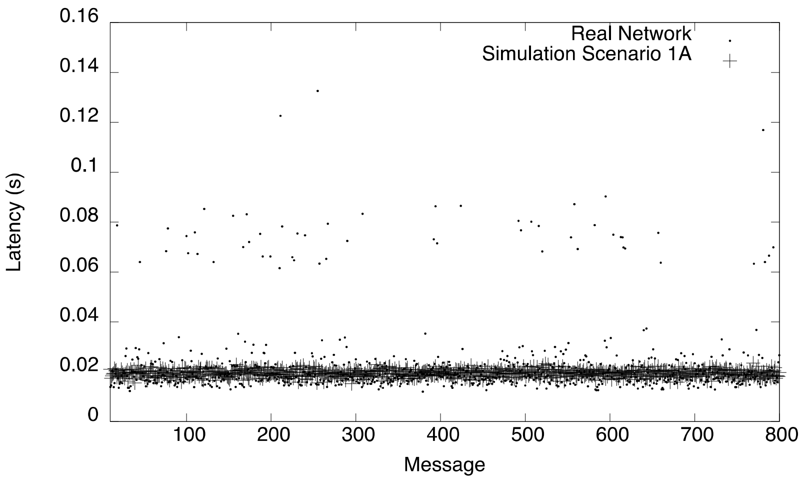


Figure 29 - Latencies: Real Network vs. Simulation Scenario 1–A

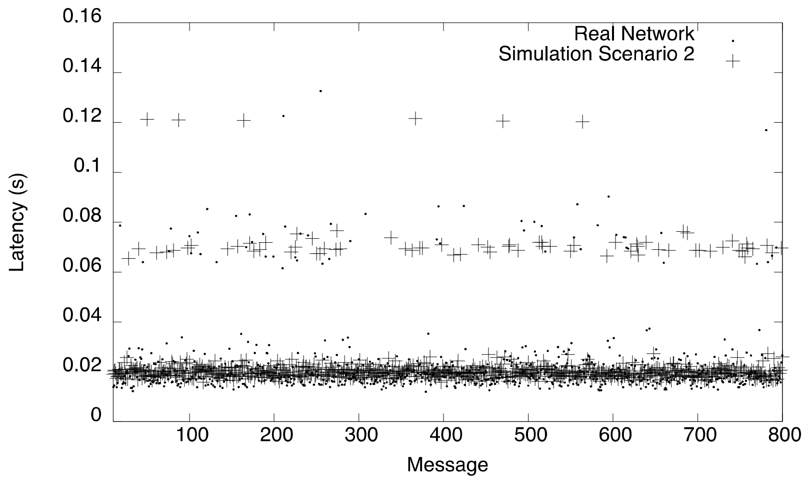


Figure 30 – Latencies: Real Network vs. Simulation Scenario 2

The second result is that when we introduced a source of interferences in the simulation we were able to almost perfectly replicate the observed empirical results of the simple linear network scenario, as observed in Figure 30 – Latencies: Real Network vs. Simulation Scenario 2. Since the real network was designed to minimize the sources of interference (e.g., the tests were run during the night, in a place with little RF activity in the used 2.4 GHz bands), this substantiates our claim that interferences can have a non-trivial impact, even if their presence and amount can be uncertain.

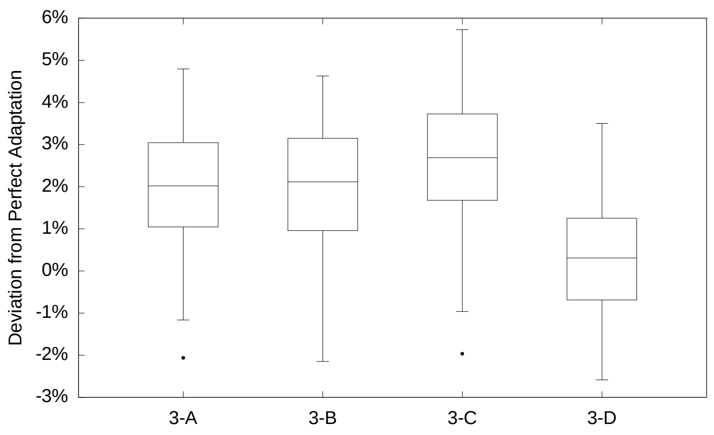


Figure 31 - Differences in Deadline Missing

Figure 31 - Differences in Deadline Missing compares, for the more complex simulation scenario 3 and its variants, the observed differences regarding the ratio of deadlines that were hit, for various probability/bound trade-offs, between the theoretical model and the simulation results. A difference of 0% means that deadlines would be missed with the exact probability predicted by the theoretical model, while a larger difference (either positive or negative) means that the optimal bounds were not estimated/predicted, resulting in a diverging probability of missing deadlines.

We observe that the difference between the theoretical model and the simulation results was always less than 6%, with most differences (the body of the box plot) being around 3% or less. Indeed, the other results obtained, reported in the annex, reveal even better results, with lower deviations between the theoretical model and the observations. In particular, the results showed improved results when the sample size was increased, and even when additional sources of complexity or interference were introduces (e.g., larger network sizes, increased PER, etc.).

This indicates that our assumption, that the operating environment is stable, does hold in the evaluated scenarios. Otherwise we would be guaranteed to observe greater deviations, since that is the only factor that can contribute to these results.

The implication is, therefore, that the evaluated scenarios (802.15.4 networks, with various characteristics) are indeed suitable targets where to apply a probabilistic approach to provide real-time guarantees, even if those guarantees are not as strict as those of a hard real-time model. This probabilistic model should therefore allow systems to adapt, at runtime, to the actual experienced conditions, and therefore improve the performance of the systems, compared with those employing only hard real-time models, while still providing more than best-effort real-time guarantees. A detailed discussion of the issues addressed in this section is performed in Annex A.2.1.

In annex: reprint of the paper "Fighting ***Uncertainty in Highly Dynamic Wireless Sensor Networks with Probabilistic Models***", L. Marques and A. Casimiro, in 32nd International Symposium on Reliable Distributed Systems (SRDS 2013), Braga, Portugal, Sept. 2013.

3. Vehicular Coordination Algorithms

The vehicle coordination problem requires making sure these vehicles will not have accidents, for example, when crossing intersections, or run into objects, such as road fences. Current driver-less vehicle implementations that operate on the road, e.g., Google car, do not consider cooperative path plans nor the coordination among self-driving vehicles. As a result, the risk exclusion of causing severe damage with sufficient certainty requires longer headways, which are the inter-vehicles distances (expressed in the manner of time). Recent developments in the automotive domain, as well as in communication networks, follow a more structured approach regarding information dissemination about road hazards, vehicle whereabouts and cooperative driving. We follow this structured- and cooperative-driving approach and consider a system in which vehicle crash is avoided via periodic agreement on cooperative service level and recalculation of the vehicle path plans accordantly. Namely, the safety kernel sets the system performance level according to the environment and data validity conditions. We demonstrate how, new and existing, vehicular coordination algorithms can perform the transition between service levels. By that we explore the safety kernel unique ability to facilitate both fault tolerance, and recovery. The focus here is the challenge in the integration of the safety kernel as a novel software architectural concept, but we also look into related issues, such as the safe transition after the system performance-level adjustment.

3.1 The Vehicular System

We present a system in which the vehicles use on board sensors, external references and communication networks for retrieving the information and constructing a local dynamic map that depicts the localization, velocity of nearby vehicles. Once all vehicles receive the information, each determines safe paths for all the level of services that the vehicles support before taking their joint actions. These paths ensure a safe driving for any coherent, yet possibly changing, service level. For example, in the case of cooperative functionalities such as adaptive cruise control and vehicle platooning, the plan considers the safe distance headways for each of the possible cooperative service level. Note that the cooperative service level may change during the plan term and, in turn, cause the vehicles to accordingly adjust their safe distance headways and inter-vehicle separation.

Cooperative driving may not only be dependent on the expected vehicle paths and the local sensory information but also the data validity. Thus, these functionalities are based on periodic information exchange with nearby vehicles, and validity assessment before deciding on the cooperative service level. One may consider systems in which vehicles have individual service levels that are based on their currently received information validity. The safety analysis in this case, must keep track of all the possible service level combinations of nearby vehicles and timely decide on the joint and safe driving behaviour. We use a cooperative service level evaluator in a way that guarantees safe transition between two different cooperative service levels, i.e., autonomous and cooperative. The system lets all nearby vehicles coherently refer to the cooperative service level. Changes to the data validity level are not the only reason why the cooperative service level may change. The system ability to agree on the cooperative service level is based on inter vehicle communications and thus it is prone to faults. During the recovery period, the cooperative service level evaluator may decide on a cooperative service level that is lower than any local service level.

3.2 Management of Cooperative and Autonomous Driving

The key objective of the safety kernel architecture is to provide system solutions for predictable and safe coordination of smart vehicles that autonomously cooperate and interact. This model for cooperative driving provides instructions that the vehicle actuator can use whenever the safety kernel decides on a cooperative service level. We show how the proposed system and safety kernel architecture address each test case, see the Annex for details. The system considers the models for providing local dynamic map and vehicle-to-vehicle communication, as well as the future horizon of events.

In cooperative driving, vehicle must be aware of the localization, velocity, acceleration, intention, etc., of each other within a given horizon. For the vehicular coordination we propose a variant of existing models and agreement on the cooperative service level. In each round, the vehicles execute four phases: (1) observe the environment for a fixed period of time, (2) compute a local plan according to a deterministic algorithm, (3) agree on the cooperative service level, and (4) move accordingly. We structure the algorithm using two parts: one for the autonomous driving and one for the cooperative driving. We let the safety kernel determine the system perform-level, and accordingly select which output should be direct to the pilot actuator.

3.3 Test Cases

WP5 plan is to demonstrate that the safety kernel has the ability to facilitate the implementation of vehicular coordination algorithms that need to deal with both degradation and recovery of the system data validity.

3.3.1 Adaptive Cruise Control and Vehicle Platooning

This application adjusts the headway between vehicles according to the data validity and the performance level. Namely, the safety kernel adjusts the inter-vehicle distance according to the performance level. We base our implementation on existing algorithms, see the Annex for details. The key information in this vehicular application is the relative distance and velocity the vehicle ahead. We assume that the autonomous abilities of the system are able to use information that is coming from on-board sensors to calculate the inter-vehicle distance (without the use of remote sensing) at a sufficiently high quality. The vehicle platooning algorithms that we base ours on uses vehicle-to-vehicle communication for agreeing on the joint (cooperative) cruising speed.

3.3.2 Intersection Crossing

This application schedules a safe intersection crossing by adjusting the vehicle speed according to the data validity and the performance level that is determined by the safety kernel. The cooperative intersection crossing algorithm is based on existing algorithms, see the Annex for details and there reference therein for the formal safety proof. We use the first-come first served approach for deciding who crosses the intersection first. We break symmetry using the right hand rule, i.e., we assume road-based priorities. Such symmetry breaking techniques are essential especially at low service levels for which there are no communication-based coordination guarantees. We note that our approach is extendable to systems that further consider the details and optimality of trajectory planning.

3.3.3 Coordinated Lane-Change

This application schedules a safe lane-change manoeuvre by adjusting the inter-vehicle distances in the subject lane accordingly to the data validity and the performance level that is determined by the safety kernel. We consider the existing literature for the lane change manoeuvre in both non-cooperative and cooperative contexts and base our algorithm on them, see the Annex A.3.1 for details.

In annex: preprint of the technical report "***Vehicular Coordination with a Safety Kernel in the Gulliver Test-Bed***". O. M. Ponce; T. Petig, and E. M. Schiller, Technical Report. 2013.

4. Conclusion

The primary focus of this *Final Report in Network Characteristics and Coordination Techniques* is on maintaining a high functional level in spite of environment uncertainties and faults of the communication and the computational systems.

This document included a conceptual framework, the models, the methods and the tools required to deal with and evaluate all the uncertainty aspects of communications. This report provided an outline of the services and structure of the supporting technologies. In particular, the report encompassed:

- a detailed analysis of network inaccessibility models, aiming to allow the evaluation of all the periods of network inaccessibility using the IEEE 802.15.4 as a use-case.
- the protocols and the protocol layers required to control uncertainties in communications, thus providing reliable and timely communication services despite the presence of errors;
- propose protocols that deal with uncertainties of networks. Particular emphasis is on providing abstractions to handle faults and support environment perception;
- provide a relevant set of tools for the analysis and verification of previous models. This includes model verification by fault injection techniques.
- a specification of the structure and of the service interface of the different components intended to cope uncertainty;
- reporting advances with respect to the reliable cooperation between mobile nodes and how they can set up vehicle coordination using the safety kernel.

The present document mainly described the activities developed and the achievements reached in the execution of task T3.1, between M12 and M24, which is now completed. Task T3.1 was specifically related with fundamental aspects of predictability and resilience in embedded networks. Beyond the technical work of more formal and theoretical nature, a set of tools was developed within the context of the work developed in Task T3.1, namely:

- IEEE 802.15.4 Network Inaccessibility Evaluation Tool (OpenOffice)
- NS-2 Simulator module with GTS (Guaranteed Time Slots) support for frame transmissions on IEEE 802.15.4 wireless networks
- NS-2 Simulator extension module for fault-injection and timeliness evaluation of IEEE 802.15.4 wireless networks under error conditions
- Wireshark extension module for fault-injection, monitoring and evaluation of IEEE 802.15.4 wireless networks

These software packages already are (or will be very soon) publically available at the project website.

This page is intentionally left blank.

5. References

[1] Abramson, N.: Development of the ALOHANET. Info. Theory, IEEE Trans. on 31(2) (1985) 119–123.

[2] Aad, I.; Hofmann, P.; Loyola, L.; Riaz, F. and Widmer, J. ***"E-MAC: Self-Organizing 802.11-Compatible MAC With Elastic Real-Time Scheduling"*** *In IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, October 2007, Pisa, Italy.

[3] Alon, N., Mohar, B.: The chromatic number of graph powers. Combinatorics, Probability & Computing 11(1) (2002) 1–10.

[4] Arumugam, M., Kulkarni, S.: Self-stabilizing deterministic time division multiple access for sensor networks. AIAA Journal of Aerospace Computing, Info., and Comm. (JACIC) 3 (2006) 403–419.

[5] Avižienis, A.; Laprie, J. C.; Randell, B. and Landwehr, C. ***"Basic Concepts and Taxonomy of Dependable and Secure Computing"*** *In IEEE Transactions on Dependable and Secure Computing.* vol. 1 (1), March 2004, USA.

[6] Åkerberg, J.; Gidlund, M. and Björkman, M. ***"Future research challenges in wireless sensor and actuator networks targeting industrial automation"*** *In 9th IEEE International Conference on Industrial Informatics (INDIN)*, July 2011, Lisbon, Portugal.

[7] Babaoglu, O. and Drummond, R. ***"Streets of Byzantium: Network Architectures for Fast Reliable Broadcasts"***. *In IEEE Transactions on Software Engineering.* vol. se-11 (6), June 1985, USA.

[8] Bernhard Haeupler, David R. Karger: Faster information dissemination in dynamic networks via network coding. PODC 2011: 381-390.

[9] Bernhard Haeupler: Analyzing network coding gossip made easy. STOC 2011: 293-302.

[10] Bohm, A. and Jonsson, M. ***"Supporting Real-Time Data Traffic in Safety-Critical Vehicle-to-Infrastructure Communication"***. *In Proceedings of 33rd IEEE Conference on Local Computer Networks (LCN 2008)*. October 2008, Montreal, Canada.

[11] Bharghavan, V.; Demers, A.; Shenker, S. and Zhang, L. ***"MACAW: A Media Access Protocol for Wireless LANs"***. *In Proceedings of the Conference of Special Interest Group on Data Communications (SIGCOMM)*. October 1994, New York, USA.

[12] Busch, C., Magdon-Ismail, M., Sivrikaya, F., Yener, B.: Contention-free MAC protocols for asynchronous wireless sensor networks. Distributed Computing 21(1) (2008) 23–42.

[13] Carlo Alberto Boano, et al. "Controllable radio interference for experimental and testing purposes in wireless sensor networks." In Proceedings of the IEEE 34th Conference on Local Computer Networks, 2009, pp. 865-872.

[14] "CC2520 and STM32 RF boards", http://vedder.se/2013/04/cc2520-and-stm32-rf-boards/, Accessed 2013-11-18.

[15] Dolev, S.: Self-Stabilization. MIT Press (2000).

[16] Do-Sik, Y. and Stark, W. E. ***"Characterization of Multipath Fading Channels: Channels with Specular Components"*** *In IEEE Transactions on Wireless Communications.* vol. 4 (4), July 2005, USA.

[17] Eckhardt, D. and Steenkiste, P. ***“Measurement and Analysis of The Error Characteristics of An In-Building Wireless Network”***. *In Proceedings of the Conference of Special Interest Group on Data Communications (SIGCOMM)*. October 1996, Palo Alto, USA.

[18] E-López, E.; V-Alonso, J.; M-Sala, A.; G-Haro, J.; P-Mariño, P. and Delgado, M. ***“A Wireless Sensor Networks MAC Protocol For Real-Time Applications”*** *In Personal Ubiquitous Computing Journal*. vol. 12 (2) January 2008, UK.

[19] Fujiwara, T.; Kasami, T.; Kitai, A. and Lin, S. ***“On The Undetected Error Probability For Shortened Hamming Codes”***. *In IEEE Transactions on Communications*. vol. 33 (6), June 1985, USA.

[20] Guillermo Barredo Garcíasa “Design of a Session Managing Algorithm for Solving K-Token Dissemination Problem Using Network Coding,” Submitted for review as a Master of Science Thesis in the Computer Science and Engineering Chalmers University of Technology, Göteborg, Sweden, June 2013.

[21] Hameed, M.; Trsek, H.; Graeser, O. and Jasperneite, J. ***“Performance Investigation And Optimization of IEEE 802.15.4 For Industrial Wireless Sensor Networks”***. *In IEEE International Conference on Emerging Technologies and Factory Automation (ETFA).* September 2008, Hamburg, Germany.

[22] Herman, T., Tixeuil, S.: A distributed TDMA slot assignment algorithm for wireless sensor networks. In: ALGOSENSORS. Volume 3121 of Lecture Notes in Computer Science., Springer (2004) 45–58.

[23] Herman, T., Zhang, C.: Best paper: Stabilizing clock synchronization for wireless sensor networks. In Datta, A.K., Gradinariu, M., eds.: SSS. Volume 4280 of Lecture Notes in Computer Science., Springer (2006) 335–349.

[24] Hu, Y. –C.; Perrig, A. and Johnson, D. ***“Wormhole Attacks in Wireless Networks”***. *In IEEE Journal on Selected Areas in Communications*. vol. 24 (2), February 2006, USA.

[25] IEEE 802.11P. ***“Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications - amendment 6: Wireless access in vehicular environments - IEEE standard 802.11p”***. *IEEE 802.11 Working Group*. July 2010, USA.

[26] IEEE 802.15.4. ***“Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications For Low-Rate Wireless Personal Area Networks (WPANs) – IEEE Standard 802.15.4”***. *IEEE P802.15 Working Group, Revision of IEEE Standard 802.15.4-2006*. September 2011, USA.

[27] ISO26262-5, “Road vehicles – Functional safety – Part 5: Product development at the hardware level”, 2011.

[28] Jung, C.; Hwang, H.; Sung, D. and Hwang, G. ***“Enhanced Markov Chain Model and Throughput Analysis Of The Slotted CSMA/CA For IEEE 802.15.4 Under Unsaturated Traffic Conditions”***. *In IEEE Transactions on Vehicular Technology*. vol. 58 (1), January 2009, USA.

[29] Khattab, S.; Mosse, D. and Melhem, R. ***“Modeling of The Channel-Hopping Anti-Jamming Defense In Multi-Radio Wireless Networks”***. *In 5th ACM International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services (MobiQuitous)*. July 2005, Dublin, Ireland.

[30] Koubâa, A.; Alves, M.; Tovar, E. and Cunha, A. ***“An implicit GTS Allocation Mechanism In IEEE 802.15.4 For Time-Sensitive Wireless Sensor Networks: Theory***

And Practice" *In Springer Real-Time Systems Journal*. vol. 39 (1-3), August 2008, USA.

[31] Kulkarni, S.S., Arumugam, M.: Transformations for write-all-with-collision model, . Computer Communications 29(2) (2006) 183–199.

[32] Mitton, N., Fleury, E., Lassous, I.G., Sericola, B., Tixeuil, S.: Fast conver- gence in self-stabilizing wireless networks. In: 12th Int. Conf. Parallel and Distributed Systems (ICPADS'06). (2006) 31–38.

[33] Molloy, M., Salavatipour, M.R.: A bound on the chromatic number of the square of a planar graph. J. Comb. Theory, Ser. B 94(2) (2005) 189–213.

[34] Mpitziopoulos, A.; Gavalas, D.; Konstantopoulos, C. and Pantziou, G. *"**A Survey on Jamming Attacks and Countermeasures in WSNs**". In IEEE Communications Surveys & Tutorials*. vol. 11 (4), December 2009, USA.

[35] Petrova, M., Riihijarvi, J., Mahonen, P. and Labella, S. ***"Performance study of IEEE 802.15.4 using measurements and simulations"***. *In Proceedings of the Wireless Communications and Networking Conference (WCNC)*. April 2006, Las Vegas, USA.

[36] Pickholtz, R.; Milstein, L. and Schilling, D. ***"Spread Spectrum for Mobile Communications"***. *In IEEE Transactions on Vehicular Technology*. vol. 40 (2), May 1991, USA.

[37] Ramachandran, I.; Das, A. K. and Roy, S. ***"Analysis Of The Contention Access Period Of IEEE 802.15.4 MAC"***. *In ACM Transactions on Sensor Networks*. vol. 3 (1), March 2007, USA.

[38] Rufino, J.; Almeida, C.; Veríssimo, P. and Arroz, G. ***"Enforcing Dependability and Timeliness in Controller Area Networks"***. *In 32nd Annual Conference of the IEEE Industrial Electronics Society (IECON)*. November 2006, Paris, France.

[39] Scopigno, R., Cozzetti, H.A.: Mobile slotted aloha for VANETs. In: 70th IEEE Vehicular Technology Conf. (VTC-Fall'09). (2009) 1 – 5.

[40] Souza, J. L. R. and Rufino, J. ***"Building Fundamental Properties For Real-Time Wireless Sensor Networks"***. *AIR-II Technical Report RT-10-01*. January 2010, Lisbon, Portugal.

[41] Souza, J. L. R. and Rufino, J. ***"Characterization of Inaccessibility In Wireless Networks - A Case Study on IEEE 802.15.4 Standard"***. *In Analysis, Architectures, and Modelling of Embedded Systems: Third IFIP TC 10 International Embedded Systems Symposium, IESS*. vol. 310, September 2009, Langernargen, Germany.

[42] Stone, T.; Alena, R.; Baldwin, J. and Wilson, P. ***"A Viable COTS Based Wireless Architecture for Spacecraft Avionics"***. *In IEEE Aerospace Conference*. March 2012, Big Sky, Montana.

[43] Veríssimo, P.; Rufino, J. and Rodrigues, L. ***"Enforcing Real-Time Behaviour on LAN-Based Protocols"***. *In 10th IFAC Workshop on Distributed Computer Control Systems*. September 1991, Semmering, Austria.

[44] Viqar, S., Welch, J.L.: Deterministic collision free communication despite continuous motion. In: 5th Inter. Workshop Algo. Wireless Sensor Net. (ALGOSENSORS). (2009) 218–229.

[45] Wei, S.; Tingting, Z.; Gidlund, M. and Dobslaw, F. ***"SAS-TDMA: A Source Aware Scheduling Algorithm For Real-Time Communication In Industrial Wireless Sensor Networks"*** *In Wireless Networks*. vol. 19 (6), August 2013, USA.

[46] WIRELESSHART. ***"Control With WirelessHART"***. 2009, USA. Available in: http://www.hcf-files.com/webasyst/published/DD/2.0/file_link.php?sl=a9d3da14df068c4e93ef2af9598a39ed&DB_KEY=V0VCRklMRVM%3D. Last accessed: 2013-12-02.

[47] Wood, Anthony D., John A. Stankovic, and Gang Zhou. "DEEJAM: Defeating energy-efficient jamming in IEEE 802.15. 4-based wireless networks," In Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'07), 2007.

[48] Xu, W.; Ma, K.; Trappe, W. and Zhang, Y. (2006a). ***"Jamming Sensor Networks: Attack And Defense Strategies"***. *In IEEE Network.* vol. 20 (3), June 2006, USA.

[49] Yu, F., Biswas, S.: Self-configuring TDMA protocols for enhancing vehicle safety with DSRC based vehicle-to-vehicle communications. Selected Areas in Communications, IEEE Journal on 25(8) (oct. 2007) 1526 –1537.

Annex A Papers and Reports

A.1 Enhancing Dependability and Timeliness in Wireless Communications

A.1.1 Self-stabilizing TDMA Algorithms for Wireless Ad-Hoc Networks without External Reference

"Self-stabilizing TDMA Algorithms for Wireless Ad-Hoc Networks without External Reference". T. Petig,; E. M. Schiller, and P. Tsigas. In Stabilization, Safety, and Security of Distributed Systems: Proceedings of 15th International Symposium, SSS 2013. Lecture Notes in Computer Science. November 2013, Osaka, Japan.

This page is intentionally left blank.

Self-stabilizing TDMA Algorithms for Wireless Ad-hoc Networks without External Reference [*]

Thomas Petig [†]
petig@chalmers.se

Elad M. Schiller[†]
elad@chalmers.se

Philippas Tsigas[†]
tsigas@chalmers.se

Abstract

Time division multiple access (TDMA) is a method for sharing communication media. In wireless communications, TDMA algorithms often divide the radio time into timeslots of uniform size, ξ, and then combine them into frames of uniform size, τ. We consider TDMA algorithms that allocate at least one timeslot in every frame to every node. Given a maximal node degree, δ, and no access to external reference, we consider the problem of the existence of collision-free self-stabilizing TDMA algorithms that have constant communication delay of τ.

We demonstrate that this problem has no solution when $\tau \leq \max((2-\varepsilon)\delta, \chi_2)$, where $\varepsilon > 0$, and χ_2 is the chromatic number for distance-2 vertex coloring. We observe the bound relevance to the bandwidth utilization when the network topology is a planar graph. As a complement to this lower bound, we focus on proving the existence of probabilistic collision-free self-stabilizing TDMA algorithms that have constant communication delay of τ. We consider basic settings (no hardware support for collision detection and no prior clock synchronization), and the collision of concurrent transmissions from transmitters that are at most two hops apart. In the context of self-stabilizing systems that have no external reference, we are the first to study this problem (to the best of our knowledge).

1 Introduction

Autonomous and cooperative systems will ultimately carry out risk-related tasks, such as piloting driverless cars, and liberate mankind from mundane labor, such as factory and production work. Note that the implementation of these cooperative systems implies the use of wireless ad hoc networks and their critical component – the *medium access control* (MAC) layer. Since cooperative systems operate in the presence of

[*]The work of this author was partially supported by the EC, through project FP7-STREP-288195, KARYON (Kernel-based ARchitecture for safetY-critical cONtrol).

[†]Computer Science and Engineering, Chalmers University of Technology, Sweden.

people, their safety requirements include the provision of real-time guarantees, such as constant communication delay. Infrastructure-based wireless networks successfully provide high bandwidth utilization and constant communication delay. They divide the radio into *timeslots* of uniform size, ξ, that are then combined into *frames* of uniform size, τ. Base-stations, access points or wireless network coordinators can schedule the frame in a way that enables each node to transmit during its own timeslot, and arbitrate between nearby nodes that wish to communicate concurrently. We strive to provide the needed MAC protocol properties, using limited radio and clock settings, i.e., no external reference for collision detection, time or position. For these settings, we demonstrate that there is no solution for the studied problem when $\tau < \max((2-\varepsilon)\delta, \chi_2)$, where $\varepsilon > 0$, δ is a bound on the node degree, and χ_2 is the chromatic number for distance-2 vertex coloring. Note that $\chi_2 = \delta + 1$ and $\chi_2 = 5\delta/3 + \mathcal{O}(1)$ for the cases of tree, and respectively, planar graphs [23]. The main result is the existence of probabilistic collision-free self-stabilizing TDMA algorithms that have constant communication delay of $\tau > \max(4\delta, X_2) + 1$, where $X_2 \geq \chi_2$ is a number that depends on the coloring algorithm in use. In the context of self-stabilizing systems that have no external reference, we are the first to study this problem (to the best of our knowledge).

Wireless ad hoc networks have a dynamic nature that is difficult to predict. This gives rise to many fault-tolerance issues and requires efficient solutions. These networks are also subject to transient faults due to temporal malfunctions in hardware, software and other short-lived violations of the assumed system settings, such as changes to the communication graph topology. We focus on fault-tolerant systems that recover after the occurrence of transient faults, which can cause an arbitrary corruption of the system state (so long as the program's code is still intact). These *self-stabilizing* [9] design criteria simplify the task of the application designer when dealing with low-level complications, and provide an essential level of abstraction. Consequently, the application design can easily focus on its task – and knowledge-driven aspects.

ALOHAnet protocols [1] are pioneering MAC algorithms that let each node select one timeslot per TDMA frame at random. In the Pure Aloha protocol, nodes may transmit at any point in time, whereas in the Slotted Aloha version, the transmissions start at the timeslot beginning. The latter protocol has a shorter *exposure period* during which packets may collide, because each transmission can collide only with transmissions that occur within its timeslot, rather than with two consecutive timeslots as in the Pure Aloha case. Note that the random access approach of ALOHAnet cannot provide constant communication delay. Distinguished nodes are often used when the application requires bounded communication delays, e.g., IEEE 802.15.4 and deterministic self-stabilizing TDMA [3, 16]. Without such external references, the TDMA algorithms have to align the timeslots while allocating them. Existing algorithms [5] circumvent this challenge by assuming that $\tau/(\Delta+1) \geq 2$, where Δ is an upper bound on the number of nodes with whom any node can communicate with using at most one intermediate node for relatying message. This guarantees zero exposure period with respect to at least one timeslot, s, and *all* transmissions from transmitters that are at most two hops away, where Δ is a bound on their number. However, the $\tau/(\Delta+1) \geq 2$ assumption implies bandwidth utilization that is up to $\mathcal{O}(\delta)$ times lower than the proposed algorithm, because $\Delta \in \mathcal{O}(\delta^2)$ (also for planar graphs [2]).

As a basic result, we show that $\tau/\delta \geq 2$, and as a complement to this lower bound,

we focus on considering the case of $\tau/\delta \geq 4$. We present a probabilistic collision-free self-stabilizing TDMA algorithm that has constant communication delay of τ. We show that it is sufficient to guarantee zero exposure period with respect to a single timeslot, s, and a *single* receiver, rather than *all* neighbors. This narrow opportunity window allows control packet exchange, and timeslot alignment. After convergence, there are no collisions of any kind, and each frame includes at most one control packet.
Related work The first proposal for a self-stabilizing TDMA algorithm for wireless ad hoc networks [11] considers external time reference for dividing the radio time. The proposal in [10] considers shared variable emulation. Several self-stabilizing algorithms adopt this abstraction, e.g., a generalized version of the dining philosophers problem for wireless networks in [7], topology discovery in anonymous networks [21], random distance-k vertex coloring [22], deterministic distance-2 vertex coloring [4], two-hop conflict resolution [25], a transformation from central demon models to distributed scheduler ones [27], to name a few. The aforementioned algorithms assume that if a node transmits infinitely many messages, all of its communication neighbors will receive infinitely many of them. We do not make such assumptions about *(underlying) transmission fairness*. We assume that packets, from transmitters that are at most two hops apart, can collide *every time*.

The authors of [17] present a MAC algorithm that uses convergence from a random starting state (inspired by self-stabilization). In [18, 19, 24], the authors use computer network simulators for evaluating self-$\star$ MAC algorithms. A self-stabilizing TDMA algorithm, that accesses external time references, is presented in [20]. Simulations are used for evaluating the heuristics of MS-ALOHA [26] for dealing with timeslot exhaustion by adjusting the nodes' individual transmission signal strength. We provide analytical proofs and consider basic radio settings. The results presented in [8, 14] do not consider the time it takes the algorithm to convergence, as we do. We mention a number of MAC algorithms that consider onboard hardware support, such as receiver-side collision detection [5, 6, 8, 26, 29]. We consider merely basic radio technology that is commonly used in wireless ad hoc networks. The MAC algorithms in [28, 29] assumes the accessibility of an external time or geographical references or the node trajectories, e.g., Global Navigation Satellite System (GNSS). We instead integrate the TDMA timeslot alignment with an existing self-stabilizing clock synchronization technique for wireless ad hoc networks [12]. Unlike existing self-stabilizing MAC algorithms, our correctness proof considers the TDMA algorithm together its mechanisms for timeslot alignment and clock synchronization.

We consider minimal transmission duration, uniform packet priority, and a predefined frame size, τ, as in [22]. This assumption holds whenever the node degree is bounded by a known constant, the node distribution is sparse, or when nodes can adjust their transmission power in overpopulated areas. The system designer may decide to abandon real-time guarantees, and prefer the participation of all nodes over a uniform communication delay. Such design alternatives are beyond the scope of this work.
Our contribution Given a maximal node degree, δ, we consider the problem of the existence of collision-free self-stabilizing TDMA algorithms that have constant communication delay of τ. In the context of self-stabilizing systems that have no external reference, we are the first to study this problem (to the best of our knowledge).

We make no assumptions about (underlying) transmission fairness, or external reference. Yet we assume that transmissions can collide *every time* the transmitters are at most two hops apart and their transmission times intersect (Section 2). For these settings, we establish a basic limit on the bandwidth utilization of TDMA algorithms in wireless ad hoc networks (Section 3). Namely, $\tau < \max(2\delta, \chi_2)$, where $\varepsilon > 0$, and χ_2 is the chromatic number for distance-2 vertex coloring. We note that $\chi_2 = \delta + 1$ holds for tree graphs, as well as $\chi_2 = 5\delta/3 + \mathcal{O}(1)$ for the case of planar graphs [23].

We prove the existence of collision-free self-stabilizing TDMA algorithms that have constant communication delay of τ without assuming the availability of external references (Section 4). The convergence period is within $\mathcal{O}(\mathsf{diam} \cdot \tau^2 + \tau^3\delta)$ steps starting from an arbitrary configuration, where diam is the network diameter. We note that in case the system happens to have access to external time references, i.e., start from a configuration in which clocks are synchronized, the convergence time is within $\mathcal{O}(\tau^3)$, and $\mathcal{O}(\tau^3\delta)$ steps when $\tau > 2\Delta$, and respectively, $\tau > 4\delta$.

2 System Settings

The system consists of a set, $P := \{p_i\}$, of communicating entities, which we call *nodes*. An upper bound, $\nu > |P|$, on the number of nodes in the system is known. Subscript font is used to point out that X_i is p_i's variable (or constant) X. Node p_i has a unique identifier, id_i, that is known to p_i but not necessarily to $p_j \in P \setminus \{p_i\}$.
Communication graphs At any instance of time, the ability of any pair of nodes to communicate, is defined by the set, $\delta_i \subseteq P$, of *(direct) neighbors* that node $p_i \in P$ can communicate with directly. The system can be represented by an undirected network of directly communicating nodes, $G := (P, E)$, named the *communication graph*, where $E := \{\{p_i, p_j\} \in P \times P : p_j \in \delta_i\}$. We assume that G is connected. For $p_i, p_j \in P$, we define the distance, $d(p_i, p_j)$, as the number of edges in an edge minimum path connecting p_i and p_j. We denote by $\Delta_i := \{p_j \in P : 0 < d(p_i, p_j) \leq 2\}$ the 2-neighborhood of p_i, and the upper bounds on the sizes of δ_i and Δ_i are denoted by $\delta \geq \max_{p_i \in P}(|\delta_i|)$, and respectively, $\Delta \geq \max_{p_i \in P}(|\Delta_i|)$. Note that $p_i \in \delta_i$ and $p_i \in \Delta_i$. We assume that $\mathsf{diam} \geq \max_{p_i, p_j \in P} d(p_i, p_j)$ is an upper bound on the network diameter.
Synchronization The nodes have fine-grained clock hardware (with arbitrary clock offset upon system start). For the sake of presentation simplicity, our work considers zero clock skews (although the proposed algorithm borrows a clock synchronization algorithm [12] that does consider skews). We assume that the *clock* value, $C \in [0, c-1]$, and any time stamp in the system have c states. The pseudo-code uses the *GetClock*() function that returns a timestamp of C's current value. Since the clock value can overflow at its maximum, and wrap to the zero value, arithmetic expressions that include timestamp values are module c, e.g., the function $AdvanceClock(x) := C \leftarrow (C + x) \bmod c$ adds x time units to clock value, C, modulo its number of states, c. We assume that the maximum clock value is sufficiently large, $c \gg \mathsf{diam}\tau^2$, to guarantee convergence of the clock synchronization algorithm, before the clock wrap around, as in [12] Section 3.3. We say that the clocks are *synchronized* when $\forall p_i, p_j \in P : C_i = C_j$.

Periodic pulses invoke the MAC protocol, and divide the radio time into *(broadcasting) timeslots* of ξ time units in a way that provides sufficient time for the transmission of a single packet. We group τ timeslots into *(broadcasting) frames*. The pseudo-code uses the event $timeslot(s)$ that is triggered by the pulse, t_{i_y}, where C_i is p_i clock value, $t_{i_y} := C_i \div \xi$ (integer division) and $s := t_{i_y} (\bmod \tau)$ is the *timeslot number*.

Operations The communication allows message exchange between the sender and the receiver. After the sender, p_i, fetches message $m \leftarrow MAC\_fetch_i()$ from the upper layer, and before the receiver, p_j, delivers it to the upper layer in $MAC\_deliver_j(m)$, they exchange m via the operations $transmit_i(m)$, and respectively, $m \leftarrow receive_j()$. We model the communication channel, $q_{i,j}$ (queue), from node p_i to node $p_j \in \delta_i$ as the most recent message that p_i has sent to p_j and that p_j is about to receive, i.e., $|q_{i,j}| \leq 1$. When p_i transmits message m, the operation $transmit_i(m)$ inserts a copy of m to every $q_{i,j}$, such that $p_j \in \delta_i$. Once m arrives, p_j executes $receive()$ and returns the tuple $\langle i, t_i, t_j, m\rangle$, where $t_i = C_i$ and $t_j = C_j$ are the clock values of the associated $transmit_i(m)$, and respectively, $m \leftarrow receive_j()$ calls. We assume zero propagation delay and efficient time-stamping mechanisms for t_i and t_j, as in [12]. Moreover, the timeslot duration, ξ, allows the transmission and reception of at least a single packet, see Property 1.

Property 1. Let $p_i \in P$, $p_j \in \delta_i$. At any point in time t_i in which node p_i transmits message m for a duration of ξ, node p_j receives m if there is no node $p_k \in (\delta_i \cup \delta_j) \setminus \{p_i\}$ that transmits during $[t_k, t_k + \xi)$, such that $[t_i, t_i + \xi)$ and $[t_k, t_k + \xi)$ intersect.

Interferences Wireless communications are subject to interferences when two or more neighboring nodes transmit *concurrently*, i.e., the packet transmission periods overlap or intersect. We model communication interferences, such as unexpected peaks in ambient noise level and concurrent transmissions of neighboring nodes, by letting the *(communication) environment* to selectively omit messages from the communication channels. The environment can also choose to inform the sending node about this error.

The environment can use the operation $omission_{i,j}(m)$ for removing message m from the communication channel, $q_{i,j}$, when p_i transmission of m to $p_j \in \delta_i$ is concurrent with the one of $p_k \in \Delta_i$. Immediately after $transmit_i(m)$, the environment selects a subset of p_i's neighbors, $Omit_m \subseteq \delta_i$, removes m from $q_{i,j} : p_j \in Omit_m$ and by that it prevents the execution of $m \leftarrow receive_j()$. Note that $Omit_m = \delta_i$ implies that no direct neighbor can receive message m. The environment uses Property 2 to decide whether to append an error, $TransmissionError$, to the error channel, e_i, which then in turn triggers the event $TransmissionError()$.

Property 2. Suppose that nodes $p_i, p_j \in P : p_i \in \delta_j$, concurrently transmit in $\alpha \in \mathbb{N}$ successive frames. We assume that p_j receives at least one message from p_i or that p_i receives a *(transmission) error*, i.e., invokes $TransmissionError_i()$, at least once.

The implementation of Property 2 can consider a basic technique for fault detection, say, based on the packet acknowledgment timeouts or ratio. Moreover, basic capabilities of the radio unit can also facilitate the implementation. For example, random separation time between timeslots [17, 19, 20]. Furthermore, RTS-CTS techniques can also address such implementation issues, as well as the use of dedicated hardware, for example, when using ultra-wideband for ranging and localization [15].

Self-stabilization Every node, $p_i \in P$, executes a program that is a sequence of *(atomic) steps*, a_i. The state, st_i, of node $p_i \in P$ includes p_i's variables, including the clocks and the program control variables, and the communication channels, $q_{i,j} : p_j \in$

δ_i. The *(system) configuration* is a tuple $c := (st_1, \ldots, st_{|P|})$ of node states. Given a system configuration, c, we define the set of *applicable steps*, $a = \{a_i\}$, for which p_i's state, st_i, encodes a non-empty communication channel or an expired timer. An *execution* is an unbounded alternating sequence $R := (c[0], a[0], c[1], a[1], \ldots)$ (Run) of configurations $c[k]$, and applicable steps $a[k]$ that are taken by the algorithm and the environment. The task $\mathscr{T}$ is a set of specifications and LE (legal execution) is the set of all executions that satisfy $\mathscr{T}$. We say that configuration c is *safe*, when every execution that starts from it is in LE. An algorithm is called *self-stabilizing* if it reaches a safe configuration within a bounded number of steps.
Task definition We consider the task $\mathscr{T}_{\text{TDMA}}$, that requires all nodes, p_i, to have data packet timeslots that are unique within Δ_i. We note that $\mathscr{T}_{\text{TDMA}}$'s requirements obviously hold when the ratio between the extended degree and the frame size is less than one, i.e., there is no timeslot exhaustion when $\forall p_i \in P : 1 < \tau/|\Delta_i|$. Therefore, the studied task also deals with timeslot exhaustion by delivering busy channel indications, $\perp$, to the nodes for which there were no timeslot left, and thus cannot transmit data packets. We define LE_{TDMA} to be the set of legal executions, R, for which $\forall p_i \in P$: $(((s_i \in [0, \tau-1]) \wedge (p_j \in \Delta_i)) \Rightarrow s_i \neq s_j) \vee (s_i = \perp \Rightarrow \forall t \in [0, \tau-1]\ \exists p_j \in \Delta_i : s_j = t)$ holds in all of R's configurations.

3 Basic Results

We establish a basic limitation of the bandwidth utilization for TDMA algorithms in wireless ad hoc networks. Before generalizing the limitation, we present an illustrative example (Lemma 1) of a starting configuration for which $\tau < \max(2\delta, \chi_2)$, where $\varepsilon > 0$, and χ_2 is the chromatic number for distance-2 vertex coloring.

Lemma 1. *Let $n \in \mathbb{N}$, $\varepsilon > 0$ and $\tau \leq (2-\varepsilon)\delta$. Suppose that the communication graph, $G := (\{p_0, \ldots p_\delta\}, E)$, has the topology of a star, where the node p_δ is the center (root) node and $E := \{p_\delta\} \times L$, where $L := \{p_0, \ldots p_{\delta-1}\}$ are the leaf nodes. There is a starting configuration, $c[x]$, and an execution, R, which follows $c[x]$, during which not all nodes, but one, are allocated with a timeslot, and yet the ones that are allocated follow a fixed schedule (with constant communication delay).*

Proof. We show by induction on x that all nodes in L are allocated with a timeslot in a fixed schedule, but p_δ is never properly allocated in R. Suppose that in the starting configuration, $c[x]$, all leaf nodes, $p_i \in L$, have a TDMA schedule in which their broadcasting timeslot is $s_i = s \in [0, \tau-1]$ (same value for all), and their clock values, $C_i = i(2-\varepsilon)\xi$, where ξ is the timeslot size. We assume that, in $c[x]$, there is no message in transit and that the communication queues are empty. Thus, p_i can only transmit a packet in the step $a[x]$ that immediately follows $c[x]$. However, as long as p_i's communication channels are empty, node p_i does not receive any packet. In this case, we say that p_i's timeslot does not change, and the schedule is fixed. Similarly, p_i clock values are not adjusted. Suppose that p_δ timeslot is $s_\delta = s \in [0, \tau-1]$. Let p_δ attempt to transmit in $a[x]$. We note that for any clock value that p_n may have in $c[x]$, there exists $p_i \in L$, such that the transmissions of p_i and p_n intersect. By definition, the environment can add the event *TransmissionError*, to p_n error channel, e_n and let the event go unnoticed with respect to p_i error and communication channels. Thus, Property 2 holds. The lemma is proved since $c[x+1]$ follows the definition of $c[x]$. We note that the same proof holds when each leaf node, $p_i \in L$, has a set of neighbors, $\delta_i \setminus \{p_\delta\}$, that do not include p_δ. The clock of each such neighbor, $p_j \in \delta_i$, can synchronize with the one of p_i and its timeslot, $s_j \neq s_i$ can allow communication with p_i once in every TDMA frame. □ □

The proof of Lemma 1 considers that case of $\tau \leq (2-\varepsilon)\delta$ and the star topology graph. We note that the same scenario can be demonstrated in every graph that includes a node with the degree δ. Thus, we can establish a general proof for $\tau < \max((2-\varepsilon)\delta, \chi(G^2))$ using the arguments in the proof of Lemma 2, where χ_2 is the chromatic number when considering distance-2 coloring.

Lemma 2. *Let $\xi \in \mathbb{R}, \tau \in \mathbb{N}$ and $S := \{[a\xi, (a+1)\xi) \colon a \in [0, \tau-1]\}$ be a partition of $[0, \xi\tau)$. The intervals $C := \{[b_i, b_i + \xi) \colon b_i \in \mathbb{R}\}_i$ intersects maximum $2|C|$ elements of S.*

Proof. Suppose that $[b, b+\xi) \in C$ intersects $I := [a\xi, (a+1)\xi) \in S$ for some a. Either $I = [b, b+\xi)$, $b \in I$ or $b + \xi \in I$. Therefore, any element $[b_i, b_i + \xi)$ of C intersects maximum 2 elements of S, one that contains b_i and one that contains $b_i + \xi$. □ □

4 Self-stabilizing TDMA Allocation and Alignment Algorithm

We propose Algorithm 1 as a self-stabilizing algorithm for the $\mathscr{T}_{\text{TDMA}}$ task. The nodes transmit data packets, as well as control packets. Data packets are sent by active nodes during their data packet timeslots. The passive nodes listen to the active ones and do not send data packets. Both active and passive nodes use control packets, which include frame information that includes recently received packets from direct neighbors. Each node aggregates the frame information it receives. It uses this information for avoiding collisions, acknowledging packet transmission and resolving hidden node problems. Passive nodes, p_i, can become active by uniformly, at random, selecting timeslots, s_i, that are not used by active nodes, sending a control packet in s_i and waiting for confirmation. Once p_i succeeds, it becomes an active node that uses timeslot s_i for transmitting data packets. Node p_i becomes passive whenever it learns about conflicts with nearby nodes. For example, when a data packet transmission fails, p_i, concludes that its data packet collided.

The hidden node problem refers to cases in which node p_i has two neighbors, $p_j, p_k \in \delta_i$, that use intersecting timeslots. The algorithm uses random back off techniques for resolving this problem in a way that assures at least one successful transmission from all active and passive nodes within $\mathscr{O}(\tau)$, and respectively, $\mathscr{O}(1)$ frames in expectation. The passive nodes count a random number of unused timeslots before transmitting a control packet. The active nodes use their clocks for defining frame numbers. They count down only during TDMA frames whose numbers are equal to s_i, where $s_i \in [0, \tau-1]$ is p_i's data packet timeslot. These back off processes connect

Algorithm 1: Self-stabilizing TDMA Timeslot Allocation, code for node p_i

```
$status_i$: current node status in $\{\mathsf{active}, \mathsf{passive}\}$
$s_i$: current data packet timeslot in $[0, \tau - 1]$
$lastTx_i$: type of the last transmission in $\{passiveCSMA, activeCSMA, activeTDMA\}$
$wait_i, waitAdd_i$: current back off countdown in $[0, maxWait]$
$FI_i := \{id_k, type_k, occurrence_k, rxTime_k\}_k \subset \mathscr{F}\mathscr{I}$: frame information
$timeOut$: age limit of an element in a frame information set
$BackOff() :=$ let $(tmp, r) \leftarrow (waitAdd_i, random([1, 3\Delta]))$; return $(r + tmp, 3\Delta - r)$
$frame() :=$ the current frame number $(GetClock() \div \xi\tau) \bmod \tau$,
$Slot(t) := (t \div \xi \bmod \tau), s() := Slot(GetClock())$ convert time to slot numbers
$Local(set) := \{\langle \bullet, \mathsf{local}, \bullet \rangle \in set\}$ $Ids(set) := \{id \in \mathsf{ID} : \langle id, \bullet \rangle \in set\}$
$Used(set) := \bigcup_{\langle \bullet, t_k \rangle \in set} [Slot(t_k), Slot(t_k + \xi - 1)]$; $Unused(set) := [0, \tau - 1] \setminus Used(set)$
$ConflictWithNeighbors(set) := (id_i \notin Ids(set) \vee s_i \in [Slot(t_i), Slot(t_i + \xi)] \vee$
$\exists_{\langle k, \bullet, rxTime \rangle \in set, k \neq id_i} : s_i \in [Slot(rxTime - t_j + t_i), Slot(rxTime - t_j + t_i + \xi)])$
$AddToFI(set, offset) := FI_i \leftarrow FI_i \cup \{\langle x, y, \mathsf{remote}, (z + \max\{0, offset\}) \bmod c \rangle : \langle x, y, \bullet, z \rangle \in set\}$
$IsUnused(s) := s \in Unused(FI_i) \vee (Unused(FI_i) = \emptyset \wedge s \in Unused(Local(FI_i)))$

1  upon $timeslot()$ do
2      if $s() = s_i \wedge status_i = \mathsf{active}$ then
3          transmit$(\langle status_i, Local(FI_i), MAC\_fetch() \rangle)$; $lastTx_i \leftarrow activeTDMA$;
4      else if $\neg(status_i = \mathsf{active} \wedge frame() \neq s_i)$ then
5          if $IsUnused(s()) \wedge wait_i \leq 0$ then
6              transmit$(\langle status_i, Local(FI_i), 0 \rangle)$; $\langle wait_i, waitAdd_i \rangle \leftarrow BackOff()$;
7              if $status_i = \mathsf{active}$ then $lastTx_i \leftarrow activeCSMA$;
8              else $\langle s_i, status_i, lastTx_i \rangle \leftarrow \langle s(), \mathsf{active}, passiveCSMA \rangle$
9          else if $wait_i > 0 \wedge IsUnused((s() - 1) \bmod \tau)$ then $wait_i \leftarrow \max\{0, wait_i - 1\}$
10     $FI_i \leftarrow \{\langle \bullet, rxTime \rangle \in FI_i : GetClock() < (timeOut + rxTime) \bmod c\}$;

11 upon $TransmissionError()$ do
12     if $lastTx_i \neq activeCSMA$ then $\langle \langle wait_i, waitAdd_i \rangle, status_i \rangle \leftarrow \langle BackOff(), \mathsf{passive} \rangle$

13 upon $\langle j, t_j, t_i, \langle status_j, FI_j, m' \rangle \rangle \leftarrow$ receive() do
14     if $ConflictWithNeighbors(FI_j) \wedge status_i = \mathsf{active}$ then
           $\langle \langle wait_i, waitAdd_i \rangle, status \rangle \leftarrow \langle BackOff(), \mathsf{passive} \rangle$;
15     if $status_j = \mathsf{active}$ then
16         if $m' \neq \bot$ then $FI_i \leftarrow \{\langle id_i, \bullet \rangle \in FI_i : id_i \neq j\} \cup \{\langle j, \mathsf{message}, \mathsf{local}, t_i \rangle\}$
17     else if $t_j = t_i \wedge Slot(t_j) \notin Used(FI_i)$ then
18         $FI_i \leftarrow \{\langle id_i, \bullet \rangle \in FI_i : id_i \neq j\} \cup \{\langle j, \mathsf{welcome}, \mathsf{local}, t_i \rangle\}$;
19     if $t_i < t_j$ then
20         $AdvanceClock(t_j - t_i)$; $FI_i \leftarrow \{\langle \bullet, (rxTime + t_j - t_i) \bmod c \rangle : \langle \bullet, rxTime \rangle \in FI_i\}$;
21         if $s_i \in Used(Local(FI_i))$ then
               $\langle \langle wait_i, waitAdd_i \rangle, status_i \rangle \leftarrow \langle BackOff(), \mathsf{passive} \rangle$
22     $AddToFI(FI_j, t_i - t_j)$;
23     if $m' \neq \bot$ then $MAC\_deliver(m')$
```

all direct neighbors and facilitate clock synchronization, timeslot alignment and timeslot assignment. During legal executions, in which all nodes are active, there are no collisions and each node transmits one control packet once every τ frames.

Algorithm details The node status, $status_i$, is either active or passive. When it is active, variable s_i contains p_i timeslot number. We use $lastTx_i$ for distinguishing between data and control packet transmissions.

The frame information is the set $FI_i := \{id_k,\ type_k,\ occurrence_k,\ rxTime_k\}_k \subset \mathscr{F}\mathscr{I} = \mathsf{ID} \times \{\mathsf{message},\ \mathsf{welcome}\} \times \{\mathsf{remote},\ \mathsf{local}\} \times \mathbb{N}$ that contains information about recently received packets, where $\mathsf{ID} := \{\perp\} \cup \mathbb{N}$ is the set of possible ids and the null value denoted by $\perp$. An element of the frame information contains the id of the sender id_k. The type $type_k = \mathsf{message}$ indicates that the sender was active. For a passive sender $type_k = \mathsf{welcome}$ indicates that there was no known conflict when this element was added to the local frame information. If $occurrence_k = \mathsf{local}$, the corresponding packet was received by p_i, otherwise it was copied from a neighbor. The reception time $rxTime_k$ is the time when this packet was received, regarding the local clock C_i, i.e., it is updated whenever the local clock is updated. The algorithm considers the frame information to select an unused timeslot. An entry in the frame information with timestamp t covers the time interval $[t, t+\xi]$.

Nodes transmit control packets according to a random back off strategy for collision avoidance. The passive node, p_i, chooses a random back off value, stores it in the variable $wait_i$, and uses $wait_i$ for counting down the number of timeslots that are available for transmissions. When $wait_i = 0$, node p_i uses the next unused timeslot according to its frame information. During back off periods, the algorithm uses the variables $wait_i$ and $waitAdd_i$ for counting down to zero. The process starts when node p_i assigns $wait_i \leftarrow waitAdd_i + r$, where r is a random choice from $[1, 3\Delta]$, and updates $waitAdd_i \leftarrow 3\Delta - r$, cf. $BackOff()$.

The node clock is the basis for the frame and timeslot starting times, cf. $frame()$, and respectively, $s()$, and also for a given timeslot number, cf. $Slot(t)$. When working with the frame information, set, it is useful to have restriction by local occupancies, cf. $Local(set)$, to retrieve its ids, cf. $Ids(set)$, and to list the sets of used and unused timeslots, cf. $Used(set)$, and respectively, $Unused(set)$. We check whether an arriving frame information, set, conflicts with the local frame information that is stored in FI_i, cf. $ConflictWithNeighbors(set)$, before merging them together, cf. $AddToFI(set, offset)$, after updating the timestamps in set, which follow the sender's clock.

Node p_i can test whether the timeslot number s is available according to the frame information in FI_i and p_i's clock. Since Algorithm 1 complements the studied lower bound (Section 3), the test in $IsUnused(s)$ checks whether FI_i encodes a situation in which there are no unused timeslots. In that case, $IsUnused(s)$ tests whether we can say that s is unused when considering only transmissions of direct neighbors. The correctness proof considers the cases in which $\tau > 2\Delta$ and $\tau > 4\delta$. For the former case, Lemma 4 shows that there is always an unused timeslot s' that is not used by any neighbor $p_j \in \Delta_i$, whereas for the latter case, Lemma 5 shows that for any neighbor $p_j \in \delta_i$, there is a timeslot s'' for which there is no node $p_k \in \delta_i \cup \delta_j \cup \{p_j, p_i\}$ that transmits during s''.

The code of Algorithm 1 considers three events: (1) periodic timeslots (line 1), (2) transmission errors (line 11) and (3) reception of a packet (line 13).

(1) *$timeslot()$, line 1:* Actives nodes transmit their data packets upon their timeslot and update their $lastTx_i$ (line 3). Passive nodes transmit control packets when the back off counter, $wait_i$, reaches zero (line 6) and updates $lastTx_i$ afterwards. Note that passive nodes count only when the local frame information says that the previous timeslot was unused (line 9). Active nodes also send control packets, but rather

than counting all unused timeslots, they count only the unused timeslots that belong to frames with a number that matches the timeslot number, i.e., $frame() = s_i$ (line 4).
(2) *$TransmissionError()$, line11:* Transmission errors indicate failure of the previous attempt to transmit, i.e., due to concurrent transmissions or transient faults. Active nodes become passive when they learn whose data packets collide (line 12). Passive nodes always strive to become active by sending control packets during timeslots that they view as unused. This implies that, during convergence, control packets can collide with other control packets or even with data packets. We note that passive do not become active when they receive transmission errors for their control packets. Moreover, the correctness proof shows that no collision occurs during a legal execution.
(3) ***receive**(), line 13:* Active nodes, p_i, become passive when they identify conflicts in FI_j between their data packet timeslots, s_i, and data packet timeslots, s_j of other nodes $p_j \in \Delta_i$ (line 14). When the sender is active, the receiver records the related frame information. Note that the payload of data packets is not empty in line 16, c.f., $m' \neq \perp$. Passive nodes, p_j, aim to become active. In order to do that, they need to send a control packet during a timeslot that all nearby nodes, p_i, view as unused $Slot(t) \notin Used(FI_i)$, where t is the packet sending time. Therefore, when the sender is passive, and its data packet timeslots are aligned, i.e., $t_i = t_j$, node p_i welcomes p_i's control packet whenever $Slot(t_j) \notin Used(FI_i)$. Algorithm 1 uses a self-stabilizing clock synchronization algorithm that is based on the converge-to-the-max principle [12]. When the sender clock value is higher (line 19), the receiver adjusts its clock value and the timestamps in the frame information set, before validating its timeslot, s_i, (lines 20 to 21). The receiver can now use the sender's frame information and payload (lines 22 to 23).
Correctness The proof of Theorem 1 starts by showing that p_i's active neighbors in δ_i stop interfering with each other's communications (Lemma 3). Then the proof shows the existence of unused timeslots by considering the cases in which $\tau > 2\Delta$ and $\tau > 4\delta$ (Lemmas 4, and respectively, 5). This facilitates the proof of network connectivity (Lemma 6), clock synchronization (Theorem 2) and bandwidth allocation (Theorem 3).

Theorem 1. *Algorithm 1 is a self-stabilizing implementation of task $\mathscr{T}_{\text{TDMA}}$ that converges within $\mathscr{O}(\mathsf{diam} \cdot \tau^2 + \tau^3\delta)$ starting from an arbitrary configuration. In case the system happens to have access to external time references, i.e., start from a configuration in which clocks are synchronized, the convergence time is within $\mathscr{O}(\tau^3)$, and $\mathscr{O}(\tau^3\delta)$ steps when $\tau > 2\Delta$, and respectively, $\tau > 4\delta$.*

We bound the duration in which direct and active neighbors interfere with each other's communications (Lemma 3). The proof argues that nodes, $p_i \in P$, can invoke $TransmissionError()$ within a bounded time and set $status_i \leftarrow \mathsf{passive}$. The proof considers the following definitions. Given a configuration c, we denote by $\mathscr{A}(c) = \{p_i \in P : status_i = \mathsf{active} \wedge wait_i = 0\}$ for the set of active nodes, and by $\mathscr{P}(c) = P \setminus \mathscr{A}(c)$ the set of passive ones. Let R be an execution, $a_i[x]$ and $a_j[x']$ two steps that include the $transmit_i(m)$, and respectively, $transmit_j(m')$ operations, and $a_k[x'']$ is a step that includes either the $obmission_k(m)$ or $receive(m)$ operations, where $p_i, p_j, p_k \in P$ and $x \leq x' < x''$. We note that $a_i[x]$ and $a_j[x']$ include concurrent transmissions. A *frame* for a node $p_i \in P$ is the time between two successive events of $timeSlot(0)$. We say that the step sequence $(a_i[x_\ell])_\ell : p_i \in P$ has ℓ *successive transmissions* if $a_i[x_k]$ includes a transmission operation, and each of the pairs $a_i[x_k], a_i[x_{k+1}] \in (a_i[x_\ell])_\ell$ occurs in successive frames with respect to p_i's clock. We say that $(a_i[x_\ell])_\ell$ and $(a_j[x'_\ell])_\ell$ are *sequences of pairwise concurrent successive transmissions* of $p_i \in P$ and, respectively, $p_j \in P$ when $(a_i[x_\ell])_\ell$ and $(a_j[x'_\ell])_\ell$ are successive transmission sequences, and for all l, the step $a_i[x_\ell]$ includes a transmission operation that is concurrent to the one included in $a_j[x'_\ell]$.

Lemma 3. *Let R be an execution of Algorithm 1 that starts from an arbitrary configuration $c[x]$. Let $p_i \in \mathscr{A}(c[x])$ be an active node and $p_j \in \delta_i \cap \mathscr{A}(c[x])$ be p_i's neighbor. Let $(a_i[x_\ell])_\ell$ and $(a_j[x'_\ell])_\ell$ maximal sequences of pairwise concurrent successive transmission of p_i and, respectively, p_j. We have that $|((a_i[x_\ell])_\ell| \leq \alpha$.*

Proof. Suppose, by the way of a proof by contradiction, that R includes sequences, $(a_i[x_\ell])_\ell$ and $(a_j[x'_\ell])_\ell$, of pairwise concurrent successive transmissions that are longer than $\alpha < |(a_i[x_\ell])_\ell|$. We show that within $|(a_i[x_\ell])_\ell|$ pairwise concurrent successive transmissions, either p_i, p_j, or both become passive; $\mathscr{P}(c[x_{l'}]) : l' \leq \alpha$. Thus, a contradiction since the sequence is interrupted. Note that by the definition of sequences of pairwise concurrent successive transmission, nodes p_i and p_j, concurrently transmit in successive frames. By Property 2, within α such frames, $p_k : k \in \{i, j\}$ execute a step, a_k, that invokes $TransmissionError_k()$ and by line 12 holds $status_k =$ passive in the configuration that immediately follows a_k. So, the transmit operation in line 3 is not invoked until the node is reactivated. Hence, the lemma. □ □

Communication among neighbors is possible only when there are timeslots that are free from transmissions in the local neighborhood. Lemma 4 assumes that $\tau > 2\Delta$ and shows that every node, $p_i \in P$, has an unused timeslot, s, with respect to p_i's clock, that satisfies the conditions of Property 1 with respect to *all* of p_i's neighbors $p_j \in \delta_i$. The proof considers the definitions of the start and the end of frames and timeslots, as well as unused timeslots. A configuration, $c_i^{FrameStart}[x_\ell] = c[x]$, in which $GetClock_i() \bmod (\xi\tau) = 0$ holds, marks the start of one of p_i's frames. This frame ends when the next frame starts, i.e., the next configuration $c_i^{FrameStart}[x_{\ell+1}]$. A timeslot of p_i is, respectively, bounded by two successive configurations $c_i^{TimeSlot}[x_\ell]$ and $c_i^{TimeSlot}[x_{\ell+1}]$, such that in those configurations $GetClock_i() \bmod \xi = 0$ holds. The slot number for this timeslot is given as $GetClock_i() \div \xi \bmod \tau$ at configuration $c_i^{TimeSlot}[x_\ell]$. Given execution R, we denote a timeslot starting at $c_i^{TimeSlot}[x_\ell]$ as unused if there is no active node $p_j \in \Delta_i$ exists such that it has in R an intersecting data packet timeslot. Namely, there is no configuration $c_j^{TimeSlot}[x'_\ell]$ in R with slot number $GetClock_i() \div \xi \bmod \tau = s_j$ occurs before $c_i^{TimeSlot}[x_{\ell+1}]$ and a configuration $c_j^{TimeSlot}[x'_{\ell+1}]$ occurs after $c_i^{TimeSlot}[x_\ell]$.

Lemma 4. *Suppose that $\tau > 2\Delta$ and $p_i \in P$. Let R be an execution of Algorithm 1 that includes a complete frame start with respect to p_i's clock. Between any two successive frame starts, $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$, there is at least one unused timeslot.*

Proof. Let us consider all the configurations, c', that are between $c_i^{FrameStart}[x_\ell]$, and $c_i^{FrameStart}[x_{\ell+1}]$. Let S be the partition of p_i's frame in τ timeslots of length ξ and C be the maximal set of data packet timeslots of active nodes $p_j \in \Delta_i$ (and their respective clocks, C_j). We show that $\tau > 2\Delta$ implies the existence of at least one unused timeslot between $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$, by requiring that C's elements cannot interest all τ elements in S. The nodes p_j periodically transmit a data packet once every τ timeslots (line 2). Note that there are at most Δ active nodes, p_j, in all (possibly arbitrary) configurations c'. Namely, every p_j has a single data packet timeslot, s_j, but s_j's timing is arbitrary with respect to p_i's clock. By the proof of Lemma 2, C interests maximum $2|C|$ elements of the set S. Since $|S| = \tau$, $|C| \leq \Delta$, and the assumption that C's elements cannot interest all elements in S, we have $\tau > 2\Delta$ implies the existence of at least one unused timeslot between $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$. □ □

Lemma 5 extends Lemma 4 by assuming that $\tau > 4\delta$ and showing that every node, $p_i \in P$, has an unused timeslot, s, with respect to p_i's clock, that satisfies the conditions of Property 1 with respect to *one* of p_i's neighbors $p_j \in \delta_i$, rather than all p_i's neighbors $p_j \in \delta_i$, as in the proof of Lemma 4.

Lemma 5. *Suppose $\tau > 4\delta$, $p_i \in P$ and $p_j \in \delta_i$. Let R be an execution of Algorithm 1 that includes a complete frame with respect to p_i's clock. With respect to p_i's clock, between any two successive frame starts, $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$, there is at least one timeslot that is unused by any of the nodes $p_k \in \delta_i \cup \delta_j \cup \{p_j, p_i\}$.*

Proof. Let C be the maximum set of data packet timeslots of active nodes $p_j \in \delta_i \cap \delta_j \cap \{p_j\}$ (and their respective clocks, C_j). The proof follows by arguments similar to those of Lemma 4. We show that $\tau > 4\delta$ implies the existence of at least one timeslot that is unused by any of the nodes $p_k \in \delta_i \cup \delta_j \cup \{p_j, p_i\}$ between $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$, by requiring that C's elements cannot interest all τ elements in S, c.f., proof of Lemma 4 for S's definition. By the proof of Lemma 2, C interests maximum $2|C|$ elements of the set S. Since $|S| = \tau > 4\delta$, $|C| \leq 2\delta$, and the assumption that C's elements cannot interest all elements in S, we have $\tau > 4\delta$ implies the existence of at least one unused timeslot between $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$. □ □

Lemma 6 shows that the control packet exchange provides network connectivity. Recall that lemmas 4 and 5 imply that there is a single timeslot, s, that is unused with respect to the clocks of node p_i and *all*, respectively, *one* of p_i's neighbors. Lemmas 4 and 5 refer to the cases when $\tau > 2\Delta$ and $\tau > 4\delta$ for which Lemma 6 shows that the communication delay (during convergence) of the former case is δ times shorter than the latter. The proof shows that we can apply the analysis of [13], because the back off process of a passive node counts r unused timeslots, where r is a random choice in $[1, 3\Delta]$. The lemma statement denotes the latency period by $\ell := (1 - e^{-1})^{-1}$.

Lemma 6. *Let R be an execution of Algorithm 1 that starts from an arbitrary configuration $c[x]$. Let R' be a suffix of R that starts from a configuration $c[x + \mathcal{O}(timeOut)]$. Every expected $E(\beta)$ frames in R', node $p_i \in P$ receives at least one message from all direct* passive *neighbors, $p_j \in \delta_i$, where $E(\beta)$ is $(3/2)\ell$ frames if $2\Delta < \tau$, and $(3/2)\ell\delta$ frames if $\tau > 4\delta$. Moreover, every expected $E(\gamma)$ frames, p_i receives at least one message from all* active *neighbor, $p_k \in \delta_i$, where $E(\gamma)$ is $(3/2)\tau\ell$ frames if $2\Delta < \tau$, or $(3/2)\tau\ell\delta$ frames if $\tau > 4\delta$.*

Proof. Starting from an arbitrary configuration $c[x]$, within $\mathcal{O}(timeOut)$ steps a configuration $c[x+\mathcal{O}(timeOut)]$ is reached, such that in any frame information set, FI_i from any node $p_i \in P$, contains only elements added by an actual received packet. Lemmas 4 and 5, together with Property 1, are proving that there is for every $p_k \in \delta_i$ at least one timeslot such that p_k can use it to send a packet to p_i. Assume node p_k is passive. Node p_k counts down a random number of unused timeslots regarding FI_k. Since FI_k does not necessarily contain information about conflicting neighbors, i.e., $p_\ell, p_m \in \delta_k \cap \mathscr{A}$ whose data packet timeslots are intersecting, p_k could use a slot which is used by some neighbors. Suppose that some nodes in $\delta_k \cap \mathscr{A}$ are not in FI_k due to conflicts. Then they intersect maximum $2/3$ of the unused timeslots in FI_k, since two conflicting nodes can only intersect with three timeslots, but for each node we add two slots to our frame ($\tau > 2\Delta$). Therefore, a factor of $3/2$ is added to our expected value ℓ. The same holds if p_k is active. In the case of $\tau \in [4\delta+1, 2\Delta]$, there are maximum δ different timeslots for transmitting packets to all neighbors; one for each $p_k \in \delta_i$. The choice of a timeslot is random and, thus, there is an additional factor of δ to hit this timeslot. □ □

We borrow the proof of a self-stabilizing clock synchronization algorithm that is based on the converge-to-the-max principle [12].

Theorem 2. *Let R be an execution of Algorithm 1 that starts from an arbitrary configuration $c[x]$. Within expected $E(\Phi)$ frames, a configuration $c[x^{synchro}]$ is reached after which all clocks are synchronized, where $E(\Phi)$ is* $\mathsf{diam}(3/2)\tau\ell$ *when* $2\Delta < \tau$*, and* $\mathsf{diam}(3/2)\tau\ell\delta$ *when* $\tau > 4\delta$*.*

Proof Sketch. The correctness proof of the algorithm in [12] considers the set of nodes, $M \subseteq P$, that have the maximum clock value. The proof in [12] assumes that if node p_i transmits message m periodically, once every ϕ time units, then within Φ time units, $p_j \in \delta_i$ receives m. We argue that for the case, our clock model, which has zero clock skew, we can consider the expected time, $E(\Phi)$, for p_j to receive m, rather than the constant Φ. When the clock skews are zero, the assumption in [12] is required for arguing that the maximum clock values propagates among neighbors in a timely manner, rather than arguing that the clock offset among neighbors does not grow larger than a certain bound, as it is required when the clock skews are not zero. This facilitates the proof in [12], which argues by induction on M, that within a period of $\mathsf{diam} \cdot (\Phi + \phi)$, we have that M becomes P. Therefore, the proof of this theorem can substitute $(\Phi + \phi)$ with $E(\Phi)$. The proof of the clock synchronization algorithm in [12] consider warp around of the clock values, see Lemma 7 in [12]. In the first case, all clock values, C_i, are smaller than the maximal clock value, $c-1$, by at least the algorithm convergence time, i.e., $\forall_{p_i} : C_i \in [0, c-1-E(\Phi)\tau\xi]$. The proof of this case follows that arguments above. The second case, $\forall_{p_i} : C_i \in [0, E(\Phi)\tau\xi - 1] \vee C_i[c - E(\Phi)\tau\xi, c-1]$, is that all clocks are near wrapping around. Clocks can change from the lower interval to the higher one, but after expected $E(\Phi)\tau\xi$ rounds, all clocks have wrapped around, and reached the lower interval $[0, E(\Phi)\tau\xi - 1]$, or have left the lower interval by counting up normally, but not by calling *AdvanceClock*(). Thus, the proof of the second case is followed by the arguments of the first case. The third case supposes that there are nodes in configuration $c[x]$ whose clock values are in the range $[0 + E(\Phi)\tau\xi], c-1-E(\Phi)\tau\xi]$, and at least one with a clock in $[c - E(\Phi)\tau\xi, c-1]$.

When there is no node in $[c-2E(\Phi)\tau\xi, c-1-E(\Phi)\tau\xi]$, it takes $E(\Phi)\tau\xi$ steps from $c[x]$ until configuration $c[x']$ is reached. Between $c[x]$ and $c[x']$, all large clocks, $C \in [c-E(\Phi)\tau\xi, c-1]$, and those who adjusted to this value, have wrapped around. Now no clock is in $[c-E(\Phi)\tau\xi, c-1]$, and the first case applies. Suppose that immediately after $c[x']$, there is a node, p_j, with clock $C_j \in [c-2E(\Phi)\tau\xi, c-1-E(\Phi)\tau\xi]$, such that p_j does not adjust its clock to a large value. Then after $E(\Phi)\tau\xi$ steps from $c[x']$, a configuration $c[x'']$ is reached in which each node in P is expected to: (1) wrapped around a large clock, (2) have a large clock, or (3) have adjusted to p_j's clock. However, in $c[x'']$ it holds that p_j's clock gets large, and the rest of the proof follows by the arguments of the second case. □

Once the clocks are synchronized the TDMA timeslots are aligned, as well. Thus, the number of timeslots that conflicting nodes can block is $4/3$, as Lemma 7 shows, rather than $3/2$, as Lemma 6 showed (for the case of arbitrary clock offsets).

Lemma 7. *Let R be an execution of Algorithm 1 that starts from a configuration, $c[x^{synchro}]$, in which all clock are synchronized. Every expected $E(\beta')$ frames in R, node $p_i \in P$ receives at least one message from all direct* passive *neighbors, $p_j \in \delta_i$, where $E(\beta')$ is $(4/3)\ell$ frames if $2\Delta < \tau$, and $(4/3)\ell\delta$ frames if $\tau > 4\delta$. Moreover, every expected $E(\gamma')$ frames, p_i receives at least one message from all* active *neighbor, $p_k \in \delta_i$, where $E(\gamma')$ is $(4/3)\tau\ell$ frames if $2\Delta < \tau$, or $(4/3)\tau\ell\delta$ frames if $\tau > 4\delta$.*

Proof Sketch. Let $p_i \in P$ and partition the set $\Delta_i = A_i \uplus I_i \uplus P_i =: F$, where A_i is the set of active neighbors of p_i having a unique (from p_i's point of view) data packet time slot and I_i is the subset of active neighbors that interfere with each other, i.e., $\forall_{p_\ell \in I_i} \exists_{p_k \in I_i}$: p_ℓ's timeslots intersects p_k's. Note that p_ℓ and p_k using the same timeslot in this case and thus I_i intersects maximum $|I_i|/2$ timeslots. P_i is the set of passive neighbors of p_i. Then the data packet time slots of the nodes A_i are contained in FI_i. Maximum $2\Delta - |A_i| = 2(|I_i| + |P_i|) + |A_i|$ and minimum $2\Delta - |A_i| - |I_i|/2 = 1/2|I_i| + 2|P_i| + |A_i|$. Thus, maximum $1/2|I_i|$ out of $2|I_i|$ timeslots are unused regarding FI_i. Moreover, a random choice of a timeslot is successful with probability $3/4$. Therefore, a factor of $4/3$ to the expected communication delay. The same arguments hold for $\tau > 4\delta$. □

Once the clocks are synchronized, and the TDMA timeslots are aligned, Algorithm 1 allocates the bandwidth using distance-2 coloring. This happens within $\mathcal{O}(\tau^3)$, and $\mathcal{O}(\tau^3\delta)$ steps when $\tau > 2\Delta$, and respectively, $\tau > \max\{4\delta, \Delta+1\}$, see Theorem 3.

Theorem 3. *Let R be an execution that starts from an arbitrary configuration $c[x^{synchro}]$ in which all clocks are synchronized. Within expected $E(\zeta)$ frames from $c[x^{synchro}]$, the system reaches a configuration, $c[x^{alloc}]$, in which each node $p_i \in P_j$ has a timeslot that is unique in Δ_i, where $E(\zeta)$ is $(4/3)\tau\ell\max\{\alpha, (4/3)\tau\ell\}$ when $2\Delta < \tau$, or $(4/3)\tau\ell\max\{\alpha, (4/3)\tau\ell\}\delta$ when $\tau > 4\delta$.*

Proof Sketch. We have showed that active nodes get feedback within an expected time. Therefore, also conflicting active nodes. Active nodes with positive feedback stay active, but conflicting active nodes get a negative feedback and change their status to passive. Passive nodes are transmitting from time control packets. They are successful and stay active on this timeslot with probability $1-1/e$. Otherwise, they get a negative feedback within expected $\max\{\alpha,(4/3)\tau\ell\}$ if $2\Delta \leq \tau$, or $\max\{\alpha,(4/3)\tau\ell\}\delta$ if $\tau > 4\delta$, frames. Hence, the number of active nodes without conflicts is monotonically increasing until every node is active.

The convergence time of this timeslot assignment is dominated by the time for a successful transmission and the time for a negative feedback in case of a unsuccessful transmission. This leads to an expected convergence time of $(4/3)\tau\ell\max\{\alpha,(4/3)\tau\ell\}$, where the first factor $(4/3)\tau\ell$ is introduced by the delay for a negative feedback. $\square$

The proof of Theorem 1 is concluded by showing that configuration, $c[x^{alloc}]$ (Theorem 3), is a safe configuration with respect to LE_{TDMA}, see Lemma 8.

Lemma 8. *Configuration $c[x]$ is a safe configuration with respect to LE_{TDMA}, when (1) $\forall_{p_i,p_j\in P} : C_i = C_j$, (2) $\forall_{p_i\in P} : status_i = \mathsf{active}$, (3) $\forall_{p_i\in P}\forall_{p_j\in\Delta_i} : s_i \neq s_j$, (4) $\forall_{p_i\in P} : \forall_{p_j\in\Delta_i\cup\{p_i\}}\exists!_{\langle id,\mathsf{message},\bullet\rangle\in FI} : id = id_j$.*

Proof Sketch. Obviously, the properties of legal executions LE_{TDMA} are fulfilled for $c[x]$. Since the clocks are equal no node transmits a packet that is leading to clock adjustment of another node. Furthermore, the timeslot and frame numbers are synchronized. There are no collisions, since every node is transmitting at a unique timeslot. $\square$

5 Conclusions

This work considers fault-tolerant systems that have basic radio and clock settings without access to external reference for collision detection, time or position, and yet require constant communication delay. We study collision-free TDMA algorithms that have uniform frame size and uniform timeslots and require convergence to a data packet schedule that does not change. Our analysis considers the timeslot allocation aspects of the studied problem, together with transmission timing aspects. Interestingly, we show that the existence of the problem's solution depends on convergence criteria that include the ratio, τ/δ, between the frame size and the node degree. We establish that $\tau/\delta \geq 2$ as a general convergence criterion, and prove the existence of collision-free TDMA algorithms for which $\tau/\delta \leq 4$. Unfortunately, our result implies that, for our systems settings, there is no distributed mechanism for asserting the convergence criteria within a constant time. For distributed systems that do *not* require constant communication delay, we propose to explore such criteria assertion mechanisms as future work.

Acknowledgments This work would not have been possible without the contribution of Marina Papatriantafilou in many helpful discussions, ideas, problem definition and analysis.

References

[1] N. Abramson. Development of the ALOHANET. *Info. Theory, IEEE Trans. on*, 31(2):119–123, 1985.

[2] N. Alon and B. Mohar. The chromatic number of graph powers. *Combinatorics, Probability & Computing*, 11(1):1–10, 2002.

[3] M. Arumugam and S. Kulkarni. Self-stabilizing deterministic time division multiple access for sensor networks. *AIAA Journal of Aerospace Computing, Info., and Comm. (JACIC)*, 3:403–419, 2006.

[4] J. R. S. Blair and F. Manne. An efficient self-stabilizing distance-2 coloring algorithm. *Theor. Comput. Sci.*, 444:28–39, 2012.

[5] C. Busch, M. Magdon-Ismail, F. Sivrikaya, and B. Yener. Contention-free MAC protocols for asynchronous wireless sensor networks. *Distributed Computing*, 21(1):23–42, 2008.

[6] H. A. Cozzetti and R. Scopigno. RR-Aloha+: a slotted and distributed MAC protocol for vehicular communications. In *Vehicular Networking Conference (VNC), 2009 IEEE*, pages 1 –8, Oct. 2009.

[7] P. Danturi, M. Nesterenko, and S. Tixeuil. Self-stabilizing philosophers with generic conflicts. *ACM Tran. Autonomous & Adaptive Systems (TAAS)*, 4(1), 2009.

[8] M. Demirbas and M. Hussain. A MAC layer protocol for priority-based reliable multicast in wireless ad hoc networks. In *BROADNETS*. IEEE, 2006.

[9] S. Dolev. *Self-Stabilization*. MIT Press, 2000.

[10] T. Herman. Models of self-stabilization and sensor networks. In S. R. Das and S. K. Das, editors, *IWDC*, volume 2918 of *Lecture Notes in Computer Science*, pages 205–214. Springer, 2003.

[11] T. Herman and S. Tixeuil. A distributed TDMA slot assignment algorithm for wireless sensor networks. In *ALGOSENSORS*, volume 3121 of *Lecture Notes in Computer Science*, pages 45–58. Springer, 2004.

[12] T. Herman and C. Zhang. Best paper: Stabilizing clock synchronization for wireless sensor networks. In A. K. Datta and M. Gradinariu, editors, *SSS*, volume 4280 of *Lecture Notes in Computer Science*, pages 335–349. Springer, 2006.

[13] J.-H. Hoepman, A. Larsson, E. M. Schiller, and P. Tsigas. Secure and self-stabilizing clock synchronization in sensor networks. *Theor. Comput. Sci.*, 412(40):5631–5647, 2011.

[14] A. Jhumka and S. S. Kulkarni. On the design of mobility-tolerant TDMA-based media access control (MAC) protocol for mobile sensor networks. In T. Janowski and H. Mohanty, editors, *ICDCIT*, volume 4882 of *Lecture Notes in Computer Science*, pages 42–53. Springer, 2007.

[15] J. Johnson and B. Dewberry. Ultra-wideband aiding of gps for quick deployment of anchors in a gps-denied ad-hoc sensor tracking and communication system. In *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011)*, pages 3959–3966, Oregon Convention Center, Portland, Oregon, September 2011.

[16] S. S. Kulkarni and M. Arumugam. Transformations for write-all-with-collision model' . *Computer Communications*, 29(2):183–199, 2006.

[17] P. Leone, M. Papatriantafilou, and E. M. Schiller. Relocation analysis of stabilizing MAC algorithms for large-scale mobile ad hoc networks. In *5th Inter. Workshop Algo. Wireless Sensor Net. (ALGOSENSORS)*, pages 203–217, 2009.

[18] P. Leone, M. Papatriantafilou, E. M. Schiller, and G. Zhu. Analyzing protocols for media access control in large-scale mobile ad hoc networks. In *Workshop on Self-Organising Wireless Sensor and Comm. Net. (Somsed)*, 2009.

[19] P. Leone, M. Papatriantafilou, E. M. Schiller, and G. Zhu. Chameleon-MAC: adaptive and self-$\star$ algorithms for media access control in mobile ad hoc networks. In *12th Inter. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS'10)*, pages 468–488, 2010.

[20] P. Leone and E. M. Schiller. Self-stabilizing TDMA algorithms for dynamic wireless ad-hoc networks. *To appear in International Journal of Distributed Sensor Networks and also avalable in CoRR*, abs/1210.3061, 2012.

[21] T. Masuzawa and S. Tixeuil. On bootstrapping topology knowledge in anonymous networks. *ACM Trans. Auton. Adapt. Syst.*, 4(1):8:1–8:27, Feb. 2009.

[22] N. Mitton, E. Fleury, I. G. Lassous, B. Sericola, and S. Tixeuil. Fast convergence in self-stabilizing wireless networks. In *12th Int. Conf. Parallel and Distributed Systems (ICPADS'06)*, pages 31–38, 2006.

[23] M. Molloy and M. R. Salavatipour. A bound on the chromatic number of the square of a planar graph. *J. Comb. Theory, Ser. B*, 94(2):189–213, 2005.

[24] M. Mustafa, M. Papatriantafilou, E. M. Schiller, A. Tohidi, and P. Tsigas. Autonomous TDMA alignment for VANETs. In *76th IEEE Vehicular Technology Conf. (VTC-Fall'12)*, pages 1–5. IEEE, 2012.

[25] S. Pomportes, J. Tomasik, A. Busson, and V. Vèque. Self-stabilizing algorithm of two-hop conflict resolution. In *12th Inter. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS'10)*, pages 288–302, 2010.

[26] R. Scopigno and H. A. Cozzetti. Mobile slotted aloha for VANETs. In *70th IEEE Vehicular Technology Conf. (VTC-Fall'09)*, pages 1 – 5, 2009.

[27] V. Turau and C. Weyer. Randomized self-stabilizing algorithms for wireless sensor networks. In H. de Meer and J. P. G. Sterbenz, editors, *IWSOS/EuroNGI*, volume 4124 of *Lecture Notes in Computer Science*, pages 74–89. Springer, 2006.

[28] S. Viqar and J. L. Welch. Deterministic collision free communication despite continuous motion. In *5th Inter. Workshop Algo. Wireless Sensor Net. (ALGO-SENSORS)*, pages 218–229, 2009.

[29] F. Yu and S. Biswas. Self-configuring TDMA protocols for enhancing vehicle safety with dsrc based vehicle-to-vehicle communications. *Selected Areas in Communications, IEEE Journal on*, 25(8):1526 –1537, oct. 2007.

A.1.2 On the Trade-off Between Accuracy and Delay in Cooperative UWB Localization: Performance Bounds and Scaling Laws

“On the Trade-off Between Accuracy and Delay in Cooperative UWB Localization: Performance Bounds and Scaling Laws” Gabriel E. Garcia, Student Member, L. Srikar Muppirisetty, Elad M. Schiller, and Henk Wymeersch (submitted for publication).

This page is intentionally left blank.

On the Trade-off Between Accuracy and Delay in Cooperative UWB Localization: Performance Bounds and Scaling Laws

Gabriel E. Garcia, *Student Member, IEEE,* L. Srikar Muppirisetty, Elad M. Schiller, *Member, IEEE* and Henk Wymeersch, *Member, IEEE*

***Abstract*—Ultra-wide bandwidth (UWB) systems allow for accurate positioning in environments where global navigation satellite systems may fail, especially when complemented with cooperative processing. While cooperative UWB has led to centimeter-level accuracies, the communication overhead is often neglected. We quantify how accuracy and delay trade off in a wide variety of operation conditions. We also derive the asymptotic scaling of accuracy and delay, indicating that in some conditions standard cooperation offers the worst possible trade-off. Both avenues lead to the same conclusion: indiscriminately targeting increased accuracy incurs a significant delay penalty. Simple countermeasures can be taken to reduce this penalty and obtain a meaningful accuracy/delay trade-off.**

***Index Terms*—Ultra-wideband positioning, S-TDMA, MAC delay, navigation, positioning**

I. Introduction

POSITION information is a necessary part of today's location-aware applications [3], including inventory tracking in warehouses [4], habitat [5] and health monitoring [6]. Even though Global Navigation Satellite Systems (GNSSs) can help to provide position information in many situations, they may not be viable in weak signal environments such as urban canyons, or indoors [7]. In consequence, there exists an ongoing need for accurate positioning in scenarios where GNSS-only implementations are not feasible.

Ultra-wide bandwidth (UWB) ranging and communication has been shown to be a promising technology to tackle the positioning problem in GPS-challenged scenarios. Given the absolute bandwidths of more than 500 MHz employed by this pulse-based technology [8], UWB offers a range of characteristics to avail of, both in terms of communication and localization. UWB communication advantages include robustness against interference and mitigation of small-scale fading [9]. Moreover, considering a two-way time of-arrival (TW-TOA) ranging procedure, UWB enables accurate and reliable ranging, making it convenient for localization and navigation purposes [10]. Improving positioning accuracy has gathered a significant amount of attention in the research community. The general conclusion is that traditional methods, such as enhanced ranging [7], the use of more anchors, higher transmission powers [11], and the distribution and sharing of information over the network (cooperation) among nodes [12]–[14], all improve the positioning accuracy.

In practice, performance gains in terms of accuracy come at a cost in delay, due to the channel access required in the medium access control (MAC) layer. This cost, which was neglected in [7], [11]–[14], has been analyzed in [15], [16], for cooperative positioning and target tracking, respectively, though based solely on computer simulations. Moreover, the IEEE 802.11.b MAC considered in [15] leads to excessively pessimistic delays. Dedicated MAC protocols were discussed in [17]–[21]: a decentralized self-stabilizing MAC protocol suitable for cooperative UWB navigation in multi-hop networks was proposed in [17], while in [18] and [19], a distributed and decentralized scheduling for cooperative localization was explored. A MAC design for cooperative localization networks is investigated in [20], focusing exclusively on the analysis and design of the MAC protocol. Finally, the authors in [21] proposed enhancements to the IEEE 802.15.4a standard using a time division multiple access (TDMA) scheme for clique networks.

In this paper, we tackle the trade-off between accuracy and delay through performance bounds. First, we derive lower bounds on UWB positioning accuracy in terms of the position error bound (PEB) [11] and on the required MAC delay. We also evaluate two methods to reduce the delay: selective ranging [22] and eavesdropping [14]. Finally, we derive scaling laws for the PEB and MAC delay for several scenarios. Our specific contributions are as follows:

- A derivation of the PEB and minimum MAC delay for noncooperative and cooperative networks with selective ranging and eavesdropping, assuming spatial reuse TDMA (STDMA) and two-way ranging;
- A numerical evaluation of the PEB and MAC delay with parameters based on off-the-shelf UWB radios;
- Scaling laws on the positioning accuracy and minimum MAC delay for dense networks, for several distinct operating conditions; and
- The introduction of a delay/accuracy trade-off parameter, which can uniquely quantify the trade-off between PEB

G. Garcia, L.S. Muppirisetty and H. Wymeersch are with the Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden, e-mails: {ggarcia,srikar.muppirisetty,henkw}@chalmers.se. E.M. Schiller is with the Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden, e-mail: elad.schiller@chalmers.se. This research was supported, in part, by the European Research Council, under Grant No. 258418 (COOPNET), and the Swedish Research Council, under Grant No. 2010-5889, and by the EC, through project FP7-STREP-288195, KARYON (Kernel-based ARchitecture for safetY-critical cONtrol). Part of this work has been presented at IEEE Wireless Communications and Networking Conference 2013 [1], and published in IEEE Communication Letters [2].

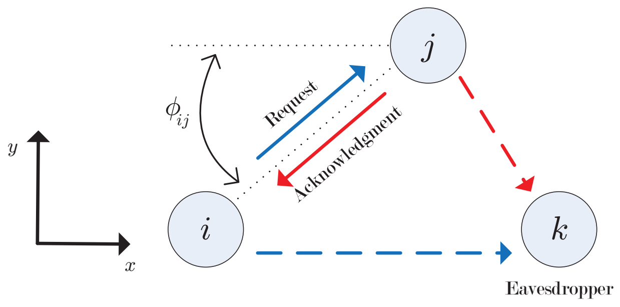


Figure 1. Two-dimensional graphic representation of the TW-TOA ranging transaction between nodes i and j, while node k performs an eavesdropping measurement.

and MAC delay as a function of the agent and anchor density.

The remainder of the paper is structured as follows. Section II presents the measurement and network models for UWB positioning. In Section III, we review the derivation of the lower bound on positioning accuracy and minimum MAC delay. Then, in Section IV we derive the scaling laws for the positioning accuracy and the MAC delay. Finally, numerical results are given in Section V, followed by the conclusions in Section VI.

II. System Model

A. UWB Positioning

We consider a wireless network consisting of M anchor nodes (collected in the set $\mathcal{S}_{\text{anchors}}$) with known positions and N agent nodes (collected in the set $\mathcal{S}_{\text{agents}}$) with unknown time-varying positions. The agents move in discrete time slots of duration T, and we focus on a specific time slot, at which the position of node i is denoted by $\mathbf{x}_i = [x_i\, y_i]^{\text{T}}$, with an a priori distribution $p(\mathbf{x}_i)$. Before moving on to the next time slot, the agent corrects its distribution to an a posteriori distribution $p(\mathbf{x}_i|\mathbf{z}_i)$, where $\mathbf{z}_i$ represents the available measurements. The time slot duration T is lower bounded by the measurement time T_{meas}, required by the agents to gather all the UWB measurements. We aim to quantify how traditional methods to improve accuracy, such as adding more anchors or agents, increasing the transmission power by means of the communication range, and the employment of cooperation among the agents affect the position accuracy (see Section III) and T_{meas} (see Section III.C). Note that T also comprises other components, such as a computation time and possibly a data exchange time. Since these operations may be performed with an alternative radio technology, these times are not strictly UWB-related, and are thus not included in our study. Moreover, dedicated methods to reduce the delay can also be derived for those operations [23], [24].

B. Measurement Models

We consider two types of UWB measurements: *two-way time-of-arrival* (TW-TOA) and *eavesdropping*, shown in Fig. 1. In a TW-TOA transaction, agent i sends a request to node j, which responds back with an acknowledgment. The set of neighbors of node i is denoted by $\mathcal{N}_i = \{j \neq i : i$ and j can communicate$\}$. When $j \in \mathcal{S}_{\text{agents}} \cup \mathcal{S}_{\text{anchors}}$, we say that the network is *cooperative*, while when j is constrained to belong to $\mathcal{S}_{\text{anchors}}$, we say that the network is *noncooperative*. In either case, both nodes i and j estimate the TOA for the request and acknowledgment, respectively. Agent i employs the round trip delay between itself and node j to estimate their distance. The TW-TOA measurement between agent i and node j is given by [14]:

$$z_{ij} = \underbrace{d_{ij} + \frac{cT_{\text{proc}}}{2}}_{=\mu_{ij}} + \frac{n_{ij}}{2} + \frac{n_{ji}}{2}, \tag{1}$$

where $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$, n_{ij} is the TOA error of the request from node i to node j and n_{ji} is the TOA error from the acknowledgment from node j to node i, c is the speed of light, and T_{proc} is a known processing time. The TOA errors are modeled as independent zero-mean Gaussian random variables: $n_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2)$ and $n_{ji} \sim \mathcal{N}(0, \sigma_{ji}^2)$. For the eavesdropping measurements, any node $k \in \mathcal{N}_i \cap \mathcal{N}_j$ is able to measure the TOA of the signals exchanged between nodes i and j. We obtain the eavesdropping measurement by subtracting those two TW-TOA measurements [14]:

$$z_{ij}^k = \underbrace{d_{ij} + d_{jk} - d_{ik} + cT_{\text{proc}}}_{=\mu_{ij}^k} + n_{ij} + n_{jk} - n_{ik}. \tag{2}$$

It is important to note that there exists a common noise term between (1) and (2), since one of the TOA measurements collected by node k depends on the TOA measurement of node j.

Under line-of-sight conditions, the ranging error variance between two nodes at a distance d apart can be modeled as in [12]:

$$\sigma^2(d) = \begin{cases} \sigma^2 & d \leq R_{\text{hw}} \\ \sigma^2 f(d) & R_{\text{hw}} < d \leq R_{\text{max}} \\ +\infty & d > R_{\text{max}}, \end{cases} \tag{3}$$

where R_{hw} is the range for which the variance is dominated by the hardware (e.g., ADC, filters), R_{max} is the maximum communication range, and $f(d)$ is a non-decreasing function with $f(R_{\text{hw}}) = 1$, capturing the degradation of the signal-to-noise ratio or the ranging information intensity [11]. Based on our off-the-shelf UWB hardware [25] and previous experiments [12], R_{hw} can be around 30 meters, which is sufficient for many indoor environments. In this paper, we limit ourselves to this range, i.e., $d \in (0, R_{\text{hw}}]$.

C. Network Model

We assume that nodes i and j can communicate with probability $P_{ij} = \exp\left(- \parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2 / (2R^2)\right)$, where R is the nominal communication range in meters, as in [26]. Node i can perform UWB measurements with any node in $\mathcal{N}_i$. For increased flexibility, we introduce the set $\mathcal{S}_i \subseteq \mathcal{N}_i$, which consists of *selected* neighbors with which node i performs TW-TOA ranging.

When two TW-TOA transactions are performed simultaneously, they can interfere if a node in one transaction can receive a packet from a node in the other transaction. As 802.15.4a radios use a common preamble, similar to our off-the-shelf radios, we do not rely on time hopping to deal with interference. Traditional protocols such as ALOHA or slotted ALOHA [27] have poor efficiency in terms of the successful number of transactions. Hence, similar to [21], [28], we consider an STDMA approach in which one TDMA slot is needed for a TW-TOA transaction, but two transactions can occur simultaneously if they do not interfere. The total STDMA delay within one time slot[1] depends only on the network topology in the current time slot. In contrast to [21] and [28], we do not consider ranging packet aggregation or other enhancements, as they are hard to justify with real hardware due to the tight synchronization constraints.

III. Lower Bound on Positioning Accuracy and MAC Delay

In this section, we will derive a generic expression for the PEB for both TW-TOA and eavesdropping measurements, for both cooperative and noncooperative networks. We first review the concepts of Fisher information matrix (FIM), equivalent Fisher information matrix (EFIM), and PEB. We end with a short discussion on bounds on the minimum MAC delay.

A. PEB: Basic Concepts

Let $\mathbf{x} = \left[\mathbf{x}_1^{\mathrm{T}}\mathbf{x}_2^{\mathrm{T}}\cdots\mathbf{x}_N^{\mathrm{T}}\right]^{\mathrm{T}}$ be the vector containing the positions of all agents and $\hat{\mathbf{x}}$ its estimate, based on the observation $\mathbf{z}$. The FIM is defined as $\mathbf{J} = -\mathbb{E}_{\mathbf{x},\mathbf{z}}\left\{\nabla_{\mathbf{x}}\nabla_{\mathbf{x}}^{\mathrm{T}}\log p(\mathbf{z},\mathbf{x})\right\}$ for random $\mathbf{x}$, and as $\mathbf{J}(\mathbf{x}) = -\mathbb{E}_{\mathbf{z}}\left\{\nabla_{\mathbf{x}}\nabla_{\mathbf{x}}^{\mathrm{T}}\log p(\mathbf{z}|\mathbf{x})\right\}$ for nonrandom $\mathbf{x}$ [29]. We will drop the argument $\mathbf{x}$ in the FIM as it will be clear from the context whether the variable is random or not. The EFIM of the first agent is defined as follows. Let

$$\mathbf{J} = \left[\begin{array}{cc}\mathbf{A} & \mathbf{B}\\ \mathbf{B}^{\mathrm{T}} & \mathbf{C}\end{array}\right], \tag{4}$$

where $\mathbf{A} \in \mathbb{R}^{2\times 2}$, $\mathbf{B} \in \mathbb{R}^{2\times 2(N-1)}$, and $\mathbf{C} \in \mathbb{R}^{2(N-1)\times 2(N-1)}$, then the EFIM for agent 1 is given by $\mathbf{J}_1^{\mathrm{E}} = \mathbf{A} - \mathbf{B}\mathbf{C}^{-1}\mathbf{B}^{\mathrm{T}}$ [11] . Using the Schur complement it is easy to verify that $\left[\mathbf{J}_1^{\mathrm{E}}\right]^{-1}$ is the top-left 2×2 block diagonal element of $\mathbf{J}^{-1}$. The EFIM for any agent i can be computed through a reordering of the agents. A similar definition of the EFIM holds for random $\mathbf{x}$. Finally, the PEB of the network is defined as $\mathcal{P} = \sqrt{\mathrm{tr}\left\{\mathbf{J}^{-1}\right\}/N}$, while the PEB of agent i is defined as $\mathcal{P}_i = \sqrt{\mathrm{tr}\left\{\left[\mathbf{J}_i^{\mathrm{E}}\right]^{-1}\right\}}$. From the theory of the Crámer-Rao lower bound (CRLB) [29], it is well known that, under suitable technical conditions, $\mathcal{P}^2 \le \mathbb{E}\left\{\|\mathbf{x}-\hat{\mathbf{x}}\|^2\right\}$, and $\mathcal{P}_i^2 \le \mathbb{E}\left\{\|\mathbf{x}_i-\hat{\mathbf{x}}_i\|^2\right\}$, where the expectation should be taken of the relevant random variables.

Note that the PEB is expressed in meters and that $\mathcal{P}$ and $\mathcal{P}_i$ are related through $\mathcal{P} = \sqrt{\sum_i \mathcal{P}_i^2/N}$.

[1]Note that we make a distinction of *time slots* of duration T, which capture the slow time scale of mobility, and much shorter *TDMA slots*, in which ranging transactions are scheduled.

B. PEB: Derivation

We again collect the positions of all agents in a vector $\mathbf{x} = \left[\mathbf{x}_1^{\mathrm{T}}\mathbf{x}_2^{\mathrm{T}}\cdots\mathbf{x}_N^{\mathrm{T}}\right]^{\mathrm{T}}$ and determine the FIM. For mathematical convenience and since we are focusing on a single time slot, we assume that every agent has an a priori distribution $p(\mathbf{x}_i)$, modeled as a symmetric Gaussian distribution with mean $\mathbf{m}_{\mathrm{prior},i}$ and variance $\sigma^2_{\mathrm{prior},i}$ per dimension. Recalling the definition of the set $\mathcal{S}_i \subseteq \mathcal{N}_i$, we construct a measurement vector $\mathbf{z}$ for both the noncooperative and cooperative cases as $\mathbf{z} = [\mathbf{z}_{ij}|i \in \mathcal{S}_{\mathrm{agents}},\ j \in \mathcal{S}_i]$, where $\mathbf{z}_{ij}$ contains the TW-TOA estimate between agent i and node j, as well as all the corresponding eavesdropping measurements at node k, $\mathbf{z}_{ij} = \left[z_{ij},\ \left\{z_{ij}^k \middle| k \in \mathcal{N}_i \cap \mathcal{N}_j\right\}\right]$.

Due to (1) and (2), $\mathbf{z}$ conditioned on $\mathbf{x}$ is a Gaussian random variable with mean $\boldsymbol{\mu}$, constructed from (1) and (2) in the same way as $\mathbf{z}$, and covariance matrix $\boldsymbol{\Sigma}$. As a result of the independence of the TW-TOA measurements, the covariance matrix $\boldsymbol{\Sigma}$ is a block diagonal matrix, with the block corresponding to $\mathbf{z}_{ij}$ given by

$$\mathbf{C}_{ij} \;=\; \mathbb{E}\left\{(\mathbf{z}_{ij}-\boldsymbol{\mu}_{ij})(\mathbf{z}_{ij}-\boldsymbol{\mu}_{ij})^{\mathrm{T}}\right\}, \tag{5}$$

with

$$\mathbb{E}\left\{(z_{ij}-\mu_{ij})^2\right\} = (\sigma_{ij}^2+\sigma_{ji}^2)/4 \tag{6}$$

$$\mathbb{E}\left\{(z_{ij}-\mu_{ij})(z_{ij}^k-\mu_{ij}^k)\right\} = \sigma_{ij}^2/2 \tag{7}$$

$$\mathbb{E}\left\{(z_{ij}^l-\mu_{ij}^l)(z_{ij}^k-\mu_{ij}^k)\right\} = \begin{cases}\sigma_{ij}^2+\sigma_{jk}^2+\sigma_{ik}^2 & k=l\\ \sigma_{ij}^2 & k\neq l.\end{cases} \tag{8}$$

Note that when there are only TW-TOA measurements, $\mathbf{C}_{ij}$ reverts to the scalar $(\sigma_{ij}^2+\sigma_{ji}^2)/4$. The FIM is now given by

$$\mathbf{J} \;=\; -\mathbb{E}_{\mathbf{x}}\left\{\nabla_{\mathbf{x}}\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\nabla_{\mathbf{x}}^{\mathrm{T}}\boldsymbol{\mu}\right\}+\mathbf{J}_{\mathrm{prior}}, \tag{9}$$

where $\nabla_{\mathbf{x}}$ denotes the derivative with respect to $\mathbf{x}$ and $\mathbf{J}_{\mathrm{prior}} = \mathrm{diag}\left[\sigma^2_{\mathrm{prior},1},\sigma^2_{\mathrm{prior},1},\ldots,\sigma^2_{\mathrm{prior},N},\sigma^2_{\mathrm{prior},N}\right]^{-1}$. The entries in the matrix $\nabla_{\mathbf{x}}\boldsymbol{\mu}^{\mathrm{T}}$ can be easily computed since they are all zero, except for $\partial\mu_{ij}/\partial\mathbf{x}_i = [\cos(\phi_{ij})\ \sin(\phi_{ij})]^{\mathrm{T}}$, $\partial\mu_{ij}^k/\partial\mathbf{x}_i = [\cos(\phi_{ij})-\cos(\phi_{ik})\ \sin(\phi_{ij})-\sin(\phi_{ik})]^{\mathrm{T}}$, $\partial\mu_{ij}^k/\partial\mathbf{x}_k = [\cos(\phi_{kj})-\cos(\phi_{ki})\ \sin(\phi_{kj})-\sin(\phi_{ki})]^{\mathrm{T}}$, and $\partial\mu_{ij}^k/\partial\mathbf{x}_j = [\cos(\phi_{ji})+\cos(\phi_{jk})\ \sin(\phi_{ji})+\sin(\phi_{jk})]^{\mathrm{T}}$, where ϕ_{ij} represents the angle between node i and node j with respect to to the horizontal axis (see Fig. 1). The entries in $\boldsymbol{\Sigma}^{-1}$ are also readily computed since $\boldsymbol{\Sigma}$ is block-diagonal so $\boldsymbol{\Sigma}^{-1}$ is block-diagonal as well (see Appendix A for additional details). Thus $\nabla_{\mathbf{x}}\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\nabla_{\mathbf{x}}^{\mathrm{T}}\boldsymbol{\mu}$ in (9) can be determined efficiently, even for large networks. Finally, the expectation over $\mathbf{x}$ in (9) can be performed through Monte Carlo integration.

Once the FIM is computed, the EFIM, and the PEB for individual nodes as well as for the entire network can be determined numerically.

C. Bounds on Minimum MAC Delay

Each TW-TOA transaction must be scheduled such that both nodes involved in the transaction are free from primary and secondary interference. Primary interference refers to a node not being able to transmit and receive at the same time, while secondary interference refers to a node not being able to receive multiple transmissions at the same time [30]. This can be cast as a coloring problem on a suitable communication graph [31]. In [1], we have constructed tight lower and upper bound on the minimum MAC delay, based on graph-theoretic arguments, in a complexity that is at most quadratic in the number of nodes. Due to space limitations, a detailed description is omitted here, and the reader is instead referred to [1], [2].

IV. Scaling Laws

While the numerical PEB and minimum MAC delay are useful to analyze particular networks, further insight can be gleaned from their asymptotic behavior, as the number of agents and anchors increases. The resulting scaling laws give fundamental understanding into the benefit and drawbacks of cooperation in UWB positioning, and simultaneously allow us to analyze methods to reduce the impact of the MAC delay, such as eavesdropping.

For reasons of tractability, we focus on dense networks [13], where the area remains fixed and the node density increases by adding more nodes into the network. Moreover, we assume the ranging error variance to be constant for all transactions, as motivated in Section II.B. A general expression for the FIM under these assumptions is provided in Appendix B, where for notation and mathematical convenience we assume that anchors can also initiate TW-TOA procedures with other nodes. We recall that the PEB is expressed in meters, while the minimum MAC delay has a unit of seconds.

A. Operating Conditions

We analyze the scaling behavior of the PEB and minimum MAC delay for six distinct operating conditions: (i) noncooperative (*No*), where agents perform TW-TOA with all anchors in communication range (i.e., $\mathcal{S}_i = \mathcal{N}_i \cap \mathcal{S}_{\text{anchors}}$); (ii) cooperative (*Co*), where agents perform TW-TOA with all nodes in communication range (i.e., $\mathcal{S}_i = \mathcal{N}_i$); (iii) noncooperative with anchors eavesdropping (*No-Ea*), where agents perform TW-TOA with all anchors while other anchors are able to perform eavesdropping measurements; (iv) noncooperative with all nodes eavesdropping (*No-E*), where agents perform TW-TOA with all anchors and all neighboring nodes are allowed to eavesdrop; (v) cooperative with anchors eavesdropping (*Co-Ea*), where agents perform TW-TOA with all nodes in communication range and anchors are able to eavesdrop; and (vi) cooperative with eavesdropping (*Co-E*), where agents perform TW-TOA with all nodes in communication range and all nodes are able to eavesdrop.

Theorem 1. *For a clique network, where measurements are given by (1)–(2), with a constant TOA variance, the PEB ($\mathcal{P}$), and the MAC delay ($\mathcal{M}$) for the six operating conditions scale as listed in Table I.*

Scenario	$\mathcal{P} \in$	$\mathcal{M} \in$
No	$\mathcal{O}\left(M^{-1/2}\right)$	$\mathcal{O}(MN)$
Co	$\mathcal{O}\left((M+N)^{-1/2}\right)$	$\mathcal{O}(MN+N^2)$
No-Ea	$\mathcal{O}(M^{-1})$	$\mathcal{O}(MN)$
No-E	$\mathcal{O}\left((3M^2+MN)^{-1/2}\right)$	$\mathcal{O}(MN+M^2)$
Co-Ea	$\mathcal{O}\left((M^2+MN)^{-1/2}\right)$	$\mathcal{O}(NM+N^2)$
Co-E	$\mathcal{O}((M+N)^{-1})$	$\mathcal{O}\left((M+N)^2\right)$

Table I
Scaling laws of PEB and MAC delay for the 6 operating conditions.

Proof: See Appendices C–H. ■

Remarks: As was already noted in [13], in terms of the asymptotic PEB, agents play the same roles as anchors in the *Co* case. This causes the PEB to go down rapidly with the total number of nodes. However, we see that the MAC delay scaling also treats anchors as agents, thus causing a quadratic scaling in terms of the number of agents. This is the main reason why indiscriminate cooperation is prohibitive in terms of delay. The noncooperative case with only anchors eavesdropping (*No-Ea*) shows how the PEB is improved further, just by letting neighboring anchors listen to the TW-TOA ranging procedures. The MAC delay scaling is the same as in the noncooperative scenario, as no additional TDMA slots are required to enable eavesdropping. Letting not only the anchors but also the agents eavesdrop (*No-E*) results in further enhancements in terms of PEB reduction. There is, however, an additional penalty in terms of delay, as anchors are allowed to range with each other, while agents eavesdrop. We clearly see the asymmetric role of agents and anchors in this scenario. The *Co-Ea* case, compared to standard cooperation, yields additional gains in terms of PEB. Note that in terms of the MAC delay, since there are no anchor-to-anchor transactions, but agent-to-agent transactions, the term M^2 present in the *No-E* case is replaced by N^2 in the *Co-Ea* case. In the last scenario (*Co-E*), an order of magnitude reduction in PEB can be achieved by an order of magnitude increase in the number of nodes (agent or anchors), and two orders of magnitude increase in terms of the MAC delay. The role of agents and anchors is again symmetric.

B. Trade-off Analysis

The scaling laws above depend on the rate at which the number of agents N increases with respect to the number of anchors M. To provide a unified view of the trade-off of the above scenarios, we will model $N = \kappa M^\rho$ and introduce the notion of the delay/accuracy trade-off parameter $\delta(\rho) \in \mathbb{R}$, where ρ is called the agent growth rate and $\kappa > 0$.

Definition 2 (Delay/accuracy trade-off parameter)**.** *Let* $N = \kappa M^\rho$, *so that the PEB scales as* $\mathcal{P} \in \mathcal{O}(f_{\mathcal{P}}(M,\rho))$*, while the MAC delay scales as* $\mathcal{M} \in \mathcal{O}(f_{\mathcal{M}}(M,\rho))$*. The delay/accuracy trade-off is determined by*

$$\delta(\rho) = -\lim_{M\to+\infty} \frac{\log f_{\mathcal{P}}(M,\rho)}{\log f_{\mathcal{M}}(M,\rho)}. \tag{10}$$

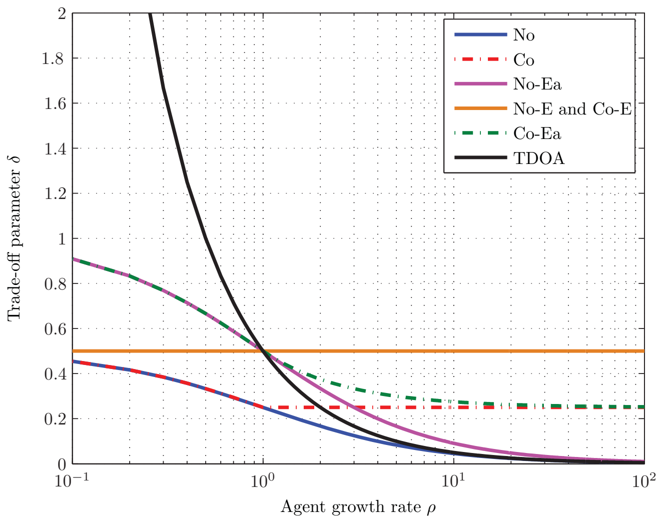


Figure 2. Delay/accuracy trade-off parameter δ as a function of the agent growth rate ρ (i.e., $N \propto M^{\rho}$) for the different operating conditions.

The trade-off parameter can be interpreted as the slope of the accuracy versus delay line in a log-log scale. Hence, operating conditions with a larger $\delta(\rho)$ will lead to a faster reduction in PEB as the delay increases than operating conditions with a smaller $\delta(\rho)$. In Fig. 2, we visualize the trade-off parameters for the six operating conditions as a function of the agent growth rate (note that the value of κ is irrelevant). Additionally, for the sake of completeness and comparison purposes, the trade-off analysis for a time difference of arrival (TDOA) scenario with N agents and M synchronized anchors is included. For this specific case the MAC delay scales as $\mathcal{M}_{\text{TDOA}} \in \mathcal{O}(N)$ and the PEB as $\mathcal{P}_{\text{TDOA}} \in \mathcal{O}(1/\sqrt{M})$, which can be easily derived from [32, Equation (21)].

We observe that when $\rho < 1$, so when anchors are added faster than agents, the best trade-off is achieved for TDOA followed by the *Co-Ea* and *No-Ea* cases. Interestingly, cooperation does not affect the trade-off (for example, *No* has exactly the same trade-off as *Co*), as the gain in PEB is canceled out by the increase in delay. For $\rho > 1$, when agents are added faster than anchors, the situation changes: cooperative methods exhibit a better trade-off than their non-cooperative counterparts including TDOA. For very large growth rates, *TDOA, No,* and *No-Ea*, have a δ that tends to zero, meaning that there is almost no gain in terms of PEB when adding a few anchors and many agents. Similarly, *Co* and *Co-Ea* converge to $\delta = 1/4$ for large values of ρ, since the few additional anchors that eavesdrop do not significantly affect the PEB or delay. The value of $\delta = 1/4$ should be interpreted as $\mathcal{M} \approx 1/\mathcal{P}^4$, so that a 50% reduction in PEB leads to a 16-fold increase in MAC delay. For $\rho > 1$, the best trade-off is offered by the *No-E* and *Co-E* cases. Interestingly, standard cooperation never offers the best trade-off.

V. Numerical Results and Discussion

In this section, we evaluate the PEB and the upper and lower bounds on the MAC delay for four of the five operating conditions from Section IV. We analyze the impact of the number of anchors, number of agents, the communication range, and cooperation among nodes. For each of the operating conditions, we also consider a selective variant where $\mathcal{S}_i$ can be a strict subset of $\mathcal{N}_i$.

A. Simulation Setup

We consider a 20 m $\times$ 20 m square area: anchors are placed according to a scaled network topology from [12, Fig. 13] while agents are uniformly distributed in the fixed area. Based on our experimental results with the P400 UWB radios [25], we consider a ranging standard deviation of 2 cm (irrespective of distance under line-of-sight propagation, as argued in Section II.B) and TDMA time slot duration of 20 ms. The a priori distributions of the agents' positions are Gaussian with unit variance. In addition to the scenarios *No*, *Co*, *No-E*, *Co-E*, we also introduce selective ranging, leading to additional operation conditions: Noncooperative Selective (*No-S*) and Noncooperative Eavesdropping Selective (*No-E-S*) (where $\mathcal{S}_i \subseteq \mathcal{N}_i \cap \mathcal{S}_{\text{anchors}}$, such that $|\mathcal{S}_i| \leq 4$), and Cooperative Selective (*Co-S*) and Cooperative Eavesdropping Selective (*Co-E-S*) (where $\mathcal{S}_i \subseteq \mathcal{N}_i$, such that $|\mathcal{S}_i| \leq 4$). The selection of the nodes in $\mathcal{S}_i$ is implemented using a distributed greedy algorithm to minimize the local PEB, described in [2].

B. Impact of Number of Anchors

The impact on the localization accuracy and MAC delay for a clique network with 10 agents and increasing number of anchors (from 2 to 10) is illustrated in Fig. 3. The upper and lower bounds for the MAC delay for a clique network are the same, thus, Fig. 3 only depicts the lower bounds. Each curve in the figure contains 9 markers, each marker corresponds to the increasing number of anchors M (2 to 10) from left to right. All scenarios show a decrease in the PEB when increasing the number of anchors as validated in previous works. However, the improvement in accuracy comes with a cost in delay, which is linear in M. From the figure, it is also clear that cooperation (*Co*) exhibits a poor delay/accuracy tradeoff. For example, consider a network with 5 anchors, for the *No* case the PEB is ≈ 1.3 cm, while for *Co* the PEB is ≈ 0.7 cm. However, the accuracy comes with a cost in delay to the amount of ≈ 1 s for *No* and ≈ 3.3 s for *Co*.

For the eavesdropping cases (*No-E* and *Co-E*), we observe an improvement with respect to the non-eavesdropping cases (*No* and *Co*) without incurring into any extra delay since the eavesdropping measurements do not require any scheduling. Hence all corresponding curves shift downward. We can observe that the results are in accordance with the scaling laws, where the reduction of the PEB and the increase in MAC delay are dependent on the number of anchors, with more PEB reduction for the eavesdropping cases while maintaining the same MAC delay similarly to the non-eavesdropping cases.

When selective ranging is employed, we see that for *No-S* the PEB no longer improves for $M \geq 4$ and the MAC delay stays constant just below 1 s. Thus, Fig. 3 only shows 3 distinct markers for *No-S*. The *Co-S* case corresponds to closely placed markers in Fig. 3, since the agents can always find enough neighbors (agents or anchors) to cooperate with.

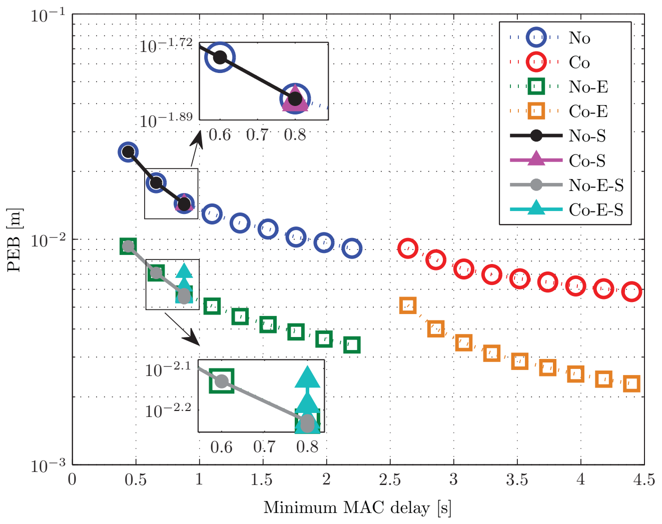


Figure 3. MAC delay and PEB lower bounds for a clique network with 10 agents and increasing number of anchors (2 to 10). Each curve consists of 9 markers, each marker representing the increase in the number of anchors M from left to right, except for the selective cases.

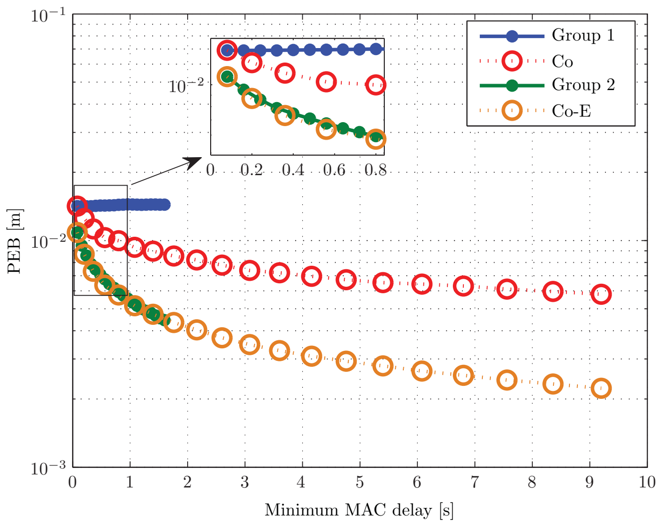


Figure 4. MAC delay and PEB lower bounds for a clique network with 3 anchors and increasing number of agents (1 to 20). Each curve consists of 20 markers, each marker representing the increase in the number of agents N from left to right. Group 1 consists the scenarios: *No*, *No-S*, and *Co-S*. Group 2 contains the cases: *No-E*, *No-E-S*, and *Co-E-S*.

The *No-E-S* and *Co-E-S* cases result in a reduction of the PEB without any additional MAC delay. The advantage of adding anchors is clearer for *Co-E-S* than for *Co-S*.

C. Impact of Number of Agents

Fig. 4 depicts the impact on the localization accuracy and the MAC delay for a clique network with 3 anchors and an increasing number of agents (from 1 to 20). Each curve in the figure consists of 20 markers, each one corresponding to the increasing number of agents N (1 to 20) from left to right. Note that the *No*, *No-S*, and *Co-S* cases coincide and therefore are grouped into one single label: Group 1. Similarly, the *No-E*, *No-E-S*, and *Co-E-S* are grouped into the label: Group 2. Once again, the lower and upper bounds on the MAC delay are the same, and only the lower bound is presented in Fig. 4.

The addition of agents to the network has no impact in the PEB for Group 1, but increases the MAC delay since more agent-to-anchor TW-TOA procedures need to be scheduled. For Group 2, adding agents to the network translates to a rapid decrease in the PEB, since for this particular group adding agents means more information, as agents are able to eavesdrop TW-TOA ranging transactions between agents and anchors. For the *Co* case, adding more agents decreases the PEB as compared with Group 1, since more agent-to-agent TW-TOA information is available in the network. However, cooperation comes with a cost in delay which grows quadratic in N. The *Co* PEB can be reduced without any additional MAC delay by eavesdropping, leading to the *Co-E* curve.

We conclude that full cooperation with many agents is not feasible when there are tight delay constraints, for example in the case of highly dynamic agents. Hence, theoretical cooperative gains cannot be exploited. We can observe the importance of the number of agents specially for the cooperative cases, with the corresponding cost in delay and PEB reduction consistent with the scaling laws.

D. Impact of Communication Range

For the analysis of the communication range variable we consider a network consisting of 20 agents and 13 anchors, and an increasing communication range R from 1 m to 30 m. Each marker in the curves represents the nominal communication range R used in the communication model (see Section II.C) increasing from left to right, i.e., the leftmost and rightmost markers of each curve represent $R = 1$ m, and $R = 30$ m, respectively. Figs. 5 and 6 show the influence in the PEB and MAC delay trade-off for the noncooperative cases (*No*, *No-S*, *No-E*, *No-E-S*) and the cooperative cases (*Co*, *Co-S*, *Co-E*, and *Co-E-S*), respectively. For low values of R, the lower and upper bounds on the minimum MAC delay do not coincide, though the bounds are sufficiently tight for all R under consideration.

In Fig. 5, as expected, increasing the communication range decreases the PEB since more agent-to-anchor transactions are injected into the network, though once again with a cost in delay. For the *No-S* case, the curve remains constant in both accuracy and delay once agents are able to communicate with at least four anchors (this happens when R is around 6 m). In contrast, even though *No-E-S* remains constant once agents can communicate with at least four nodes, increasing range allows for more nodes in the network to perform eavesdropping, hence the PEB improvement for this specific case results in vertical drop on PEB. This clearly shows that using high-power anchors is only meaningful with a selective ranging strategy. The *No-E* scenario outperforms *No*, once again with a cost in delay.

In Fig. 6 we observe that cooperative selective cases show a similar behavior as the noncooperative counterpart, though with more extreme values in terms of PEB and minimum MAC delay. The PEB decreases with increasing R while the MAC delay grows fast reaching up to $N \times M \times 20\,\text{ms} \approx 5\,\text{s}$, and

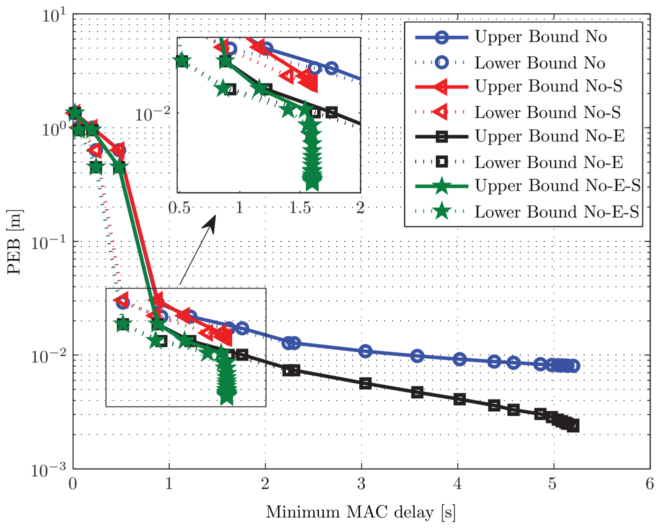


Figure 5. MAC delay and PEB lower bounds for a partially noncooperative connected network with 20 anchors and 13 agents. Each curve consists of 30 markers, each marker representing the increase in the nominal communication range R from left to right (1 to 30 m).

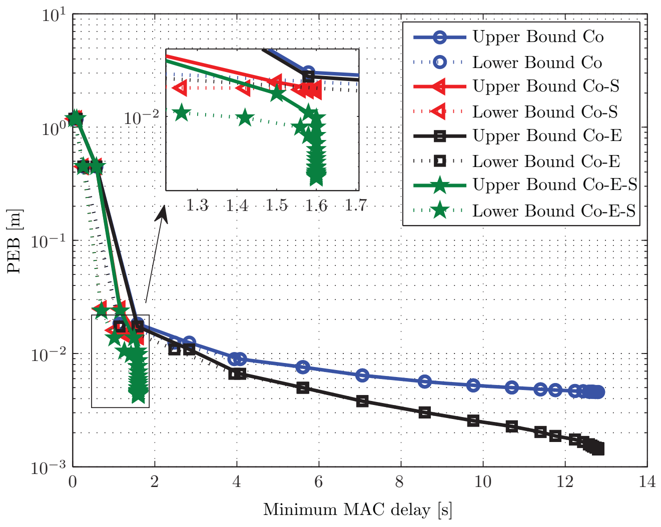


Figure 6. MAC delay and PEB lower bounds for a partially cooperative connected network with 20 anchors and 13 agents. Each curve consists of 30 markers, each marker representing the increase in the nominal communication range R from left to right (1 to 30 m).

$N\times(M+(N-1))\times 20\,\text{ms}\approx 13\,\text{s}$ for *No* and *Co*, respectively. Evidently, the latter shows that increasing R for a marginal gain in terms of accuracy can lead to large delays, specially when cooperation is implemented indiscriminately.

VI. Conclusions

We have investigated the interplay between UWB positioning accuracy and MAC delay. We presented lower bounds on the position accuracy and the MAC delay considering spatial time division multiple access for arbitrary finite networks. We have characterized the behavior for dense-location aware networks for the noncooperative and cooperative cases by means of the relevant scaling laws. We found that traditional methods to improve accuracy, such as increasing the number of anchors or the communication range, or the implementation of cooperation among nodes comes at a cost in terms of MAC delay. The latter has a direct impact in the update rate when dealing with dynamic networks with respect to mobility. Selective ranging and eavesdropping have been evaluated as possible methods to reduce the MAC delay with reasonable position accuracy. Noncooperative eavesdropping shows to outperform cooperative networks in terms of accuracy with reasonable delays. Finally, in terms of scaling, we found that, under certain conditions, standard cooperative positioning exhibits the worst possible trade-off among the considered strategies.

Possible avenues of future research includes the extension of the scaling laws to non-clique networks, to different MAC protocols and measurement aggregation techniques, and to network problems outside of positioning.

Appendix A
Structure of the Inverse Covariance Matrix

We consider a network with agents and anchors and focus on a particular agent i with a neighbor j and a collection of $U-1$ eavesdroppers $k\in\mathcal{N}_i\cap\mathcal{N}_j$. The resulting measurement is $\mathbf{z}_{ij}$, as defined in Section II.B. The corresponding covariance matrix $\mathbf{C}_{ij}$ for constant ranging error variance is given by

$$\mathbf{C}_{ij}=\sigma^2\begin{bmatrix}1/2 & 1/2 & 1/2 & \cdots & 1/2\\ 1/2 & 3 & 1 & \cdots & 1\\ 1/2 & 1 & 3 & \cdots & 1\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1/2 & 1 & 1 & \cdots & 3\end{bmatrix}. \tag{11}$$

It is readily verified that the inverse is given by

$$\mathbf{C}_{ij}^{-1}=\frac{1}{\sigma^2}\begin{bmatrix}\alpha & \gamma & \gamma & \cdots & \gamma\\ \gamma & \beta & \gamma & \cdots & \gamma\\ \gamma & \gamma & \beta & \cdots & \gamma\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ \gamma & \gamma & \gamma & \cdots & \beta\end{bmatrix}, \tag{12}$$

where

$$\alpha = 2\left(1+\frac{U-1}{U+3}\right) \tag{13}$$

$$\beta = \frac{U+2}{2(U+3)} \tag{14}$$

$$\gamma = \frac{-2}{(U+3)} \tag{15}$$

$$\eta = \frac{-1}{2(U+3)}. \tag{16}$$

In the particular case where all anchors (except anchor j) eavesdrop, $U=M$, and in the particular case where all nodes (except node j) eavesdrop, $U=M+N-1$. In the special case when there are no eavesdroppers, $\mathbf{C}_{ij}^{-1}$ reverts to the scalar value 2 (i.e., $\beta=\gamma=\eta=0$, and $\alpha=2$).

Appendix B
General Form of the Classical FIM

In this section we introduce the most generalized form of the classical FIM, denoted here by $\mathbf{J}$ (not to be confused with the Bayesian FIM in (9)). For convenience of the notation, we will assume that anchors can also initiate TW-TOA procedures with other nodes. We also omit the ranging variance σ^2, with the understanding that the FIM is to be multiplied by $1/\sigma^2$. Let $\mathcal{S}_R$ be the set of all nodes with which agents can range and $\mathcal{S}_E$ the set of nodes that perform eavesdropping. We further introduce $[\gamma_*, \beta_*, \eta_*]$, which are set to zero when only anchors eavesdrop, and set to $[\gamma, \beta, \eta]$ when both anchors and agents eavesdrop. When no-one eavesdrops, $[\gamma_*, \beta_*, \eta_*] = [\gamma, \beta, \eta] = [0, 0, 0]$. The diagonal block-elements of the FIM turn out to be given by

$$[\mathbf{J}]_{kk}$$

$$= 2\alpha \sum_{j \in \mathcal{S}_R \backslash \{k\}} \left[\left(\frac{\partial \mu_{kj}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu_{kj}}{\partial \mathbf{x}_k} \right)^{\mathrm{T}} \right] \tag{17a}$$

$$+ 4\gamma \sum_{j \in \mathcal{S}_R \backslash \{k\}} \sum_{n \in \mathcal{S}_E \backslash \{j,k\}} \left[\left(\frac{\partial \mu_{kj}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^n_{kj}}{\partial \mathbf{x}_k} \right)^{\mathrm{T}} \right] \tag{17b}$$

$$+ 2\eta \sum_{j \in \mathcal{S}_R \backslash \{k\}} \sum_{n \in \mathcal{S}_E \backslash \{j,k\}} \sum_{m \in \mathcal{S}_E \backslash \{j,k,n\}} \left[\left(\frac{\partial \mu^m_{kj}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^n_{kj}}{\partial \mathbf{x}_k} \right)^{\mathrm{T}} \right] \tag{17c}$$

$$+ 2\beta_* \sum_{j \in \mathcal{S}_R \backslash \{k\}} \sum_{m \in \mathcal{S}_R \backslash \{j,k\}} \left[\left(\frac{\partial \mu^k_{mj}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^k_{mj}}{\partial \mathbf{x}_k} \right)^{\mathrm{T}} \right] \tag{17d}$$

$$+ 2\beta \sum_{j \in \mathcal{S}_R \backslash \{k\}} \sum_{n \in \mathcal{S}_E \backslash \{j,k\}} \left[\left(\frac{\partial \mu^n_{kj}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^n_{kj}}{\partial \mathbf{x}_k} \right)^{\mathrm{T}} \right]. \tag{17e}$$

Each term has a corresponding interpretation: for example (17b) corresponds to the information regarding the position of agent k, due to correlation between the TW-TOA measurement from the ranging transaction initiated by agent k to node j, and the eavesdropping measurement by node n with respect to that same ranging transaction. The factor 4 in front allows us to capture all possible combinations.

For $k \neq l$, we have even more combinations:

$$[\mathbf{J}]_{kl}$$

$$= 2\alpha \left(\frac{\partial \mu_{kl}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu_{kl}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} \tag{18a}$$

$$+ 2\gamma \sum_{n \in \mathcal{S}_E} \left(\frac{\partial \mu_{kl}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^n_{kl}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} + \left(\frac{\partial \mu^n_{kl}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu_{kl}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} \tag{18b}$$

$$+ \gamma_* \sum_{n \in \mathcal{S}_R} \left(\frac{\partial \mu_{kn}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^l_{kn}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} + \left(\frac{\partial \mu^k_{ln}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu_{ln}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} \tag{18c}$$

$$+ \beta \sum_{m \in \mathcal{S}_E} \left(\frac{\partial \mu^m_{kl}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^m_{kl}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} + \left(\frac{\partial \mu^m_{lk}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^m_{lk}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} \tag{18d}$$

$$+ \beta_* \sum_{n \in \mathcal{S}_R} \left(\frac{\partial \mu^l_{kn}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^l_{kn}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} + \left(\frac{\partial \mu^k_{ln}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^k_{ln}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} \tag{18e}$$

$$+ \beta_* \sum_{n \in \mathcal{S}_R} \left(\frac{\partial \mu^l_{nk}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^l_{nk}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} + \left(\frac{\partial \mu^k_{nl}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^k_{nl}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} \tag{18f}$$

$$+ \eta \sum_{m \in \mathcal{S}_E} \sum_{n \in \mathcal{S}_E} \left(\frac{\partial \mu^n_{kl}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^m_{kl}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} + \left(\frac{\partial \mu^n_{lk}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^m_{lk}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} \tag{18g}$$

$$+ \eta_* \sum_{m \in \mathcal{S}_E} \sum_{n \in \mathcal{S}_R} \left(\frac{\partial \mu^m_{kn}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^l_{kn}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} + \left(\frac{\partial \mu^m_{nk}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^l_{nk}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} \tag{18h}$$

$$+ \eta_* \sum_{m \in \mathcal{S}_E} \sum_{n \in \mathcal{S}_R} \left(\frac{\partial \mu^k_{ln}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^m_{ln}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} + \left(\frac{\partial \mu^k_{nl}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^m_{nl}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}} \tag{18i}$$

$$+ \eta_* \sum_{n \in \mathcal{S}_R} \sum_{m \in \mathcal{S}_R} \left(\frac{\partial \mu^k_{nm}}{\partial \mathbf{x}_k} \right) \left(\frac{\partial \mu^l_{nm}}{\partial \mathbf{x}_l} \right)^{\mathrm{T}}. \tag{18j}$$

Denoting the cardinality of $\mathcal{S}_R$ (resp. $\mathcal{S}_E$) by N_R (resp. N_E), and assuming the nodes are dropped uniformly around agents k and l, we can use the expression for the partial derivatives from Section III.B to determine the scaling of each term in (17) and (18). The scaling of each term is show in Table II, where $\mathbf{I}_2$ stands for the 2×2 identity matrix, and where

$$\mathbf{A}(\phi_{kl}) = \begin{bmatrix} \cos^2 \phi_{kl} & \cos \phi_{kl} \sin \phi_{kl} \\ \sin \phi_{kl} \cos \phi_{kl} & \sin^2 \phi_{kl} \end{bmatrix}. \tag{19}$$

We recall that $\alpha, \gamma, \eta, \beta_*, \beta$ take on values that depend on N_E and N_R, depending on the specific scenario.

Appendix C
Proof of the No Case

A. PEB

Consider adding M anchors and N agents uniformly distributed over a fixed two-dimensional area. Moreover, no prior information is available for the agents' positions. We will determine the scaling of the equivalent Fisher information matrix (EFIM) $\mathbf{J}^{\mathrm{E}}_{\mathrm{No}}$ for agent k. Since the TW-TOA transactions are mutually independent, the FIM is a block diagonal matrix of the form $\mathbf{J}_{\mathrm{No}} = \mathrm{diag}[\mathbf{J}_{\mathrm{No},k}, \mathbf{J}_{\mathrm{No},2}, \ldots, \mathbf{J}_{\mathrm{No},N}]$. Given the definition of the EFIM from Section III.A, for this specific case we have that $\mathbf{J}^{\mathrm{E}}_{\mathrm{No},k} = \mathbf{J}_{\mathrm{No},k}$. From the generalized expression in (17), we can construct $\mathbf{J}^{\mathrm{E}}_{\mathrm{No}}$ by considering only the term (17a) , with $\mathcal{S}_R = \mathcal{S}_{\text{anchors}}$, $N_R = M$. Therefore, from Table II the EFIM scales as $\mathbf{J}^{\mathrm{E}}_{\mathrm{No},k} = \mathbf{J}_{\mathrm{No},k} \to 2\alpha N_R \mathbf{I}_2 = 4M\mathbf{I}_2$. Finally, following the definition of the PEB, we find that for large M $\mathcal{P}_{\mathrm{No}} \to \sqrt{1/(4M)}$. Hence, $\mathcal{P}_{\mathrm{No}} \in \mathcal{O}\left(1/\sqrt{M}\right)$.

Term	Scaling
(17a) in $[\mathbf{J}]_{kk}$	$2\alpha N_R \mathbf{I}_2$
(17b) in $[\mathbf{J}]_{kk}$	$2\gamma N_E N_R \mathbf{I}_2$
(17c) in $[\mathbf{J}]_{kk}$	$\eta N_R N_E^2 \mathbf{I}_2$
(17d) in $[\mathbf{J}]_{kk}$	$2\beta_* N_R^2 \mathbf{I}_2$
(17e) in $[\mathbf{J}]_{kk}$	$2\beta N_R N_E \mathbf{I}_2$
(18a) in $[\mathbf{J}]_{kl}$	$-2\alpha \mathbf{A}(\phi_{kl})$
(18b) in $[\mathbf{J}]_{kl}$	$-4\gamma N_E \mathbf{A}(\phi_{kl})$
(18c), (18h), (18i), (18j) in $[\mathbf{J}]_{kl}$	$\mathbf{0}$
(18d) in $[\mathbf{J}]_{kl}$	$-2\beta N_E \mathbf{A}(\phi_{kl})$
(18e) in $[\mathbf{J}]_{kl}$	$-2\beta_* N_R \mathbf{A}(\phi_{kl})$
(18f) in $[\mathbf{J}]_{kl}$	$-2\beta_* N_R \mathbf{A}(\phi_{kl})$
(18g) in $[\mathbf{J}]_{kl}$	$-2\eta N_E^2 \mathbf{A}(\phi_{kl})$

Table II
SCALING OF THE DIFFERENT TERMS IN $[\mathbf{J}]_{kk}$ AND $[\mathbf{J}]_{kl}$.

B. Minimum MAC Delay

For a clique network with N agents, each agent must range with M anchors. Each transaction requires a separate TDMA time slot. Thus, all agent-to-anchor links need to be scheduled within the network, therefore the minimum MAC delay is exactly $\mathcal{M}_{\text{No}} \in \mathcal{O}(MN)$.

APPENDIX D
PROOF OF THE CO CASE

A. PEB

When the N agents cooperate, the FIM $\mathbf{J}_{\text{Co}}$ comprises 2×2 block matrices. From the generalized expressions in equations (17) and (18), the diagonal blocks $(k = l)$ $[\mathbf{J}_{\text{Co}}]_{kk}$ are computed by considering only the term (17a) while the non-diagonal blocks $(k \neq l)$ consist of the term (18a) with $\mathcal{S}_R = \mathcal{S}_{\text{anchors}} \cup \mathcal{S}_{\text{agents}}$, with $\alpha = 2$. Hence, following Table II, since $N_R = M + N - 1$ and $\alpha = 2$,

$$[\mathbf{J}_{\text{Co}}]_{kl} \to \begin{cases} 4(M+N)\mathbf{I}_2 & k = l \\ -4\mathbf{A}(\phi_{kl}) & k \neq l. \end{cases} \tag{20}$$

The EFIM itself is hard to compute, so we will follow the procedure from [13] and determine matrices $\mathbf{J}^{\text{L}}_{\text{Co}}$ and $\mathbf{J}^{\text{U}}_{\text{Co}}$ that satisfy $\mathbf{J}^{\text{L}}_{\text{Co}} \preceq \mathbf{J}_{\text{Co}} \preceq \mathbf{J}^{\text{U}}_{\text{Co}}$, based on which we can determine an EFIM in closed form. Without loss of generality, we focus on agent $k = 1$ and analyze the corresponding EFIM lower and upper bounds.

EFIM Lower Bound: We construct $\mathbf{J}^{\text{L}}_{\text{Co}}$ from $\mathbf{J}_{\text{Co}}$ by removing all cooperation information except where agent 1 is involved in a ranging transaction, so that the diagonal elements of the FIM are given by $\left[\mathbf{J}^{\text{L}}_{\text{Co}}\right]_{11} = 4(M+N)\mathbf{I}_2$ and, for $k \neq 1$, $\left[\mathbf{J}^{\text{L}}_{\text{Co}}\right]_{kk} = 4(M+1)\mathbf{I}_2$. The off-diagonal elements are

$$\left[\mathbf{J}^{\text{L}}_{\text{Co}}\right]_{kl} = \begin{cases} -4\mathbf{A}(\phi_{kl}) & k = 1, l > 1 \\ -4\mathbf{A}(\phi_{kl}) & l = 1, k > 1 \\ \mathbf{0} & k \neq l, k, l > 1. \end{cases} \tag{21}$$

Since information is removed, it follows immediately that $\mathbf{J}^{\text{L}}_{\text{Co}} \preceq \mathbf{J}_{\text{Co}}$. Using Schur's complement, the lower bound on the EFIM of agent 1 is now given by

$$\mathbf{J}^{\text{L,E}}_{\text{Co},1} = \left[\mathbf{J}^{\text{L}}_{\text{Co}}\right]_{11} - \sum_{k \in \mathcal{S}_{\text{agents}} \setminus \{1\}} \left[\mathbf{J}^{\text{L}}_{\text{Co}}\right]_{1k} \left[\mathbf{J}^{\text{L}}_{\text{Co}}\right]^{-1}_{kk} \left[\mathbf{J}^{\text{L}}_{\text{Co}}\right]^{\text{T}}_{1k}. \tag{22}$$

The second term scales as

$$\sum_{k \in \mathcal{S}_{\text{agents}} \setminus \{1\}} \left[\mathbf{J}^{\text{L}}_{\text{Co}}\right]_{1k} \left[\mathbf{J}^{\text{L}}_{\text{Co}}\right]^{-1}_{kk} \left[\mathbf{J}^{\text{L}}_{\text{Co}}\right]^{\text{T}}_{1k} \to \frac{N}{M}\mathbf{G}, \tag{23}$$

where we have introduced

$$\mathbf{G} = \begin{bmatrix} 3/8 & 1/8 \\ 1/8 & 3/8 \end{bmatrix}. \tag{24}$$

Hence, for sufficiently large M and N, $\mathbf{J}^{\text{L,E}}_{\text{Co},1} \to 4(M+N)\mathbf{I}_2$.

EFIM Upper Bound: The EFIM upper bound $\mathbf{J}^{\text{U}}_{\text{Co}}$ is constructed from $\mathbf{J}_{\text{Co}}$ by setting the non-diagonal block elements equal to zero, i.e., $\left[\mathbf{J}^{\text{L}}_{\text{Co}}\right]_{kl} = 0$ for $k \neq l$. The EFIM for agent 1 is now $\mathbf{J}^{\text{U,E}}_{\text{Co},1} = [\mathbf{J}_{\text{Co}}]_{11} = 4(M+N)\mathbf{I}_2$.

Final Scaling: Since the EFIM scales at least as fast as $4(M+N)\mathbf{I}_2$ and at most as fast as $4(M+N)\mathbf{I}_2$, we conclude that $\mathbf{J}^{\text{E}}_{\text{Co},1} \to 4(M+N)\mathbf{I}_2$. According to the definition of the PEB, we finally find that

$$\mathcal{P}_{\text{Co}} \to \sqrt{\frac{1}{4(M+N)}}, \tag{25}$$

and thus $\mathcal{P}_{\text{Co}} \in \mathcal{O}\left(1/\sqrt{M+N}\right)$.

B. Minimum MAC Delay

For a cooperative clique network each of the N agents performs a TW-TOA transaction with every one of the remaining $N+M-1$ nodes. Each transaction requires a different TDMA slot, so the total MAC delay scales as $\mathcal{M}_{\text{Co}} \in \mathcal{O}(NM+N^2)$.

APPENDIX E
PROOF OF THE NO-EA CASE

A. PEB

Consider the addition of M anchors and N agents uniformly distributed over a fixed two-dimensional area. The FIM $\mathbf{J}_{\text{No-Ea}}$ in the case where nodes range only with anchors and only anchors eavesdrop is a block-diagonal matrix of the form $\mathbf{J}_{\text{No-Ea}} = \text{diag}[\mathbf{J}_{\text{No-Ea},1}, \ldots, \mathbf{J}_{\text{No-Ea},N}]$. Given the definition of the EFIM from Section III.A, for this case we have that $\mathbf{J}^{\text{E}}_{\text{No-Ea},k} = \mathbf{J}_{\text{No-Ea},k}$. From the generalized expression (17) we can construct $\mathbf{J}^{\text{E}}_{\text{No-Ea},k}$ by considering the summation of the terms (17a), (17b), (17c), and (17e), with $U = M$. Moreover, $\mathcal{S}_R = \mathcal{S}_E = \mathcal{S}_{\text{anchors}}$, so $N_R = N_E = M$. Substituting the appropriate values of α, β, γ, and η from (13)–(16), we easily find that the dominant term in $\mathbf{J}_{\text{No-Ea},k}$ scales as $\mathbf{J}_{\text{No-Ea},k} \to M^2/2\mathbf{I}_2$. From the definition of the PEB it follows that$\mathcal{P}_{\text{No-Ea}} \to \sqrt{2/M^2}$, and thus $\mathcal{P}_{\text{No-Ea}} \in \mathcal{O}(1/M)$.

B. Minimum MAC Delay

Since the eavesdropping measurements require no additional TDMA slots, the MAC delay scales exactly as $\mathcal{M}_{\mathrm{No-Ea}} \in \mathcal{O}(MN)$.

Appendix F
Proof of the No-E Case

A. PEB

Consider M anchors and N agents uniformly distributed over a fixed two-dimensional area where nodes range only with anchors, but all nodes can eavesdrop. The FIM $\mathbf{J}_{\mathrm{No-E}}$ consists of 2×2 block matrices, with $[\mathbf{J}_{\mathrm{No-E}}]_{kk}$ given by all terms in (17) and $[\mathbf{J}_{\mathrm{No-E}}]_{kl}$ given by (18c), (18e), (18f), and (18h)–(18j) from (18). Now, $\mathcal{S}_R = \mathcal{S}_{\mathrm{anchors}}$ and $\mathcal{S}_E = \mathcal{S}_{\mathrm{anchors}} \cup \mathcal{S}_{\mathrm{agents}}$, so we use $U = M + N - 1$ to determine the constants in (13)–(16). Hence

$$[\mathbf{J}_{\mathrm{No-E}}]_{kl} \to \begin{cases} \frac{3}{2}M^2\mathbf{I}_2 + \frac{1}{2}MN\mathbf{I}_2 & k = l \\ -2M\mathbf{A}(\phi_{kl}) & k \neq l. \end{cases} \tag{26}$$

Similarly to Appendix D, exact computation of the EFIM is difficult, so we will construct upper and lower bounds on the FIM $\mathbf{J}^{\mathrm{L}}_{\mathrm{No-E}} \preceq \mathbf{J}_{\mathrm{No-E}} \preceq \mathbf{J}^{\mathrm{U}}_{\mathrm{No-E}}$. Without loss of generality, we focus on agent $k = 1$ to analyze the corresponding EFIM lower and upper bounds.

EFIM Lower Bound: The FIM lower bound $\mathbf{J}^{\mathrm{L}}_{\mathrm{No-E}}$ can be constructed from $\mathbf{J}_{\mathrm{No-E}}$ by only allowing eavesdropping on ranging transactions involving agent 1 and only letting agent 1 eavesdrop. Inspection of the terms in (17)–(18) leads to a FIM of the form

$$\left[\mathbf{J}^{\mathrm{L}}_{\mathrm{No-E}}\right]_{kl} \to \begin{cases} \frac{3}{2}M^2\mathbf{I}_2 + \frac{1}{2}MN\mathbf{I}_2 & k = l = 1 \\ \frac{M^2}{2}\mathbf{I}_2 & k = l > 1 \\ -2M\mathbf{A}(\phi_{kl}) & k = 1, l > 1 \\ -2M\mathbf{A}(\phi_{kl}) & l = 1, k > 1 \\ \mathbf{0} & k \neq l, k, l > 1. \end{cases} \tag{27}$$

The EFIM then becomes

$$\begin{aligned} &\mathbf{J}^{\mathrm{L,E}}_{\mathrm{No-E},1} \\ &= [\mathbf{J}^{\mathrm{L}}_{\mathrm{No-E}}]_{11} - \sum_{k \in \mathcal{S}_{\mathrm{agents}\setminus\{1\}}} [\mathbf{J}^{\mathrm{L}}_{\mathrm{No-E}}]_{1k} [\mathbf{J}^{\mathrm{L}}_{\mathrm{No-E}}]^{-1}_{kk} [\mathbf{J}^{\mathrm{L}}_{\mathrm{No-E}}]^{\mathrm{T}}_{1k}. \end{aligned} \tag{28}$$

The second term is given by $8\sum_{k \in \mathcal{S}_{\mathrm{agents}\setminus\{1\}}} \mathbf{A}(\phi_{k1})\mathbf{A}^{\mathrm{T}}(\phi_{k1}) \to 4N\mathbf{G}$, where $\mathbf{G}$ was introduced in (24). Retaining only the dominant terms, $\mathbf{J}^{\mathrm{L,E}}_{\mathrm{No-E},1} \to 3/2M^2\mathbf{I}_2 + 1/2MN\mathbf{I}_2$.

EFIM Upper Bound: The EFIM upper bound is constructed from $\mathbf{J}_{\mathrm{No-E}}$ by setting the non-block diagonal elements to zero, i.e., $\left[\mathbf{J}^{\mathrm{U}}_{\mathrm{No-E}}\right]_{kl} = 0$ for $k \neq l$. The upper bound of the EFIM for agent 1 is given by $\mathbf{J}^{\mathrm{U,E}}_{\mathrm{No-E},1} \to \frac{3}{2}M^2\mathbf{I}_2 + \frac{1}{2}MN\mathbf{I}_2$.

Final Scaling: Due to the fact that the lower and upper bounds on the EFIM have the same scaling, we conclude that the EFIM itself must have the same scaling. Finally, following the definition of the PEB find that

$$\mathcal{P}_{\mathrm{No-E}} \in \mathcal{O}\left(\sqrt{\frac{1}{3M^2 + MN}}\right). \tag{29}$$

B. Minimum MAC Delay

In our model, anchors can range with agents and other anchors, while the agents eavesdrop. The total number of TDMA slots thus scales as $\mathcal{M}_{\mathrm{No-E}} \in \mathcal{O}(MN + M^2)$.

Appendix G
Proof of the Co-Ea Case

A. PEB

Consider M anchors and N agents uniformly distributed over a fixed two-dimensional area, where nodes range with all nodes but only anchors can eavesdrop. The FIM $\mathbf{J}_{\mathrm{Co-Ea}}$ consists of 2×2 block matrices. Starting from the generalized expressions, the diagonal blocks $(k = l)$ $[\mathbf{J}_{\mathrm{Co-Ea}}]_{kk}$ are computed by including terms (17a)–(17c), and (17e) from (17) and the non-diagonal blocks by summing up terms (18a), (18b), (18d), and (18g) from (18). Furthermore, $\mathcal{S}_R = \mathcal{S}_{\mathrm{anchors}} \cup \mathcal{S}_{\mathrm{agents}}$, $\mathcal{S}_E = \mathcal{S}_{\mathrm{anchors}}$, and $U = M$ for the constants in (13)–(16). Hence

$$[\mathbf{J}_{\mathrm{Co-Ea}}]_{kl} \to \begin{cases} \frac{1}{2}M^2\mathbf{I}_2 + \frac{1}{2}MN\mathbf{I}_2 & k = l \\ 4\mathbf{A}(\phi_{kl}) & k \neq l. \end{cases} \tag{30}$$

In this case it is not possible to remove measurements to determine a lower bound on the FIM. Instead, we note that that (30) behaves similarly to (20): the off-diagonal blocks do not scale with M or N, while the diagonal blocks scale at least linear in M and N. Hence, for large M and N, the diagonal blocks will determine the scaling, and $\mathbf{J}^{\mathrm{E}}_{\mathrm{Co-Ea}} \to \frac{1}{2}M^2\mathbf{I}_2 + \frac{1}{2}MN\mathbf{I}_2$, so that $\mathcal{P}_{\mathrm{No-E}} \in \mathcal{O}\left(\sqrt{1/(M^2 + MN)}\right)$.

B. Minimum MAC Delay

In this case ranging between anchors is not useful, since agents are not allowed to eavesdrop. Hence, only ranging slots between anchors and agents, and between agents are required. This leads to $\mathcal{M}_{\mathrm{Co-Ea}} \in \mathcal{O}(NM + N^2)$.

Appendix H
Proof of the Co-E Case

Consider M anchors and N agents uniformly distributed over a fixed two-dimensional area where nodes range with all nodes, and all other nodes can eavesdrop. The FIM $\mathbf{J}_{\mathrm{Co-E}}$ consists of 2×2 block matrices positioned in row k and column k. From the generalized expressions the diagonal blocks $(k = l)$ $[\mathbf{J}_{\mathrm{Co-E}}]_{kk}$ are computed by including all terms in (17) and the non-diagonal blocks by including all terms in (18). Furthermore, $\mathcal{S}_R = \mathcal{S}_{\mathrm{anchors}} \cup \mathcal{S}_{\mathrm{agents}}$ and $\mathcal{S}_E = \mathcal{S}_{\mathrm{anchors}} \cup \mathcal{S}_{\mathrm{agents}}$, and $U = M + N - 1$ for the constants in (13)–(16). This leads to

$$[\mathbf{J}_{\mathrm{Co-E}}]_{kl} \to \begin{cases} 2(M+N)^2\mathbf{I}_2 & k = l \\ -2(M+N)\mathbf{A}(\phi_{kl}) & k \neq l. \end{cases} \tag{31}$$

Comparing (31) with (26), we observe that in both cases the diagonal blocks scale quadratically in M and N, while the off-diagonal blocks only scale linearly. Hence, $\mathbf{J}^{\mathrm{E}}_{\mathrm{Co-E}} \to 2(M + N)^2\mathbf{I}_2$, so that $\mathcal{P}_{\mathrm{Co-E}} \in \mathcal{O}(1/(M + N))$.

A. Minimum MAC Delay

In this case, inter-anchor ranging is useful, since agents can eavesdrop. Thus, there will be a ranging transaction between every pair of nodes, leading to $\mathcal{M}_{\mathrm{Co-E}} \in \mathcal{O}((M+N)^2)$.

References

[1] G. E. Garcia, L. S. Muppirisetty, and H. Wymeersch, "On trade-off between accuracy and delay in cooperative UWB navigation," *IEEE Wireless Communications and Networking Conference*, pp. 1603–1608, 2013.

[2] ——, "On the trade-off between accuracy and delay in UWB navigation," *IEEE Communications Letters*, vol. 17, pp. 39–42, 2012.

[3] K. Pahlavan and X. Li, "Indoor geolocation science and technology," *IEEE Communications Magazine*, vol. 40, pp. 112–118, 2002.

[4] R. Fontana, E. Richley, and J. Barney, "Commercialization of an ultra wideband precision asset location system," in *IEEE Conference on Ultra Wideband Systems and Technologies*, 2003, pp. 369–373.

[5] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 88–97, 2002.

[6] T. Budinger, "Biomonitoring with wireless communications," *Annual Review of Biomedical Engineering*, vol. 5, pp. 383–412, 2003.

[7] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero III, R. L. Moses, and N. S. Correal, "Locating the nodes," *IEEE Signal Processing Magazine*, vol. 22, pp. 54–59, 2005.

[8] A. F. Molisch, P. Orlik, Z. Sahinoglu, and J. Zhang, "UWB-based sensor networks and the IEEE 802.15.4a standard - a tutorial," *International Conference on Communication and Networking in China*, 2006.

[9] W. Hirt, "Ultra-wideband radio technology: overview and future research," *Computer Communications*, vol. 26, pp. 46–52, 2003.

[10] S. Gezici, Z. Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, V. H. Poor, and Z. Sahinoglu, "Localization via ultra-wideband radios," *IEEE Signal Processing Magazine*, pp. 70–84, 2005.

[11] Y. Shen and M. Z. Win, "Fundamental limits of wideband localization - part I: A general framework," *IEEE Transactions on Information Theory*, vol. 56, pp. 4956–4980, 2010.

[12] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proceedings of the IEEE*, vol. 97, pp. 427–450, 2009.

[13] Y. Shen, H. Wymeersch, and M. Z. Win, "Fundamental limits of wideband localization - part II: Cooperative networks," *IEEE Transactions on Information Theory*, vol. 56, pp. 4981–5000, 2010.

[14] M. R. Gholami, S. Gezici, and E. G. Ström, "Improved position estimation using hybrid TW-TOA and TDOA in cooperative networks," *IEEE Transactions on Signal Processing*, vol. 60, pp. 3770–3785, 2012.

[15] F. Sottile, A. Vesco, R. Scopigno, and M. Spirito, "MAC layer impact on the performance of real-time cooperative positioning," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2012, pp. 1858–1863.

[16] C. Lindberg, L. S. Muppirisetty, K.-M. Dahlén, V. Savic, and H. Wymeersch, "MAC delay in belief concensus for distributed tracking," in *10th Workshop on Positioning, Navigation and Communication*, 2013.

[17] P. Leone and E. M. Schiller, "Self-stabilizing TDMA algorithms for dynamic wireless ad-hoc networks," *International Journal of Distributed Sensor Networks*, vol. 2013, p. 17.

[18] T. Wang, Y. Shen, S. Mazuelas, and M. Z. Win, "Distributed scheduling for cooperative localization based on information evolution," *IEEE International Conference on Communications*, 2012.

[19] D. Satyam, D. Zachariah, A. De Angelis, and P. Handel, "Cooperative decentralized localization using scheduled wireless transmissions," *IEEE Communications Letters*, vol. 17, pp. 1240–1243, 2013.

[20] M. Rengasamy, E. Dutkiewicz, and M. Hedley, "MAC design and analysis for wireless sensor networks with co-operative localisation," *International Symposium on Communications and Information Technologies*, 2007.

[21] I. Bucaille, A. Tonnere, L. Ouvry, and B. Denis, "MAC layer design for UWB LDR systems: PULSERS proposal," *4th Workshop on Positioning, Navigation and Communication*, 2007.

[22] B. Denis, M. Maman, and L. Ouvry, "On the scheduling of ranging and distributed positioning updates in cooperative IR-UWB networks," *International Conference on UWB*, pp. 370–375, 2009.

[23] K. Das and H. Wymeersch, "Censored cooperative positioning for dense wireless networks," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications Workshops (PIMRC Workshops)*, 2010, pp. 262–266.

[24] ——, "Censoring for Bayesian cooperative positioning in dense wireless networks," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 30, no. 9, pp. 1835–1842, 2012.

[25] "P400 data sheet," Time Domain Corp., Huntsville, AL, USA.

[26] C. Pedersen, T. Pedersen, and B. H. Fleury, "Exploiting network topology information to mitigate ambiguities in VMP localization," *4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pp. 57–60, 2011.

[27] N. Abramson, "The throughput of packet broadcasting channels," *IEEE Transactions on Communications*, vol. com-25, no.1, pp. 117–128, 1977.

[28] D. Macagnano, G. Destino, F. Esposito, and G. Abreu, "MAC performances for localization and tracking in wireless sensor networks," *4th Workshop on Positioning, Navigation and Communication*, 2007.

[29] H. Van Trees, *Detection, Estimation and Modulation Theory*. Wiley, 1968, vol. 1.

[30] W. Wang, Y. Wang, X.-Y. Li, W.-Z. Song, and O. Frieder, "Efficient interference-aware TDMA link scheduling for static wireless networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking*, ser. MobiCom '06, 2006, pp. 262–273.

[31] P. Björklund, P. Värbrand, and D. Yuan, "A column generation method for spatial TDMA scheduling in adhoc networks," *Ad Hoc Networks*, vol. 2, no. 4, pp. 405–418, 2004.

[32] J. T. Isaacs, D. J. Klein, and J. P. Hespanha, "Optimal sensor placement for time difference of arrival localization," in *Proceedings of the 48th IEEE Conference on Decision and Control held jointly with the 28th Chinese Control Conference CDC/CCC*, 2009, pp. 7878–7884.

This page is intentionally left blank.

A.1.3 Design of a Session Management Algorithm for Solving K-Token Dissemination Problem Using Network Coding

“Design of a Session Management Algorithm for Solving K-Token Dissemination Problem Using Network Coding”, Guillermo Barredo García and Iosif Salem

This page is intentionally left blank.

Design of a Session Management Algorithm for Solving K-Token Dissemination Problem Using Network Coding

Guillermo Barredo García Iosif Salem [*]

December 4, 2013

Abstract

In this thesis report we study a k-token dissemination algorithm that makes use of *network coding*. The *k-token dissemination* problem consists of propagating a total number of k tokens to all nodes in the network. The tokens are distributed between one or more nodes before the algorithm is executed, and the final goal is that all nodes must eventually have the same set of k tokens.

Network coding is a recent technique that, implemented in a proper way, helps to save bandwidth and improves the speed of distributed computation. The network model consists of a network that can change completely from round to round, therefore the nodes do not know anything about their neighbours. Moreover, when a node broadcasts a message, it does not know which are the receivers, thus, it is not possible for the nodes to know which are the tokens that their neighbours need. By using *network coding*, the time needed to achieve the final goal (all nodes possess the same k tokens) is drastically reduced in comparison to a simple random forwarding algorithm, which, as its name infers, randomly broadcasts the tokens possessed by a node.

We introduce the *session management problem* and two algorithms to solve it. The *session management problem* consists of limiting the total number of sessions that concurrently coexist in the system. In the context of dissemination problems and network coding, we consider a session as an index according to which the information is codded by the session initiator. Since in dynamic networks nodes can crash and recover and the network can partition, and merge, we wish to allow sessions to accomplish their tasks, while limiting the amount of overall system resources in use.

We propose solutions for the session management problem, and by that facilitate the solution of the k-token dissemination problem in dynamic networks. The algorithms here proposed supply an enough definite pseudo-code, which may make life easier to the programmer when it comes to implement the algorithm using a conventional programming language. Moreover, by solving the *session management problem*, we can also solve the k-token dissemination problem in a more robust way, in the sense that the algorithms can deal with several types of failures, such as, *crashes* and *crashes-recoveries*.

1 Introduction

In existing computer networks, information is transmitted from the sender to the receiver through a set of intermediary nodes which are responsible for forwarding the data in order to deliver it to the final destination. Normally, information received by intermediary nodes is stored and forwarded, this method is known as *store-and-forward*. In general, computer networks rely on routing schemes which allow the nodes in the network to select the right destination when a packet needs to be sent or forwarded.

A recent technique, known as *Network Coding*, breaks with the traditional paradigm of routing, in the sense that the packets are no longer needed to be treated as untouchable atomic packets since *Network Coding* permits the packets to be mixed with the aim of saving bandwidth. Therefore, when using *Network Coding*, the intermediary nodes have a more important task than merely acting as switches that receive information from an input link and then relay that information to an output link or set of output links. *Network Coding* is based on that, from the information-theoretic point of view, there is no reason to not use the intermediary nodes as encoders [2].

Network Coding has had a great impact in several areas of research such as networking, coding theory, complexity theory, cryptography, etc. due to its vast application potential [6]. It has been developed in various directions, and new and different applications continue to emerge [18] [5] [19] [17].

In this report, one of the algorithms introduced by Hae-

[*]This work was partially supported by the EC, through project FP7-STREP-288195, KARYON (Kernel-based ARchitecture for safetY-critical cONtrol).

upler et al. [14] is studied. This algorithm solves the problem of k-token dissemination in dynamic networks by means of *Network Coding*. The problem is defined as follows: initially there are a total number of k tokens in the network, some/all of these tokens are held by one or more nodes. The main goal is that at the end of the execution of the algorithm all participating (correct) nodes must possess the same set of k tokens.

The network model that this algorithm is intended for, represents many modern networks where the topology of the network is changing constantly. The topology of the network may change totally from round to round, hence it offers a challenging scenario, since the nodes in the network do not know neither which are their current neighbours nor which neighbours they will have in the next round. This leads to a model where the only knowledge that nodes have is the tokens that they have received so far.

The algorithm showed by Haeupler et al. lacks of detail, since one can not actually implement the algorithm by only looking at the presented pseudo-code. Moreover, faulty nodes are not considered. For that reason, we present a *session management problem* along with two pseudo-code algorithms which give a more detailed and realistic solution that can be easy to implement. The *session manager* provides extra properties compared to the studied algorithm, such as: *self-organization and -recovery*, in order to make it more robust. The algorithms presented in this work can deal with nodes that crash and stop, crash and resume, and crash and reboot.

1.1 Related Work

As mentioned before, the algorithm presented by Haeupler et al. [14] does not deal with failures. We wanted to strengthen this part, as a consequence, our two algorithms contain a failure detection mechanism that allows the nodes in the network to sense different types of failures (i.e. crash-stop, crash-recovery and crash-reboot). The failure detector used by the algorithms makes use of counters. One of those counters is the *Incarnation number* counter, which can reach the maximum value of the datatype that defines it. This leads to the *Wrapping Around Problem*, which, along with its solution, is presented in the following sections.

The studied algorithm does not provide a solution to know when all nodes in the network posses the same knowledge. For that reason, in this document we provide two different mechanisms that allow the nodes to know when they have the same knowledge. The first mechanism is used by the nodes to check whether they have the same set of tokens or not. On the other hand, the second mechanism helps the nodes to figure out whether they posses the same knowledge about the coded tokens belonging to a session or not.

Finally, Haeupler's et al. algorithm, *Greedy-forward*, only allows one leader or identified node at a time. This identified node is the node responsible for starting sending network coding packets. Since in our system, nodes can fail, allowing a single node to send coding packets at a time can lead to a noticeable inefficient performance. Consequently, one of our presented algorithm permits to have more than one identified node in the system at a time, so, in case that the identified node, which is currently carrying out the network coding phase, crashes, another identified node can take over.

1.2 Structure of the report

The remainder of this report is organized as follows. In the second section, we show the background of this report together with our contributions. The third section contains a detailed description of the system settings is given. The next section is *Network Coding*, where a brief introduction to *network coding* and *random network coding* is presented. In addition, this third section also includes some applications that employ the *network coding* technique. In the fourth section, we define the problem that our algorithm needs to solve. Following, in the fifth section, we present basic problems that are associated to the k-token dissemination and session management problem, along with their solutions. The next two sections present, respectively, the design, explanation, lemmas, and proofs of two approaches to solve the studied problems. Finally, the last section of the report consists of the conclusions and advices for future work.

2 Background

Haeupler et al. [14] show how to make use of *network coding* in order to improve the performance of distributed computation in a dynamic network model, which is previously presented by Kuhn, Lynch and Oshman [20].

The dynamic network model has very specific and restrictive system settings which make this model representative of the highly dynamic and non-converging nature of many modern networks, e.g., Wireless Sensor Networks (WSN). These system settings consist of allowing the network to change completely in every round, but it is subject to the constrain that the network must remain always connected, i.e. the topology might change in every round but this changes can not end up with two or more sets of nodes that are not connected between them. In each (synchronize) round, each node selects a message from a pool of messages, and broadcasts it to its neighbours for that particular round. In addition, the sender of the message does not know who are going to be its recipients, and therefore, it does not know whether any of its neighbours already has the message that it is going to broadcast or not. The fact that the broadcast is "anonymous", makes this problem particularly challenging.

Along with the dynamic network model, Kuhn et al. [20] show how to solve the problem of k-token dissemination in such model. The problem consists of disseminating k tokens, in such a way that eventually all nodes in the network possess the same set of tokens. The proposed approach to solve the problem is called, *token forwarding*, and it is probably the most "natural" approach. It consists of broadcasting a token during $O(n)$ rounds, so after $O(nk)$ rounds all the tokens are disseminated. Furthermore, Kuhn et al. provide a more general lower bound $\Omega(n\ log\ k)$ that applies even if the algorithm is operated under a centralized control.

On the other hand, Haeupler et al. [14] show that this lower bound cease to hold when tokens can be broadcast together. This is done by using *network coding*, which allows to send out random linear combinations of tokens. If the size of a token is $O(log\ n)$, then in order to solve the k-token dissemination problem the algorithm will need $O(kn/log\ n)$ time to finish. This bound clearly outperforms the $O(nk)$ provided by Kuhn et al. In addition, the authors of [14] show that the greater the size of the token is, the faster can be disseminated. They state that using *network coding*, k tokens can be disseminated in $O(k(n\ log\ n)/d)$ time. Therefore, if the size of the token is equal to $O(n\ log\ n)$, the k-tokens will be disseminated in $O(k)$ time which also outperforms the lower bound $\Omega(n\ log\ k)$ associated to token-forwarding algorithms.

Both papers, [14] and [20], talk about T-stable networks, where the topology of the network does not change during a T-interval of time. If the network is stable for a certain period of time T, then it is possible to use other algorithms that improve the running time. For instance, according to [20], the complexity of the algorithms used for this kind of networks is $O(nk/T + n)$ time. In contrast, Haeupler et al. [14] show that *network coding*, instead of achieving a factor-T speed-up, can achieve a factor-T^2 speed-up. In this thesis, the described algorithm has been designed for networks that change in every round. Networks that are T-stable are outside the scope of this work. In case, the network only changes once every T rounds, then extra features should be added to the algorithm in order to make it more efficient.

2.1 K-indexing

K-indexing is one of keys to solve in an efficient way the k-token dissemination problem with *network coding*. Every coded packet has associated a coefficient vector, that eventually will be used in order to decode the k-linearly-independent coded tokens. All nodes in the network must agree on which index is associated to which token, otherwise there will be inconsistencies when doing Gaussian Elimination[1]. When talking about index, we refer to the basis vector that is associated to each token, so, for instance, the basis vector associated to the first token should be $\langle 0, ..., 0, 1\rangle$, to the second token $\langle 0, ..., 1, 0\rangle$ and so on. The size of this vector will be equal to the number of tokens that need to be coded.

Once all nodes in the network agree on what token has what index, then *network coding* can be used to disseminate such tokens. The problem is how to make that all nodes agree on the same indexes for the same nodes. Haeupler et al. show two different approaches:

1. Naive solution: All nodes give unique IDs of size $O(log\ n)$ to their tokens. Then they broadcast repeatedly the smallest $\Omega(b/log\ n)$ tokens they have heard about, where b is the size of the message. After n rounds of flooding, all nodes will have the unique IDs of the $(b/logn)$ smallest tokens, and therefore they will be able to agree on the indexes associated to these tokens. But this approach is only a $(log\,n/d)$ faster than the forwarding algorithm which does not make use of *network coding*. This is because the process of agreeing on the smallest tokens is repeated $k(log\ n/b)$ times, to this it is necessary to add the $O(n)$ time[2] that *network coding* phase takes. Consequently, the total time required to disseminate k tokens is $O(nklog\ n/b)$ or $O((log\ n/d) * (nkd/b))$.

2. Gathering tokens: A more efficient solution proposed by Haeupler et al., where instead of making all nodes to participate in the indexing problem, tokens are gathered in a single node which is the responsible for assigning the indexes to them. In order to propagate and gather the tokens, Haeupler et al. make use of a simple random forwarding algorithm, which turns out to be the same as the one used for disseminating k-tokens when *network coding* is not exploited. The authors explain in the paper, how efficient this algorithm is at the beginning, but as the time passes, nodes receive more and more packets that they have already received. The expected time

[1]A extended explanation of the reason why Gaussian Elimination is needed is given later on this report.

[2]With high probability.

to disseminate all the tokens using this technique is $O(nkd/b)$. Hence, it needs to be combined with *network coding* in order to achieve a noticeable improvement in the expected time. Haeupler et al. proposes to take advantage of the efficiency of the random forwarding algorithm in its early stage in order for a certain node in the network to gather an enough amount of tokens, and later send the gathered tokens using *network coding*.

2.2 Leader Election

When treating with dynamic networks, generally the methods and algorithms used to solve problems in conventional distributed systems are not applicable. One clear example of this is the leader election algorithm, where all correct nodes in the network must eventually agree on the same leader which is responsible of carrying out a specific task.

According to Haeupler et al.'s [14] *greedy-forward* algorithm, once a node has gathered b^2/d tokens, this node is identified as a leader, and it is the responsible for broadcasting the coded packets. The process of electing a leader will be finish within at most $O(n)$ rounds if no other node proposes itself as a leader. It can be the case, that two nodes are able to gather b^2/d tokens and thus they will have to "fight" for the leadership which will be decided with respect to, for instance, the greatest lexicographic ID. This will lead to a total of $O(2n)$ rounds (in the worst-case scenario), in order for all nodes to agree on the same leader. In fact, there is not a leader election phase as such, but once a node reaches the threshold of minimum number of tokens needed to start the *network coding* phase, it, indeed, indexes the tokens and starts sending random linear combinations of them. Therefore, as the reader may notice, one node can disrupt all the previous work done from another node with greater ID.

On the other hand, [14] does not deal with failures. What happens when the leader goes down while sending random linear combinations of tokens? How do the other nodes realize that the leader went down? What happens if an old leader after going down "wakes up" again and it turns out that there is already a node responsible for the *network coding* phase?, etc. These are open questions that will be answered throughout this report.

In addition, Haeupler et al. base the good performance of its algorithm on the synchronization feature of the network and the high probability of finishing the *network coding* phase in $O(n)$ rounds. But in a more realistic scenario, where networks are not completely synchronized and where not all nodes are able to decode all the tokens after $O(n)$ rounds, problems may arise. For example, suppose that after $O(n)$ rounds sending random linear combinations of packets, the leader stops with the *network coding* phase, but one or more nodes in the network have not received all the necessary linear independent vectors in order to decode the packets. Moreover, as stated in the paper, once the *network coding* phase has finished, all broadcast tokens are removed from consideration. Consequently, some of the k initial tokens that need to be disseminated to all tokens in the network will never be possessed by at least one node and hence, it will lead to the impossibility of solving the k-token dissemination problem.

2.2.1 Stable Leader Election

One of the first conclusions that can be derived from the text above is that once a node has already started the *network coding* phase, we do not want another node with greater ID to disrupt all the work done by such node. There are two case scenarios where this can occur. The first one is when two nodes reach the threshold of minimum number of tokens needed to start the *network coding* phase, in an interval no greater than $n-1$, where n is the total number of nodes in the network. In addition, none of the two nodes has previously received a coded packet, since once a coded packet is received, the node will continue with the *network coding* phase started by another node. The second case scenario has to do with the disconnection and re-connection of an old leader, i.e. a node that started the *network coding* phase but it disconnected before finishing. If this "old" leader re-connects when there is a current leader, and the reconnected leader has greater ID than the current one, then it will disrupt all the work done by the newer leader.

For that reason, it is necessary to introduce the notion of *stable leader election* [1]. Aguilera et al. propose three different algorithms that are intended to provide stability to the leader. A leader election algorithm is said to be stable, if once a leader is elected, it remains being the leader until it disconnects or crashes, independently of the behaviour of other nodes. The model showed in [1] consists of a distributed system which is partially synchronous, processes have a drift-free clock and there is an upper bound B on the time within a process executes a step. Furthermore, Aguilera et al. claim that the algorithms presented in the paper are self-stabilizing, a feature that our algorithms also have. Even though, stable leader election algorithm introduced in [1] seems a decent solution for the problem mentioned above, we need another algorithm that is more suitable for solving the problem here presented. In particular, a solution that could be easier to integrate with the *network coding* and random forwarding phases of the original algorithm. The idea is to use a *piggybacking* technique that allows the algorithm to work with the *network coding*/ random forwarding and leader election in a coexisting way.

2.3 Gaussian Elimination

Since, in order to encode packets during the *network coding* phase the packets are randomly linearly combined, an algorithm for solving linear equations is needed. As mentioned earlier, the packets sent during the *network coding* phase are composed by a coefficient vector and a vector with the combination of the tokens that the coefficient vector indicates. For instance, if the coefficient vector, c, is equal to $\langle 1, 0, 1 \rangle$, that means that the vector that contains the combination of tokens will have a combination of the first token and the third one. Note that if a node has only received this packet, it will not be able to decode the tokens, since it does not have enough number of *equations* to solve the system. Thus, a node will be able to retrieve useful information about one of these tokens, if more linear independent packets are received. The algorithm used to carry out this task is Gaussian Elimination algorithm.

2.4 Failure Detection

The algorithms that have been designed through the life cycle of this project make use of a failure detection technique in order to provide correct operation.

2.4.1 Failure Models

In a highly dynamic system, and in any system in general, mainly five types of failures can take place:

1. Crashes. The processor or, in this case, a node stops functioning and it never starts again.
2. Omissions. Omission errors take place when one or more actions that are carried out by a node fail.
3. Time errors. These type of failures take place when responses arrive outside the specified time interval.
4. Crashes and Recoveries. A node halts, but eventually recovers.
5. Arbitrary or Byzantine. A node may fail in an arbitrary way, including sending arbitrary data its neighbours in the network.

2.4.2 Failure Detector

Once we know which type of failures can occur in our system, it should be stated which failures our algorithm will deal with.

Looking at the condition of the nodes, the same (crash) failure is treated in two different ways:

1. Regular node. If a regular node crashes, the rest of the nodes will not notice it, since it is just a node among many, whose failure will not affect the performance of the algorithm. Note that all the tokens that the crashed node possessed before crashing and which were not possessed by any other node, will disappear forever from the system.
2. Node responsible for having started the *Network Coding Phase*. If a node that is responsible for broadcasting random linear combinations of tokens goes down before sending enough linear independent vectors in order for the rest of the nodes to be able to decode all its tokens, it must be known by the rest of the tokens in the network. Otherwise, the rest of the nodes will be waiting until they have enough packets to decode the coded tokens, but since the leader is down, these packets will never arrive and therefore, the awaiting nodes will not move forward.

Moreover, Conan et al. [7] present in their paper an architecture of local and distributed detectors for mobile networks. These detectors give the network the ability to detect failures, disconnections and partitions. Partitions are not allowed in our network model, but in case this extra feature wanted to be added, we recommend the reader to have a look at [7]. Conan et al. propose an eventually perfect unreliable partition detector that exploits information provided by a failure detector and disconnection detector in order to identify partitions. Contrary to the failure detector given on our report, where no list of processes is kept, the unreliable failure detector shown in [7], regularly provides, for each process p in the network, a list of processes suspected to be unreachable.

2.5 Self-stabilization

Both algorithms presented on this paper possess a strong property that makes them highly robust algorithms, known as *Self-stabilization*. A self-stabilizing system is a system that can automatically recover after the occurrence of transient faults.

2.5.1 History

The first time the concept of self-stabilization was presented was in 1974 by Edsger W. Dijkstra in [9]. But it was not until 1985, when Leslie Lamport [21] mentioned the fault tolerance technique proposed by Dijkstra, that the research community realized how relevant and important this new concept was. Lamport regarded Dijskstra's paper as a milestone in work on fault tolerance and a very fertile field for research. Dijsktra defines *self-stabilization* in the following way: "A system is self-stabilizing when, regardless of its initial state, it is guaranteed to arrive at a legitimate state in a finite number of steps".

2.6 Wrapping Around Problem

The two different approaches presented in this report use an *incarnation number* counter in order to differentiate between correct nodes and nodes that just joined the network or have crashed and recovered. Since the size of the data type used for the counters is limited, it is necessary to find a solution for when the bound value is increased by one.

Herman et al. [15] propose two different approaches to deal with this problem. They have a bounded global clock with domain $[0, L]$. When the event where the clock rolls over from L to zero takes place, it disrupts the converge-to-max protocol presented on the paper [15]. The two techniques showed by Herman et al. are obatined from the literature of self-stabilizing phase-clocks and they consist of:

1. Redefining comparison of clock values in the clock protocol to behave modulo $L + 1$ [8].
2. Letting the event of a clock that reached L initiate a system reset, after which all clocks begin from zero [3].

In [10], the authors also deal with the *wrapping around problem* and make use of wrap around flags that indicate when a process has wrapped its counter. Dolev et al. state that even though the size of the counter is 64-bits and therefore practically infinite, this assumption is not valid within the scope of self-stabilization. The reason why it does not hold is because a single transient-fault may lead a counter to reach the above mentioned large number at once, and therefore it will disrupt the whole self-stabilizing algorithm avoiding it to reach a safe configuration.

2.7 Our Contributions

With respect to the related work, the following points describe our contributions:

1. **K-indexing.** In both algorithms presented on this document, the technique used to gather tokens is similar to the one used in the studied algorithm. Nodes randomly forward plain-text tokens until a *session* becomes stable. Note that there are no longer leaders but sessions and session creators. The creator of a session is the responsible for carrying out the indexing.

2. **Stable Leader Election.** The way of how a node is elected to start sending random linear combinations of tokens is completely different to the approach used in the studied algorithm. Instead of directly starting with the *network coding* phase once a node gathers enough number of tokens, a node will become eligible. And, if certain requirements are fulfilled, then it will create a session. Once all nodes in the network agree on this session, the session will become stable and the creator of the session will be allowed to start with the *network coding* phase.

3. **Piggybacking.** Piggybacking is the technique used in network transmission, and more specifically, in the network layer of the OSI model. It consists of adding the acknowledgements of previous received packets to the data frame, thus instead of sending a confirmation in an individual frame, i.e. a frame containing ACK information, the acknowledgement is appended with the payload of the next packet that the receiver will sent to the emitter. This practice saves bandwidth since it requires fewer frames [24].

   This technique allows us, for instance, to agree on the node responsible for broadcasting random linear combinations of tokens, while plain-text and coded packets are sent, without interrupting neither the network coding phase nor the random forwarding phase. The main advantage of using piggybacking is that the total number of packets used is reduced and hence, the efficiency of the algorithm is incremented.

4. **Gaussian Elimination.** Considering that the base q used to encode the packets is equal to 2, the general Gaussian Elimination algorithm can be simplified. In addition, a minor modification has been carried out in the algorithm in order for the nodes that have the same set of linear independent packets to detect that they have the same information.

The algorithm is well known and therefore it will not be described in depth on this report. Basically, it consists of three main operations:

I Find the lexicographically lowest row that has a pivot corresponding to the iteration of the algorithm. That means, in the first iteration of the algorithm, we look for a pivot in the first column, in the second iteration we look for a pivot in the second column and so forth. The general algorithm does not look for the lexicographically lowest row, instead, it uses the first row found that has a pivot. The reason why this lexicographical order has been implemented is for the nodes that have received the same set of coded packets to have the same matrix independently of the order in which they have received those packets. This will be explained with more detail later on the report.

II Once the row with the pivot has been found, it must be placed in the correct row. If it turns out that the selected row is already placed in the correct row, nothing is done, otherwise, the positions of two rows are swapped.

III XOR all the rows that have the same leading coefficient as the selected row, with the selected row itself. Hence, all rows that had the same leading coefficient but were not selected, end up with these coefficients equal to zero.

5. **Failure Detector.** The algorithms that will be shown and described later on this report, implement a detection failure technique that allows the nodes in the network to notice if the node responsible for sending coded packets has crashed or not. A proper explanation of how the failure detector is implemented will be given later on this document.

   On the other hand, nodes that crash and resume or reboot later are also tolerated. It might be the case, that a node, independently of its condition, crashes and eventually recovers being able to retrieve complete, partial or none of the tokens that has been sent while it was absent. It is assumed, that no node that has gone down will recover after a certain number of rounds, i.e. if a node crashes it has a certain time to recover, if this time is exceeded then it is assumed that this node will remain in a crashed state "forever".

   Our failure detector has two properties: Completeness and Accuracy. Completeness refers to the fact that every crashed node is eventually suspected by every correct node. And accuracy refers to the fact that no correct node is ever suspected. This will hold as long as no partition occurs in the network.

6. **Wrapping Around Problem.** Redefining comparison technique, first presented in [8] will be the one used to deal with the *wrapping around problem*. Moreover, a similar technique to the system reset introduced by Arora and Gouda in [3], can be used in the algorithm for the cases where two incarnation numbers are not comparable[3].

Finally, our main contributions are the *session management problem* together with two different approaches to solve it. The *session manager* is used to solve the k-token dissemination problem providing a robust solution that is capable of detecting certain failures. In addition, both algorithms presented on this paper are self-stabilizing. A strong property that allows the system to eventually reach a legitimate state, independently of its starting state.

3 System Settings

We consider a distributed system formed by nodes or processors: p_1,p_2,...,p_N, where N is a known upper-bound on the number of nodes. Nodes may *crash-stop*[4], *crash-reboot*[5] and *crash-resume*[6], but always respecting the upper-bound N. During the execution of the algorithm, processes that have not crashed are said to be correct. All nodes in the system have unique identifiers.

The nodes in the network may send and receive messages to and from their set of neighbours. The communication between two nodes p_i and p_j, where $i \neq j$, is modelled by a FIFO queue. Whenever a processor p_i broadcasts a message m to its neighbours, m is appended to all queues of all the neighbours of p_i. Whenever there is a pending message in the queue, this message is immediately processed by the node. For each neighbour, there exists a queue.

At every round, the network topology may change, however it is assumed that it is always connected. The network topology is defined by a connected undirected graph G. For every round r, any node in $G(r)$ can reach any other node that also belongs to $G(r)$. Partitions in

[3]In this project, we assume that all incarnation numbers that exist in the system are comparable

[4]A node that crashes and stops is a node that after crashing will be never part of the network again.

[5]A node that crashes and reboots is a node that after crashing it may join again the network with an initial configuration state.

[6]A node that crashes and resumes is a node that after crashing it may join again the network with the same state as the state it had just before crashing.

the network are not allowed. It is assumed that there are no collisions in the system, hence for the sake of simplicity the nodes are connected by directed communication links. When communication takes place, nodes broadcast anonymous packets through these links to all their neighbours, where a packet sent from p_i to p_j will make use of the (directed) link l_{ij} and a packet sent form p_j to p_i will make use of the (directed) link l_{ji}. We assume that in every round all correct nodes broadcast only one packet to their neighbours, and since the network is always connected, all correct nodes expect to receive at least one packet. In addition, a packet sent by a node is received by all its correct neighbours, therefore the network is assumed to be free of packets losses. It is assumed that the local operations are insignificant and thus, they do not affect the communications in the network.

The system is *self-stabilizing*. Shlomi Dolev in his book Self-Stabilization [23] defines a self-stabilizing system as a system which can be started in any arbitrary configuration but in a finite number of steps the system should recover and exhibit a desired legal behaviour.

The desired legal behaviour is defined by a set of legal executions (LE). A set of legal executions, in turn, is defined for a particular system and a particular task. It is said that a configuration c is safe with regard to a set of legal executions of a task and an algorithm, if every fair execution of the algorithm that starts from c belongs to the set of legal executions. In other words, if every system execution that starts from c is in LE, then we say that configuration c is safe. In addition, an algorithm is *self-stabilizing* for a set of legal executions of a particular task if every fair execution of the algorithm reaches a safe configuration with relation to LE.

Another property of self-stabilizing algorithms is that they never terminate. This property can be easily identified in the code of a self-stabilizing algorithm by looking at the do forever loop. Moreover, a *self-stabilizing* algorithm does not need to be initialized, since regardless the of its initial state, it eventually reaches a correct state. Once the system has reached a correct state, it will remain in this correct state as long as no fault occurs.

Every node p_i executes an algorithm which is composed by a sequence of *steps*. A step may start several local computations, but ends with a single computation either *send* or *receive* of a packet.

4 Network Coding

In the following section, a more detailed description of *network coding* will be given, along with an introduction to some of its applications.

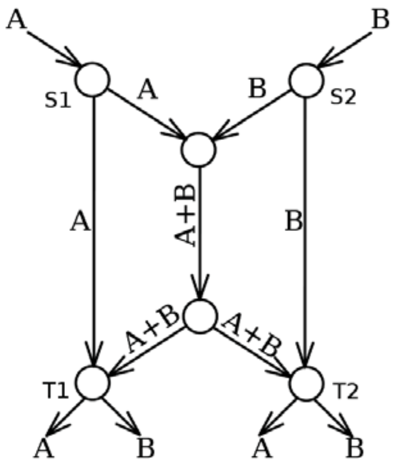


Figure 1: Butterfly network.

Network coding is a relatively new technique that can significantly improve the networks' throughput, efficiency and scalability. In the classical paradigm of routing, the packets are simply relayed when they are received, however, when using *network coding*, the nodes of a network gather a certain amount of packets and mix them for transmission.

In order to clearly show how *network coding* can outperform routing, a typical example, known as "the butterfly network" is described in the following paragraphs.

Suppose there is a total number of six nodes in a directed oriented graph, as shown in Figure 1. Where $S1$ and $S2$ are the sources of the network, i.e. the ones responsible for injecting information into the network, and $T1$ and $T2$ are the sinks. The capacity of each edge is one bit, thus the maximum data that one node can send to a neighbour in every time slot is only one bit. For this example, $A = 1$ and $B = 0$, are transmitted from $S1$ and $S2$ respectively.

If classical routing is used, after the first round one of the central nodes will receive A and B, and the exterior nodes ($T1$ and $T2$) only A or B. In the next round, the central link will be able to carry only one value or bit, that will be either A or B. In case A is transmitted the left destination ($T1$) would receive A twice, and the same will happen to the right ($T2$) destination if B is sent. It is trivial to see that there is no routing scheme that can transmit both A and B simultaneously to the two final destinations.

On the other hand, if *network coding* is used, when the first intermediary node receives A and B will use the XOR operation and generate a bit that will be equal to $A \oplus B = 1$. When $T1$ receives $A\ XOR\ B$, it will simply use XOR (i.e. $A \oplus A \oplus B$) to retrieve B and $T2$ will similarly retrieve A. It is evident that when using routing, three more bits have to be sent in order for $T1$ and $T2$ to recover A and B, whereas *network coding* only needs to send 9 bits to achieve it. Therefore, 25% of bandwidth can be saved in this basic scenario by using a simple *network coding* scheme.

4.1 Random Network Coding

When the network is dynamic or the topology of the network is not known and some information need to be broadcast to all nodes in the network, *random network coding*[16] can be used to solve the problem in an efficient way. *Random network coding* is a powerful coding scheme which provides a close to optimal throughput. When *network coding* was first stated in [2], it was presented as a technique to achieve optimality. The scenarios mentioned by Ahlswede et al. describe networks containing sources and sinks. The rest of the nodes are considered as intermediate nodes, which act according to a predefined coding scheme.

On the other hand, *random network coding* is also a type of coding scheme which, in contrast to "regular" coding schemes, behaves as a decentralized algorithm. When *random network coding* is deployed in the network, all the nodes function in a same manner. Nodes transmit random linear combinations of the packets they receive. These packets are mainly divided in two parts, one is the coefficient part and the other the coded part. The coefficients are chosen from a *Galois* field. Once a node has received enough number of packets, it can decoded them and thus, retrieve the original information generated by the sources.

4.1.1 How fast does information spread?

Haeupler, introduces in [13] a simple projection analysis technique that shows how fast information spreads when *network coding* is used and therefore, how much time is needed for all the nodes in the network to be able to decode all coded tokens.

Haeupler states that the right way to look at the spreading of information is to look at the orthogonal complement of the coefficient subspaces. In [14] and [13] a definition of knowledge is given, which says that a node *u* knows about a coefficient vector $\overrightarrow{\mu} \in F_q$ if it has received a message with a coefficient vector $\overrightarrow{\mu}'$ that it is not orthogonal to $\overrightarrow{\mu}$. Note that the bigger the q is, the more the probability to learn something new.

Right after, in [14], the Haeupler et al. give a lemma that basically shows that any node that knows about $\overrightarrow{\mu}$ will pass that knowledge to their neighbours with probability at least $1 - 1/q$. Finally, it is proved that the *network coding* algorithm with $q \geq 2$ solves the k-indexed-broadcast problem in an always connected dynamic network with probability at least $1 - q^{-n}$ in time $O(n + k)$.

4.2 Applications

Despite *network coding* is an arguably recent technique, it has already been implemented and proved that it is useful in many different areas:

1. *Avalanche*: It is a research project carried out at Microsoft, which is claimed to provide a cost effective, scalable and very fast file distribution solution compare to existing *P2P* systems[12]. The authors of the project propose a solution for the well known problem of Peer-Assisted file delivery systems, where the last "rarest pieces" of a file are harder to obtain. This, at the end, will result in slower downloads. In order to fix the problem, *network coding* is used. Instead of simply distributing the blocks of the file, peers transmit linear combinations of the blocks they already hold together with a tag that indicates the parameters used in the combination. This approach clearly solves the problem, since a peer does need to find specific pieces; any subset of encoded pieces suffices.

2. *COPE* [18]: It is an architecture for wireless mesh networks that utilizes *network coding* to increase network throughput. The routers mix the information content in the packets before forwarding them. The example given in the paper to briefly explain how COPE works consists of three computers: Alice's computer, Bob's computer and a computer that acts as a relay. Alice wants to transmit a packet to Bob and Bob wants to do the same with Alice. If *network coding* is not used, the relay will receive one packet at a time(otherwise there will be a collision) and forward it. Four transmissions will be needed in total in order for Alice and Bob to receive each others packets. However, if *network coding* is used; for instance, Alice will send first the packet and the relay will receive it, but instead of directly forward it, the relay will hold it for a certain period of time. During this period Bob will send his packet, and the central computer will only have to "XOR" the packets and forward the resulting packet. By using *network coding*, the number of transmissions is reduced from four to three. From this simple example, it can be also observed that using COPE leads to bandwidth savings.

3. *Spatial Buffer Multiplexing* [5]: It is a technique to reduce buffer utilization and delay. This approach makes use of a scheme where the intermediary nodes have no buffering capabilities for queuing transient packets (in contrast to traditional approaches). In spatial buffer multiplexing, the buffering and coding is implemented at the source and the authors of the paper claim that this will compensate for packet loss at any downstream buffer-less link.

4. *CTCP* [19]: It is a reliable transport protocol based on *network coding*. It incorporates same features of TCP such as reliability, congestion control and fairness but it additionally improves TCP's performance in lossy, interference-limited and/or dynamic networks. The authors affirm that the combination of TCP transport layer and *network coding* yields to performance gains in the presence of interference.

5. *AdapCode* [17]: It is a reliable data dissemination protocol that makes use of *network coding* in order to reduce the total amount of traffic during the process of code updates. The main idea behind *AdapCode* is that the *network coding* schemes change according to the quality of the links.

   Hou et al. state that the broadcast used when doing troubleshooting, which requires frequent upload of new code, must be fast, reliable and minimal in terms of network bandwidth consumed. *AdapCode* is intended for wireless sensor networks, it takes advantage of the fact that *network coding* reduces the total amount of traffic although increases the local computation, which is ideal for such kind of networks since the communication is slower and needs more energy compared to local computation.

   The *network coding* methodology used in [17] consists of the random combination of N coefficients and the computation of the linear combination of N packets. A node dynamically decides on N based on the number of neighbours it has. In other words, *AdapCode* adaptively decides on its coding scheme using the local knowledge of each node. In the paper, *AdapCode* is compared with *Deluge*, which is a state-of-the-art protocol used to propagate new code images, in TinyOS version 2. *Deluge* can disseminate data with 100% reliability at a speed of approximately 90 bytes per second. And according to Hou et al. the results of the comparison show that *AdapCode* uses less packets than Deluge to disseminate a image of the same size. An example of a code image of 1024 packets is given, which is sent by *AdapCode* with a reduction up to 40% of the total number of packets that are needed to disseminate the same image using *Deluge*.

   In order to achieve 100% reliability, *AdapCode* makes use of Negative-ACK. Nevertheless, it is worth to mention that, in the paper, the authors do not address temporary node failures or reboots. In addition, in contrast to the network model that this report is based on, they consider that there is a single source of data, instead of having more than one source disseminating data at the same time. The reason why Hou et al. consider only one source is because normally the scenarios that are presented in wireless sensor networks, contain a single source which is responsible for broadcasting the packets.

5 Defining the problem

The following section provides a definition of the k*-token dissemination* problem, together with some of its most relevant key points. Moreover, the core problem of this thesis is presented: a *session management problem*. Solving this problem will help us to resolve the problem here studied, the k-token dissemination problem.

5.1 K-token Dissemination: Problem description

The problem that needs to be solved is based on the dynamic network model first proposed by Kuhn et al. [20] and on the *greedy-forward* algorithm given by Haeupler et al. [14]. As explained in section 2 (*Background and Our Contributions*), the problem that needs to be solved is the k*-token dissemination problem*, where there are a total number of k tokens distributed throughout the network and all nodes need to possess them eventually. *Network coding* can be used to solve the problem in a efficient way. However, it can not be directly used. First it is necessary that a node gathers enough number of tokens in order to start sending random linear combinations of tokens.

The algorithms described in this thesis are based on the algorithm provided by [14]. However, the k-token dissemination algorithm shown by [14] lacks details and does not explain how to deal with some of the problems that raise in networks of such characteristics, such as: When do all nodes know that they have the same set of tokens?, When does the identified node, which is responsible for broadcasting up to b^2/d know that the rest of the nodes can decode all b^2/d tokens?, How do all nodes know whether the node responsible for broadcasting random linear combinations of tokens has crashed?, etc. Hence, we want our algorithm to be able to deal with the following problems:

- If the node responsible for starting sending random linear combinations of tokens crashes, all correct nodes should eventually notice it.
- All nodes must agree on the same elected node before this starts broadcasting random linear combinations of tokens.
- The system should be self-stabilizing.
- If a node that has crashed suddenly recovers, it shall not disrupt the self-stabilizing condition. In other words, the node may make the system to enter in a non-legitimate state, but after a finite number of rounds the system will exhibit a desired legal behaviour.

It is worth to mention that the algorithms presented in this thesis are not just an extension of the one given by Haeupler et al. but a realistic, fault-tolerant and self-stabilizing solution, ready to be implemented in a system whose characteristics are similar to the ones that the algorithms are intended for.

5.2 Session Management Problem

In the context of the k-token dissemination problem, a session represents a node that is able to start with the network coding phase because it has reached the minimum number of tokens needed to do so. When a node reaches such threshold, we say that this node is eligible. In the moment that a session is created, every node in the network will eventually notice this event and allocate buffer space to store the information associated to the session. One of the reasons why the sessions must be handled is that the set of eligible nodes is unknown and thus, if every eligible node creates a session, all nodes need to allocate buffer space to store the information associated to each single session. For instance, in the k-token dissemination problem each node stores all the coded packets (coefficients and coded tokens) associated to a session in order to decode the tokens. Consequently, in a system where the nodes have limited resources, we do not want all (eligible) nodes in the network, which can start with the network coding phase, to immediately create a session. Therefore, it is needed to limit the total number of sessions existing in the system at the same time to a bounded number, S, of sessions. In this document, we propose two different approaches, one for when $S = 1$ and the other one for a general S. The advantages and disadvantages of using one or another approach will be shown later on this report.

We define a *session* as an expirable lease that grants a node the permission to perform a "dissemination" task. In addition, only eligible nodes can create a *session*. The session manager bounds the number, S, of sessions that may concurrently exist in the system. The exact problem definition considers three main events:

1. **Session Creation**. There are two preconditions that must be fulfilled before this event is triggered. The first precondition is that a node must be eligible. In the context of the k-token dissemination problem, in order for a node to become eligible, it has to gather the minimum number of tokens needed to start sending coded tokens. The second precondition is that the limit of current sessions in the system is not exceeded if the eligible node creates a session. If a node satisfies these two preconditions, then it is allowed to create a session. In the context of k-token dissemination problem, once a node has enough tokens, if the limit of current sessions in the system has not been reached, then the node can immediately create a session. Once a session is created, the *session creation* event is triggered. Both the creator of a session and the rest of the nodes must allocate buffer space for the information associated to the created session.

   Finally, the creator of the session needs to wait for the session to become stable[7] in order to start sending random linear combinations of tokens. Meanwhile, the rest of the nodes in the network must verify that the creator of a session stays eligible and connected. Furthermore, there can only be one creator of a session performing its task at a time.

2. **Session Termination**. A session should terminate, once all correct nodes have received all the information associated to the task. In the context of k-token dissemination problem, this means that a node responsible for a session has to be sure that all nodes in the network are able to decode all tokens associated to that session before terminating it.

   In addition, a node, which is not responsible for a session, can not deliberately end such session, unless the session has expired. Once the node responsible for a created session is completely sure that all nodes in the network have received the information associated to the task, then it must end such session. As soon as a session is ended by its creator, the rest of the nodes in the network must eventually realize of

[7] A session becomes stable when all nodes agree on that session for at most $2(n-1)$ consecutive rounds in a highly dynamic network. This will be explained with more detail in Section 7.

this event and remove the session and its associated information from the system.

3. **Session Expiration**. The precondition for a *session expiration* event to be triggered is that a node realizes that the creator of a session, which currently exists in the system, is not connected to the network. Since the creator of the session is the only one capable of terminating the session, the action that needs to be carried out by the rest of the nodes once this event is triggered consists of removing the session from the system within a finite time. Otherwise, the session and its associated information will remain in the system "forever".

6 Basic Techniques

In this section, we show different basic problems and their respective techniques to solve them. These problems consist of minor issues associated to the more general studied k-token dissemination and session management problems, which need to be solved in order to provide a correct functioning of the algorithms. We present a comparison problem for knowing when all nodes in the network have the same information, along with two solutions for such problem. The first described solution is used by the nodes to know when all nodes have gathered the k-tokens. The second solution provides the nodes a mechanism to know whether all nodes possess the same information associated to a particular session. Finally, we talk about the *wrapping around* problem and its solution.

6.1 Knowing that All Nodes Possess Same Information

The two approaches for solving the k-token dissemination and session management problems have in common the way they deal with the problem of knowing when all nodes have the same information.

When Haeupler et al. [14] describe their algorithm, *greedy-forward*, they assume that after $O(n)$ rounds of network coding phase, all nodes will be able to decode the tokens. But what if it takes more time for the nodes to be able to decode the tokens, or on the other hand, it takes less time. It is necessary that all nodes in the network know exactly when they have the same set of k tokens so the algorithm can finish. In addition, it is also needed that the creators of sessions know the exact moment when all nodes can decode the tokens associated to their sessions.

6.1.1 Comparing Tokens

On one hand, in order to check if all nodes have the same set of tokens, these nodes append to the packet a hash value of the ordered list of tokens that they possess together with a *Time To End* counter. Two nodes agree to have the same list of tokens if the hash value is the same. Every time there is an agreement between two nodes, the maximum of the two counters is taken and is decreased by one, otherwise is set to[8] N. In this way, when the counter reaches value 0, it means that all nodes have the same set of tokens, and therefore the algorithm must trigger an event indicating that all nodes in the network possess the same set of tokens.

6.1.2 Comparing Coded Tokens

On the other hand, a creator of a session needs to know when the rest of the nodes have received enough number of linear independent packets so they can decode the tokens that those packets contain. Hence, it is necessary to implement a similar technique as the previous one. But this time, instead of just having plain-text tokens, there are coded tokens together with their respective coefficient vectors which need to be compared. It is not possible to lexicographically order these vectors and hash them as done in the above mentioned approach. The reason why this cannot be carried out is because it is possible that two nodes have received different packets but they actually have the same information or knowledge. For instance, suppose that a node *u* receives two packets[9] $\langle 1, 0, 0, 1\rangle$ and $\langle 0, 1, 1, 0\rangle$, and another node *v* receives also two packets: $\langle 1, 1, 1, 1\rangle$ and $\langle 0, 1, 1, 0\rangle$. It can be observed, that both nodes do not have enough packets to be able to decode at least one token, but they have the same knowledge about the tokens. In order words, in terms of knowledge, it can be stated that both nodes know the same, but the vectors that they have are not the same.

For this reason, it is necessary to devise a method that allows the nodes to know whether they have the same knowledge about the coded tokens, even though they have received different packets. The solution proposed for this problem consists of calculating and keeping the *Gaussian Elimination*[10] matrix of the packets received so far. This matrix will be hashed and sent together with the *Time To End* counter (as well as when comparing the set of tokens).

[8] Where N is an upper-bound on the number of nodes in the network.
[9] We only look at the coefficient vector.
[10] As explained in the Introduction, the Gaussian Elimination algorithm employed to calculate the matrix has a small modification in comparison to the general algorithm.

Moreover, it should be mentioned that this technique is not only useful for the creator of a session, but also for the rest of nodes. This is because, if the creator crashes, the rest of nodes eventually will notice it due to the fact that they all will have the same knowledge about the same coded tokens and if they can not decode all the tokens, they will realize that the node responsible for the network coding has crashed. Thereby, this method can be also used as a fault detector with *weak* completeness because it can be the case, that the elected node crashes, but before doing it, it has sent enough linear independent packets to the rest of the nodes in order for them to be able to decode all tokens, and consequently, the correct nodes will not detect that the elected node has crashed.

6.2 Wrapping Around Problem

Both proposed algorithms make use of incarnation numbers. These incarnations numbers are counters that tell the nodes whether a node just joined the network after crashing. Since the size of the data type used for the counters is limited, it is necessary to find a solution for when the bound value is increased by one. For the sake of simplicity, suppose that the *incarnation number* is represented by 2^8 bits value and that the current value of the *incarnation number* is 255. The next time a node creates a session, it will increase by one the *incarnation number*, resulting in zero value. When comparing the new created session with *incarnation number* equal to 0 and older ones, by default, the comparison will return that the newest created session is the oldest one, which is not true. Therefore, the comparison method must be modified in order to solve this problem.

6.2.1 Comparing Incarnation Numbers

To solve the problem, we need to redefine the comparison technique when the incarnation numbers that need to be compared are close to the boundary. An incarnation number is close to the boundary if it belongs to the interval $(I-R,(I+R) \mod (I+1)]$. In order to use the redefined comparison method, at least one of the incarnation numbers has to belong to the interval $(I-R,I]$ and the other one has to belong to $(I-R,(I+R) \mod (I+1)]$. If one of the two incarnation numbers that need to be compared does not belong to the big interval, then a regular comparison method is used.

Suppose there exist two nodes p_i and p_j that are neighbours. And p_j sends a packet to p_i containing $IncarnationNumber_j$, which belongs to the interval $(I-R,I]$. When p_i compares $IncarnationNumber_j$ with its incarnation number, $IncarnationNumber_i$, which belongs to the interval $(I-R,(I+R) \mod (I+1)]$, instead of directly comparing them, it uses a redefined comparison method that behaves modulo $I+1$, where I is the maximum value the incarnation number can reach. In addition it is required to add R to the incarnation number, where R represents the maximum difference between incarnation numbers[11]. Furthermore, R should satisfy two conditions: $R \ll I$ and $R \geq S$, where S is the maximum number of sessions allowed in the system at the same time. Thereby, the resultant comparison method will look like this: $(IncarnationNumber_j + R) \mod (I+1)$ compared with $(IncarnationNumber_i + R) \mod (I+1)$.

For sake of simplicity, suppose we have the previous scenario where the maximum value of I is 255, and the incarnation number, $IncarnationNumber_i$, of a processor p_i is equal to 255. Now suppose, that the maximum allowed difference between incarnation numbers is two, $R = 2$. A node p_j creates a session and therefore, it increases the incarnation number by 1, $((255+2) \mod (255+1))$, $IncarnationNumber_j = 1$. When p_i receives p_j's packet and vice-versa, they will make use of the redefined comparison method: $(0+2) \mod (255+1)$ compared with $(255+2) \mod (255+1)$. The result of the comparison shows that the session created by p_i is older than the one created by p_j.

Finally, in order to give some intuition for how large R should be, suppose that $R < S$. In addition, suppose that there currently exists S sessions in the system, and all sessions have a unique incarnation number, where the first session is s_1 and its incarnation number is $IncarnationNumber_1$, and the last session is s_S and its incarnation number is $IncarnationNumber_S$. The difference between both incarnation numbers is S ($IncarnationNumber_S - IncarnationNumber_1 = S$). Since S is greater than R (and $R > 1$), it can be the case that we are comparing $IncarnationNumber_1$ and *IncarnationNumber*$_S$, where $IncarnationNumber_1$ is equal to I, and hence, it belongs to the interval $(I-R,I]$. And where $IncarnationNumber_S$ is equal to $I + S \mod (I+1) = S-1$, therefore, it does not belong to the interval $(I-R,(I+R) \mod (I+1)]$. When comparing both incarnation numbers, since $IncarnationNumber_S$ does not belong to the interval, the regular comparison method is used. The comparison method will return that $IncarnationNumber_S$ is lower than $IncarnationNumber_1$, i.e. *IncarnationNumber*$_S$ is older than $IncarnationNumber_1$

[11]We assume that all the incarnation numbers existing in the system are comparable, i.e. $\forall\ p_i, p_j, \nexists\ IncarnationNumber_i$, $IncarnationNumber_j$, s.t. $|IncarnationNumber_i - IncarnationNumber_j| \mod (I+1) > R$.

which is not true. Moreover, if $R = S$, the comparison may also fail if the difference between $IncarnationNumber_S$−*IncarnationNumber*$_1$ is greater than S. Thus, it is necessary to set a large enough value to R, in order to avoid these scenarios where the comparison method misbehaves.

7 Algorithm Design: One Session at a Time

In this section, a first approach to solve the k-token dissemination and session management problems is described. The main algorithm consists of several algorithms and whose main purpose is to provide a stable session once a node has reached the threshold of minimum number of tokens needed to start the network coding phase and all nodes agree on that particular session. It is stable in the sense that no other node will start its own network coding phase even though it has also reached the threshold. While there is an agreement, the nodes send network coded packets, if not, packets containing plain-text tokens are sent. Moreover, this algorithm provides a mechanism to have a *back-up* session in case the current session terminates or expires.

7.1 Session Management: One Session at a Time

For this problem, it is allowed to have in the system only one session at a time. Thus, the nodes keep information associated to only one session. In addition, there exists the possibility for the nodes to switch to a *back-up* session, in case the node responsible for the an existing session crashes or terminates the session. However, the nodes do not allocate buffer space for the *back-up* session and its associated information until the session leaves the *back-up* condition to become principal session. The associated information of a session consists of all the coded packets (coefficients and coded tokens) received by a node, together with the generated Gaussian Elimination Matrix is needed to decode such tokens. The main goal of the algorithm is that all nodes agree on the same session for the session to become stable. And consequently, the node responsible for such session can start with the network coding phase.

7.1.1 Failure-Free Legal Execution

Here, we explain how the algorithm should behave when no failures take place in the network.

- When a node, p_i, gathers the minimum number of tokens needed to start broadcasting random linear combinations of tokens, an abstract event is triggered. This event indicates that the node is ready to start sending coded tokens, i.e. the node is eligible. Once this event is triggered, if there is no stable session existing in the system and the existent session (if any) is lexicographically lower, then the node creates a session, s_i. This session will be broadcast by its creator to all its neighbours.
- Once p_i broadcasts s_i, all its neighbours immediately receive it. The goal of broadcasting s_i is that all nodes eventually agree on s_i.
- While all nodes try to agree on one session, plain-text tokens are sent (piggybacking technique).
- Once all nodes have agreed on s_i, this session becomes stable and the node responsible for s_i, i.e. p_i, is allowed to start sending random linear combinations of tokens.
- When all nodes have enough number of linear independent tokens in order to decode all tokens belonging to a particular session, s_i, p_i will terminate s_i. This event will eventually be noticed by all nodes in the network, and thereby, the session will eventually become obsolete and removed from the system together with its associated information.
- After the termination of a session, If all nodes have agreed on a *back-up* session, such session becomes the principal session.

7.1.2 Non-Failure-Free Legal Execution

A node can be affected by three different types of failures: *crash-stop*, *crash-reboot* and *crash-resume*.

7.1.2.1 Crash-stop. If a node is affected by a *crash-stop* failure, once the node crashes, it will never be part of the network again. Depending on the node affected by this failure, the failure will lead to two different consequences.

1. If the crashed node was not responsible for any session (*principal* session or *back-up* session), the rest of the nodes will not even notice that another node has crashed.
2. If the crashed node, p_i, is the responsible for a stable *principal* session, s_i, or *back-up* session, bs_i, the rest of the nodes will eventually notice that the node responsible for s_i or bs_i is not longer in the system.

Once the rest of the nodes realize that no node is maintaining session s_i or bs_i, the session expires and therefore, they will remove it from the system. In the case of s_i, its associated information (if any) will be also removed.

7.1.2.2 Crash-reboot.

A node p_i may crash and reboot. The main difference between this type of failure and the previous one is that after crashing, the node may join the network again with all its variables set to their respective initial values. After the node reboots and unless the current *Incarnation number* is equal to 0, all nodes will have to agree on a new *Incarnation number*.

As in the previous failure, if p_i was responsible for a session s_i or bs_i that still exists in the system after the node crashes, this session will eventually disappear from the system.

7.1.2.3 Crash-resume.

A node p_i may also crash and resume. That means that after crashing, the node may join the network with the same state as right before crashing. During the time that the node is absent, the rest of the nodes may make some progress (e.g., creating and removing *principal* and *back-up* sessions, etc.). Depending on the progress made, the crashed-resumed node may recover all, partial or none of the tokens associated to a session.

- If p_i has an older *Incarnation number*[12] than the rest of the nodes, it will lead to a disagreement in the network and the calculation of the new *Incarnation number*.
- If p_i has the same *Incarnation number* as the rest of the nodes, nothing will happen.
- If p_i created a session s_i before crashing, once the node has resumed and as long as all nodes agree on the same incarnation number, the node will keep maintaining the session and broadcasting random linear combinations of tokens until all correct nodes in the network can decode all tokens associated to that particular session.
- If p_i has a session, s_j, that the rest of the nodes still agree on, p_i will have the chance to retrieve the tokens belonging to that session. This chance will depend on two factors: $a)$ the node, p_j, responsible for that session does not crash. And $b)$ p_j notices on time that p_i still needs to receive more packets in order to decode the tokens associated to the session.

[12]As mentioned in sectin 5.2, an incarnation number is a counter that tells the nodes whether a node just joined the network after crashing.

7.2 White and Black nodes Algorithm

Before defining the main algorithm, it is necessary to know how information is spread in a highly dynamic network. In order to do that, the *White and Black nodes* algorithm is described in this section. For the sake of simplicity, the proofs used in the main algorithm will make use of the lemma of this basic algorithm. In addition, this algorithm is also going to be exploited by the second approach, which can be viewed in the next section.

Intuition. If a black node is neighbour of a white node, then the white node becomes black.

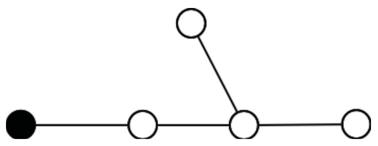
Lemma 1 *In an always connected dynamic graph, where the graph can change completely from round to round, suppose an initial configuration where all nodes are coloured with white color but one, which is coloured with black. We say that after at most $N-1$ rounds, where N is the number of nodes in the graph, all nodes will be black.*

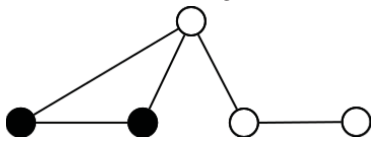
Proof. Since the graph is always connected, in every round, at least one black coloured node is the neighbour of at least one white coloured node, i.e. in every round at least one white coloured node becomes black. Thus, after at most $N-1$ rounds we can assert that all the nodes in the network are black. ■

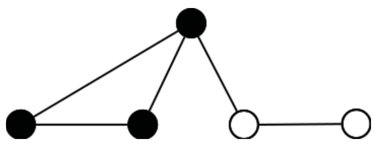
7.2.1 Why do we use $N-1$ instead of D?

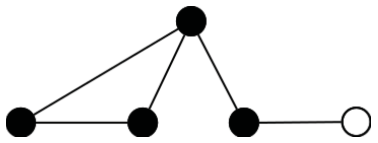
Suppose that the graph is always connected and dynamic, and that D (diameter of the graph) remains the same value throughout the execution of the algorithm, independently of the topology changes that the graph may suffer.

If the graph is static, we know that the number of steps needed for all nodes in the graph to become black coloured is D. Can we state the same for an always connected dynamic graph where D remains constant during the execution of the algorithm? The answer is: No. It can be proved with the following counter example, where D is equal to 3 throughout the execution of the algorithm:


(a) Initial Configuration


(b) After Round 1


(c) After Round 2


(d) After Round 3

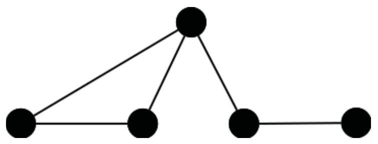

(e) After Round 4

Figure 2: Counterexample

In Figure 2, it can be observed that due to the dynamic nature of the graph the total number of steps needed to color the entire graph is 4 instead of 3, even though the diameter of the network remains unchanged throughout the execution of the algorithm. Note that a small change in the topology has been made in the first round (see Figure 2b) with respect to the *Initial Configuration*. Therefore, we conclude that at most $N - 1$ rounds are needed to spread the information from one node to the rest of the nodes in a dynamic network.

7.3 Self-stabilizing Wave Algorithm

A distributed algorithm is a *Wave* algorithm if the three following requirements are satisfied [25]:

- Termination: Each computation is finite.
- Decision: Each computation contains at least one decide event.
- Dependence: In each computation each decide event is casually preceded by an event in each process.

A wave algorithm exchanges a finite number of messages and then makes a decision, which depends on the events that take place in each process.

This algorithm is used a template for three other different algorithms that run concurrently in order to solve the session management and k-token dissemination problems. The other three algorithms are: *Maximum_ID*, *Session Verifier* and *Maximum Incarnation Number*.

The algorithm makes use of 5 main variables: N, X, x, $StabilityCounter$, and $ExistenceCounter$.

- N: It is an upper bound on the total number of nodes in the network.
- X: It represents a small x.
- x: A unique value composed of an incarnation number and a unique ID.
- $StabilityCounter$: It indicates whether X is stable or not.
- $ExistenceCounter$: It is used by the nodes to know whether the responsible for X exists in the system or not.

The reason why we make use of these two counters is because the nodes need to know whether a session is stable or not. In addition, they also need to know whether its creator exists in the system or not. The counters will provide such information, becoming this way in key elements for our algorithms.

7.3.1 Algorithm

Algorithm 1 Self-Stabilizing Wave (Template)

```
Require: $X_i$, $x_i$, $StabilityCounter_i$, and $ExistenceCounter_i$;
function UPON NEW_ROUND
    $StabilityCounter_i \leftarrow StabilityCounter_i + 1$
    $ExistenceCounter_i \leftarrow ExistenceCounter_i + 1$
    $StabilityCounter_i \leftarrow min(StabilityCounter_i, N)$
    $ExistenceCounter_i \leftarrow min(ExistenceCounter_i, N)$
    if $X_i = x_i$ then
        $ExistenceCounter_i \leftarrow 0$
    else
        if $ExistenceCounter_i = N$ then
            raise INEXISTENCE_INDICATION();
        end if
    end if
    SEND($X_i, StabilityCounter_i, ExistenceCounter_i$);
end function
```

```
function UPON RECEIVE($X_j$, $StabilityCounter_j$, $ExistenceCounter_j$)
    if $X_j = X_i$ then
        $StabilityCounter_i \leftarrow min(N, StabilityCounter_i, StabilityCounter_j)$   # Converge to the min
        $ExistenceCounter_i \leftarrow min(N - 1, ExistenceCounter_i, ExistenceCounter_j)$
    else
        raise DISAGREEMENT_INDICATION($X_j$, $StabilityCounter_j$, $ExistenceCounter_j$)
        Refresh()   # Set stability counter to 0
    end if
end function
function RESTART   # Reset the variables
    $X_i \leftarrow x_i$
    $StabilityCounter_i \leftarrow 0$
    $ExistenceCounter_i \leftarrow 0$
end function
function REFRESH
    $StabilityCounter_i \leftarrow 0$
end function
function ADOPT($\langle Incarnation, NodeID\rangle X$, $int\ StabilityCounter$, $int\ ExistenceCounter$))
    $X_i \leftarrow X$
    $StabilityCounter_i \leftarrow StabilityCounter$
    $ExistenceCounter_i \leftarrow ExistenceCounter$
end function
function ISSTABLE
    if $StabilityCounter_i = N$ and $ExistenceCounter_i < N$ then
        return $true$
    end if
    return $false$
end function
```

7.3.2 Properties

Property 1. In every round, both counters (*StabilityCounter* and *ExistenceCounter*) are increased by one, being N the maximum value they can reach.

Property 2. Every time a message, containing an $X_j \neq X_i$, is received, $StabilityCounter_i$ is set to 0.

Property 3. Every time a message, containing an $X_j = X_i$, is received, $StabilityCounter_i$ converges to the minimum value out of N, *StabilityCounter*$_j$ and itself. And $ExistenceCounter_i$ is set to the minimum value out of $N-1$, $ExistenceCounter_j$ and itself.

Property 4. When $ExistenceCounter_i$ reaches value N, an non-existence indication is raised. This means that either X_i does not longer exist in the system or it has been disconnected from the network for at least one round.

Property 5. A node is considered to be in a stable configuration, when the stability counter is equal to N and the existence counter does not exceed $N-1$.

Lemma 2 *If there is a node in the network with different X than the rest of nodes, all nodes in the network will become aware of this in at most $N-1$ rounds. After $N-1$ rounds and while there exists at least one node with different X, the stability counters of all nodes will never reach value N.*

Proof. According to lemma 1, we know that the information spreads from one node to the whole network in at most $N-1$ rounds. Hence, if there is at least one node that has a different X, this will be known by the rest of the nodes in at most $N-1$ rounds. Looking at the neighbours, there are two different cases for any arbitrary configuration:

I Neighbour with different X. Suppose that p_i and p_j are neighbours throughout the execution of the algorithm and have different X $(Xi \neq Xj)$. After one round, they will send to each other a message containing their X, and since they are not equal, the stability counter will be set to 0 according to Property 2. In the next round, the stability counter will be increased by one, but once they receive the message from the other node, they will set it again to 0. The same will happen in all successive rounds, therefore $StabilityCounter$ will never reach N for none of these two nodes.

II Neighbour with same X. Assume we have the same scenario as in *(I)*, but this time there is one more node p_r, whose $X_r = X_i$. Even if p_r does not receive any message from p_j (it only receives it from p_i) throughout the execution of the algorithm, according to property 3, $StabilityCounter$ will converge to the *min*, and thus it will adopt the value of $StabilityCounter_i$. Which will never be greater than 1 for this case. Therefore its $StabilityCounter$ will never reach N.

■

7.4 Maximum_X Algorithm

Maximum_X algorithm is an algorithm that inherits from *Self-stabilizing Wave* algorithm and its purpose is to continuously find the greater X among all nodes in the network. Moreover, the information given by this algorithm allows a node to know whether the node to which X belongs, is connected to the network or not.

7.4.1 Algorithm

Algorithm 2 Maximum_X

```
Require: $X_i$, $x_i$, $StabilityCounter_i$, and $ExistenceCounter_i$;
function UPON EVENT INEXISTENCE_INDICATION
    Restart()
end function
function UPON EVENT DISAGREEMENT_INDICATION($X_j$, $StabilityCounter_j$, $ExistenceCounter_j$)
    if LexicographicalCompare($X_i, X_j$) = GREATER then    # $X_j > X_i$
        Adopt($X_j, StabilityCounter_j, ExistenceCounter_j$)
    end if
end function
```

As it can be observed in the pseudo-code above, *Maximum_X* algorithm implements two event listeners (line 1 and 4). The variable x_i represents a value(e.g., numerical, character, etc.) and X_i represents the greatest lexicographical value seen by a node.

7.4.2 Properties

Property 1. Once an non-existence indication has been triggered, the counters are set to 0 and $X_i = x_i$.

Property 2. When a disagreement event is triggered and X_j is greater than X_i, then the node p_i adopts the values of counters of p_j and its X_j.

Lemma 3 *We can relate the black and white nodes lemma to our Maximum_X algorithm, where the node with greater x_i is the black node, and the rest are white nodes. Therefore, we assert that after at most $N-1$ rounds $X_i = X_j$, $\forall$ p_i, p_j where $i \neq j$. In other words, after at most $N-1$ rounds all nodes in the network have the same X.*

Lemma 4 *After at most $N-1$ rounds all the floating X(s)[13] will disappear from the system.*

Proof. According to Lemma 1, we know that the information spreads from one node to the whole network in at most $N-1$ rounds. There are two different cases for any arbitrary configuration:

I Suppose that $\forall\, p_i, \exists\, X_j, X_i$ s.t. $X_i > X_j$ and $\exists x_i = X_i$, but $\nexists\, x_j$ s.t. $x_j = X_j$. For this first case is trivial to see that since X_j is not the greatest value in the network, after at most $N-1$ rounds, when the node having its X equal to X_j, receive X_i, it will keep the one with greater value, in this case X_i, and thus, X_j will disappear from the system.

II Suppose that $\forall\, p_i, X_i = X$ and $\nexists\, x_i$ s.t. $x_i = X$. The $ExistenceCounter$ of the nodes will be not equal to 0. 0, since the only one allowed to set this counter to 0 is p_i with $x_i = X$, which does not exist in the system any more. The existence counters of the nodes can only increase according to Property 1.1, thus after $N-1$ rounds, the existence counters of all nodes will reach the value N, which means, according to Property 2.1, the nodes have to reset their counters and set their $X_i = x_i$. Therefore, the floating value disappears from the network.

■

Corollary 5 *After at most $2(N-1)$ rounds, $\forall\, p_j, \exists\, x_i \geq x_j$, s.t. $X_i = x_i$, and $X_i = X_j$. And within N extra rounds, all nodes know that the rest of the nodes in the network agree on the same X_i , i.e. $\forall\, p_i, p_j$ where $i \neq j$, $StabilityCounter_i = StabilityCounter_j = N$.*

Proof. Suppose there is at least one node such that its X is floating, and X is greater than any x_i existing in the network. According to Lemma 4, within at most $N-1$ rounds, this value will disappear from the system. Once the floating X has vanished, the nodes will set $X_i = x_i$, and the $StabilityCounter$ to 0. According to Lemma 3, within $N-1$ extra rounds, all nodes will have the

[13]Floating X(s) are all the X_j whose x_j does no longer exist in the system.

same X. Now, in order for the nodes to know that the rest of the nodes have the same X, it is necessary that the $StabilityCounter$ reaches N, which will happen within at most N rounds after all nodes have the same X. The reason why N rounds are enough, is because once all the nodes have the same X, the $StabilityCounter$ will never be reset to 0 since all the nodes possess the same X, therefore the counter can only increase. According to the algorithm, each round $StabilityCounter$ increases by 1, hence, after N rounds its value will be equal to N. Consequently, within a total number of $3N - 2$ rounds all nodes know that they agree on the same X. ■

7.4.3 Maximum_ID and Incarnation Number

The *Incarnation Number* and *Maximum_ID* algorithms are instantiations of the *Maximum_X* algorithm. The only difference between them are the values that x_i and X_i represent.

In the case of the *Incarnation Number* algorithm, the small *x* is a numerical value that represents the incarnation number and the big *X* represents the maximum incarnation number that all nodes will eventually agree on.

On the other hand, in the case of the *Maximum_ID* algorithm, the small *x* is a numerical or character value that represents a unique ID of a particular node and the big *X* represents the maximum ID that exists in the system and which all nodes will eventually agree on.

7.5 Session_Verifier Algorithm

The *Session_Verifier* algorithm is responsible for, as it can be deduced from its name, verifying two things: (a) The session is stable and (b) the node responsible for that session exists in the network.

7.5.1 Algorithm and Properties

Session_Verifier algorithm is simply an instantiation of the *Self-Stabilizing Wave algorithm* without any extension. Thus, it shares all its properties.

The variable x_i consists of an *incarnation number* and the ID of the node ($x_i = \langle IncarnationNumber, NodeID\rangle$). X_i represents the session that all nodes agree on.

As explained earlier, this algorithm only verifies that all nodes agree on the same session ($\langle IncarnationNumber, CreatorOfSessionID\rangle$). At the exact moment when there is a disagreement on the session, the stability is *broken* and this will spread to all nodes in the network. This leads to the following Corollary.

Corollary 6 *According to Lemma 2, if there is a node with different X_i, all nodes will notice it after at most $N - 1$ rounds.*

The reason for having this algorithm is to ensure that once a session has been created and becomes stable , this session will remain in a stable status, even though another eligible node with greater ID joins the network.

7.6 Session Manager Algorithm

The function of this algorithm is to unify all the previous mentioned algorithms, and make them work together in order to end up with a final algorithm that is able to switch from one session to another in case, for instance, the creator of the current session is down and all nodes agree on the same $MaximumID$. It is also the node responsible for increasing the incarnation number every time a change of session takes place.

7.6.1 Algorithm

Algorithm 3 Session Manager

```
Require: $Session, MaximumID, Incarnation$;
1: function UPON END_OF_ROUND
2:     if $Session.IsStable() = false$ and $MaximumID.IsStable = true$ and $Incarnation.IsStable() = true$ then
3:         $Incarnation.x_i \leftarrow Incarnation.x_i + 1$
4:         $Incarnation.X_i \leftarrow Incarnation.X_i + 1$
5:         Session.Adopt($\langle Incarnation.X_i, MaximumID.X_i\rangle, MaximumID.StabilityCounter_i, MaximumID.ExistenceCounter_i$)
6:     end if
7: end function
```

The three variables required by the algorithm are instantiations of the previous algorithms. *Session* is an instantiation of the *Session_Verifier* algorithm. *MaximumID* is an instantiation of the *Maximum_ID* algorithm and *Incarnation* is an instantiation of the *Incarnation Number* algorithm.

The *Session Manager* algorithm is executed at the end of each round, and as shown in line 4 of the pseudo-code, it checks if there is a stable session, i.e. all nodes agree on the same session. In negative case, it tries to switch to the current $maximumID$. To do so, it also checks if all nodes agree on the same $maximumID$ and on the same incarnation number. In case these two verifications return true, then the current $maximumID$ is taken as a new session, and the incarnation number is increased by one.

Remark 1 *Once all nodes know that they agree on the same X_i, there is not agreement on the session and all nodes agree on the same incarnation number, then according to the session manager algorithm, X_i will be the session and will not change unless p_i goes down or a node with different $CreatorOfSessionID$ or incarnation number joins the network.*

7.7 Advantages and Disadvantages of the Algorithm

7.7.1 Advantages of Allowing One Session at a Time

The main benefit of using an algorithm that allows only one session in the system at a time is that the nodes only have to store the information associated to that session. Therefore, this algorithm is recommended to be used in systems where their devices have memory constrains.

The information associated to a session is all coded packets received associated to that session and the Gaussian Elimination Matrix generated to decode the tokens contained in those packets. If t is the number of coded tokens and l the size of a token, then the maximum size required to store the matrix will be multiple of: $t \times (t+l)$. Where $(t \times t)$ is the square matrix containing the coefficient part, and $(t \times l)$ refers to the right side of the matrix containing the tokens.

7.7.2 Disadvantages of Allowing One Session at a Time

During the analysis of the algorithm a series of scenarios for which this algorithm can become really inefficient have been discovered:

1. Session Agreement - $O(n^2)$. Suppose the following scenario: the incarnation number is stable and the node with the lowest ID in the network is the first reaching the minimum threshold of tokens, therefore it becomes an eligible node, and since there is no other session, it creates one. According to Corollary 5, we know that after $2(N-1)$ all nodes in the network will agree on the same session. But, now suppose that after $2(N-1)-1$ rounds, the node with the second lowest ID reaches the threshold and also becomes eligible and creates a session, avoiding that the previous session becomes stable. The same can happen with the node with the third lowest ID, and so forth. All nodes will agree on the same session in this really pessimistic and unlikely scenario after $O(n^2)$ rounds, which is the same time that the simple random forwarding algorithm needs to solve the problem.
2. Recovered node may disrupt all previous work. A less unlikely scenario is that a crashed node recovers after a while, having an incarnation number that does not correspond to the incarnation number agreed by the rest of the nodes in the network. This disagreement, will lead to a change of current session (if there exists one), and therefore, all the work down by the creator of that session and the rest of the nodes, in terms of amount of random linear combinations packets that have been broadcast, will be disrupted by the recently joint node.

Moreover, it can be the case that a node crashes while there exist a creator of a session sending random linear combinations of tokens. If the creator of the session terminates such session before the crashed node resumes, the crashed node will not be able to obtain any of the already sent information.

8 Algorithm Design: S Sessions at a Time

In this section, the design of the algorithm that solves the k-token dissemination and session management problems allowing the existence of S sessions in the system at the same time is introduced.

8.1 Lessons learned from previous approach

There are several points that needed to be taken into consideration before starting designing the second algorithm. Some of these points are:

- Minimize disruption. One of the things that need to be mitigated with respect to the previous approach is the capacity that a node has to disrupt all the work done by a previous node.
- Avoid indefinite session agreement. If a node becomes eligible and creates a session, then after at most $2(N-1)$ rounds, this node or another node will start with the network coding phase, independently of if one or more nodes have created a session.
- Extend the back-up property. In the previous approach, if the creator of a session fails, then it is checked if there is agreement on the Maximum ID node and in affirmative case, the Maximum ID becomes the new creator of a session. It would be interesting to extend this property, so instead of having only one back-up, it is possible to have L back-ups.

Thereby, the algorithm will be able to tolerate until $L-1$ failures of session creators.

- Add possibility for a crashed-recovered node to recover some useful information. It is interesting if a node that has gone down and recovered within a limited period of time can have the chance to recover some of the coded tokens that have been sent while the node was absent.

8.2 Session Management: S Sessions at a Time

For this problem, instead of allowing only one session in the system, there can be up to $O(S)$ sessions existing in the system at the same time.

How these sessions must be handled is part of the new problem, as well as, how the algorithm can tolerate session failures. The final goal is that all nodes agree on the same set of sessions or at least in the same subset of sessions, so these sessions can become stable, and therefore the nodes responsible for those sessions can start broadcasting random linear combinations of tokens.

Before explaining the design of the algorithm, it is necessary to define the problem itself. These are some of the points that should be taken into account:

- A session becomes stable if all correct nodes agree on that session for at most $2N-1$ consecutive rounds.
- Once a session expires all nodes must eventually notice it.
- A node may crash and recover without disrupting all the work done by the current nodes in the network.
- The system should tolerate $O(S)$ session failures.

The idea to improve fault tolerance is that it is possible to have up to $O(S)$ stable sessions at the same time, so if one of the stable sessions fails then another stable session will take over.

8.2.1 Failure-Free Legal Execution

As with the previous algorithm, in the following points, we explain how the algorithm should behave when no failures take place in the network.

- When a node, p_i, gathers the minimum number of tokens needed to start broadcasting random linear combinations of tokens, an abstract event is triggered. This event indicates that the node is ready to start sending coded tokens. Once this event is triggered, the node creates a session, s_i, and places it in a list of sessions, l_i that will be broadcast to all its neighbours.
- Once p_i broadcasts l_i, all its neighbours immediately receive it. The goal of broadcasting s_i is that all nodes eventually agree on s_i.
- While all nodes try to agree on one or more sessions, together with the list of sessions, plain-text tokens are sent.
- Once all nodes have agreed on s_i, that session becomes stable and the node responsible for s_i, i.e. p_i, will be able to start sending random linear combinations of tokens (As long as s_i is the first stable session on the list of sessions.).
- When all nodes have enough number of linear independent tokens in order to decode all tokens belonging to a particular session, s_i, p_i will stop maintaining s_i. Therefore, it will eventually become obsolete. Before being removed from the list of sessions of all nodes in the network together with its associated information, it will be kept for a certain period in the system. The reason why it is not directly removed is because we want the nodes to have a chance to recover or provide useful information in case an crash-resume failure occurs.

8.2.2 Non-Failure-Free Legal Execution

A node can be affected by three different types of failures: *crash-stop*, *crash-reboot* and *crash-resume*. Note that the behaviour is similar to the algorithm where only one session is allowed, but instead of having one principal session and a back-up session, we have a list of sessions of size S.

8.2.2.1 Crash-stop. Depending on the node affected by this failure, the failure will lead to two different consequences.

1. If the crashed node was not responsible for any session, the rest of the nodes will not even notice that another node has crashed.
2. However, if the crashed node, p_i, had created and broadcast a session s_i before crashing, all nodes will eventually notice that the node responsible for s_i is not longer in the system. Once the rest of the nodes realize that no node is maintaining session s_i, they will remove it from the main list of sessions, and

placed temporally in another list. Eventually the session, together with its associated information, will be removed from the system.

8.2.2.2 Crash-reboot. As in the previous failure, if p_i was responsible for a session s_i that still exists in the system after crashing, this session will eventually disappear from the list of sessions of the rest of the nodes, but it will be kept for a while in another list.

8.2.2.3 Crash-resume. A node p_i may also crash and resume. That means that after crashing, the node may join the network with the same state as right before crashing. During the time that the node is absent, the rest of the nodes may do some progress (e.g., creating new sessions, removing all sessions, etc.). Depending on the progress made, the crashed-resumed node may recover all, partial or none of the tokens associated to the sessions.

- If p_i has an older *Incarnation number*, it will lead to a disagreement in the network and the calculation of the new *Incarnation number*.
- If p_i has the same *Incarnation number* as the rest of the nodes, nothing will happen.
- If p_i created a session s_i before crashing, once the node has resumed, it will keep maintaining the session until all correct nodes in the network can decode all tokens associated to that particular session.
- If p_i has in its list of sessions, sessions that are still maintained by other nodes in the network, p_i will have the chance to retrieve the tokens belonging to that sessions. This chance will depend on two factors: a) the nodes responsible for that sessions do not crash. And b) the nodes responsible for the sessions notice on time that p_i still needs to receive more packets in order to decode the tokens.

8.2.3 Managing S Sessions

In this approach, all nodes will possess a list of size S. The list is divided in three different lists of size S: *Old*, *Current* and *Future*.

- *Old list*. All sessions placed in this sub-list are either sessions whose creator is not longer in the system or sessions that have been finished because all nodes have the same information about the coded tokens or both of them[14].
- *Current list*. All sessions that belong to this sub-list are broadcast to the neighbours after the beginning of each round. Moreover, this list will indicate which kind of packet is transmitted, either a plain-text packet or a coded packet. If there is no stable session in this sub-list, then the content of the packet will be in plain-text.
- *Future list*. All sessions located in this sub-list are sessions which the node will potentially work with in the future. Every time a node reaches the threshold will firstly check if there is space in this list and if so, it will create and place the session in it.

When a node receives the sub-list of sessions, *CurrentList*, it will merge it with its *Current list*, and all the exceeding stable sessions will be merged with the sessions placed in *Future list*.

All sessions placed in *Current list* that either their *ExistenceCounter* has reached value N (i.e. the node responsible for that session does not longer exists in the system) or their *Time To End* counter has value 0 (i.e all correct nodes in the network have the same knowledge about the coded tokens and they can not learn anything new), are merged with the sessions located in *Old list*.

8.3 Archives

An *archive* is associated to a session and consists of: the *Gaussian Elimination Matrix* and all the random linear combinations of tokens belonging to that session. Once a session is removed from all sub-lists, its archive is also removed.

In the first approach, the nodes only store the coded tokens that belonged to the current elected node and its respective *Gaussian Elimination Matrix*. Once all nodes are able to decode the tokens or all nodes agree on a new session, the archives belonging to the previous session are removed. However, in this second approach a node can store up to $O(S)$ archives, since it can have up to S sessions.

8.4 S Sessions Manager Algorithm

8.5 Local Properties

Property 1. If all nodes agree on the same Gaussian Elimination Matrix from a session S_i and this session is placed in $CurrentList$ then, it is moved from $CurrentList$ to $OldList$.

Property 2. If any session in $CurrentList$ is also located in $OldList$, then it is removed from $OldList$.

[14]It is possible that the node responsible for the session has crashed after having already started to send coded tokens.

Algorithm 4 S Sessions Manager

```
Require: $session : \langle IncarnationNumber, ID \rangle$ , $FutureList[S] : \langle session \rangle$, $CurrentList[S] : \langle session \rangle$ , $OldList[S] : \langle session \rangle$;
 1: function UPON NEW_ROUND
 2:     if All nodes have same $GE\_Matrix$ from the same session $s_i$ then
 3:         Move $s_i$ from $CurrentList$ to $OldList$
 4:     end if
 5:     while $CurrentList$ is not $REPLETE$ and $FutureList$ is not $EMPTY$ do
 6:         Move sessions from $FutureList$ to $CurrentList$
 7:     end while
 8:     for all sessions in $CurrentList$ do
 9:         UpdateCounters(session)
10:     end for
11:     if session $s_i$ from $CurrentList$ is in $OldList$ then
12:         Remove $s_i$ from $OldList$
13:     end if
14: end function
15: function UPON RECEIVE($list_j$)
16:     Merge($CurrentList$, $list_j$);
17:     Move to $FutureList$, sessions that are stable, have archives and that are no longer in $CurrentList$
18: end function
19: function UPDATECOUNTERS($S_i$, $StabilityCounter_i$, $ExistenceCounter_i$ )
20:     $StabilityCounter_i \leftarrow StabilityCounter_i + 1$
21:     $ExistenceCounter_i \leftarrow ExistenceCounter_i + 1$
22:     $StabilityCounter_i \leftarrow min(N, StabilityCounter_i)$ # Converge to the min
23:     $ExistenceCounter_i \leftarrow min(N, ExistenceCounter_i)$
24:     if ($S_i = s_i$) or ($S_i.ID$ is $= s_i.ID$ and there are archives for $S_i$) then
25:         $ExistenceCounter_i \leftarrow 0$
26:     else if $ExistenceCounter_i = N$ then    # If the session's owner has crashed
27:         Remove $S_i$ from $CurrentList$
28:         if there are archives for $S_i$ then
29:             Move $S_i$ to $OldList$
30:         end if
31:     end if
32: end function
```

Property 3. If $OldList$ is full and a new session is added, then the first session is removed from $OldList$ along with its archives. Note that if the added session turns to be the first session, it is not even added to the list.

Property 4. If $FutureList$ is full and a session is added to the list, the last session along with its archives is removed from the list. Note that if the added session turns to be the last session, it is not even added to the list.

Property 5. When a list, $list_j$, is received from a neighbour, P_j, this list is merged with $CurrentList$. All the stable sessions(with stored archives) that are no longer in the final list, are added to the $FutureList$.

Property 6. The existence counter is set to 0, if and only if, S_i corresponds to a current session s_i, or if there exists any archive belonging to S_i.

Property 7. If a session in $CurrentList$ no longer exists, i.e. *ExistenceCounter* is equal to N, then this session is removed from $CurrentList$ and moved to $OldList$ only if there are some archives belonging to the session.

Property 8. A session is removed from $OldList$ when S newer sessions are added to the list.

Property 9. In every round if $CurrentList$ is not replete and $FutureList$ is not empty, sessions from $FutureList$ are moved to $CurrentList$ until $CurrentList$ is replete or $FutureList$ is empty.

Property 10. All the sessions stored in the lists are ordered according to their incarnation numbers (in case two sessions have the same incarnation number then the order will depend on the lexicographic value, where the session with greater ID is the first).

Lemma 7 *Once the creator of a session S_i, has removed this session from $CurrentList$ and $OldList$, this session will eventually disappear from the lists of the rest of the nodes.*

Proof. Suppose that there is a node that introduces a session s_i in the system. This can occur when a node has gone down and then it recovers after a while. In addition, without loss of generality, suppose that this session is the first session among the already-existing sessions in $CurrentList$. The list of this node will be broadcast and, consequently, the session S_i. According to the Black and White lemma, this session will be received by all the nodes in at most $N - 1$ rounds.

When the node with the same ID as the session, receives S_i, will see that S_i is no longer in $CurrentList$ or $OldList$, therefore according to Property 6 it will not set its $ExistenceCounter$ to 0, hence after at most $N-1$ rounds all the $ExistenceCounter$ of all nodes will reach value N. According to property 7, all nodes that do not have any archive belonging to session S_i will remove it from $CurrentList$. On the other hand, the nodes possessing archives that belong to this session will move S_i to $OldList$. Finally, according to Property 8, S_i will disappear from the system once all nodes that have S_i in their $OldList$ store S newer sessions. ■

Remark 2 *A session is removed from the lists in two possible ways:* $a)$ *When the existence counter of a session reaches* N *and it does not have any archives belonging to it, then the session is immediately removed.* $b)$ *In contrast, if there are archives belonging to the session, then the session is moved to* $OldList$ *and according to property 8, eventually removed from this list too.*

Lemma 8 *A session can always be recovered as long as the creator of the session keeps the session in one of its lists. (and as long as it behaves as a correct node.)*

Proof. There can be three possible scenarios by looking at where the session is placed:

I The session is placed in $OldList$. When a node receives a list containing a session S_i of which that node is the creator, and that session is placed in $OldList$, according to Property 6, the $ExistenceCounter$ of that session is set to 0. In addition, the session will be removed from $OldList$ and will be treated as a current session with its stability counter set to 1. In case, the node that receives the list is not the creator of S_i , the only difference is that the node will not set the $ExistenceCounter$ to 0.

II The session is placed in $CurrentList$. When a node receives a list containing a session S_i of which that node is the creator, and that session is placed in $CurrentList$, according to Property 6, the $ExistenceCounter$ of that session is set to 0 and the $StabilityCounter$ increased by one. In case, the node is not the creator both, the $ExistenceCounter$ and the $StabilityCounter$ are increased by one.

III The session is placed in $FutureList$. The session will eventually be moved to $CurrentList$.

■

Lemma 9 *There is no session in* $FutureList$ *with lower identification than at least one session in* $CurrentList$.

Proof. Assume to the contrary that there is a session in $FutureList$ with lower identification than at least one session in $CurrentList$. Note that there are two possible ways to add sessions to the $CurrentList$: by merging or by moving sessions from $FutureList$ to $CurrentList$. According to property 10, all lists are ordered, therefore, when there is space in $CurrentList$ for more sessions, the ones (if any) moved from $FutureList$ to $CurrentList$ are already ordered, hence all the remaining sessions(if any) in $FutureList$ will have a greater identification. Contradiction!. On the other hand, when merging two lists, the resulting list will be also ordered and all the exceeding sessions will be placed in $FutureList$(if possible). Therefore in order to have a session in $CurrentList$ with greater identification than at least one session in $FutureList$ after merging, one of the exceeding session that was moved to $FutureList$ had lower identification than at least another session that was placed in $CurrentList$, but since the sessions are ordered this leads to another contradiction! ■

Lemma 10 *After merging two lists, it is not possible to have the same session in both* $CurrentList$ *and* $FutureList$.

Proof. Assume to the contrary that after merging two lists there is a session S_i that is placed both in $CurrentList$ and $FutureList$, that means that the received list, $list_j$, contained a session S_i which after the merging process was added to $CurrentList$. If this is the case, that means that before merging there were sessions in $CurrentList$ that had a greater identification than S_i or that the union of $CurrentList$ and $list_j$ did not exceed the length (S) of the list. For the first case, we refer to Lemma 9 that shows that this is not possible. For the second case, according to property 9, at the beginning of every if $CurrentList$ is not replete and $FutureList$ is not empty, sessions from $FutureList$ are moved to $CurrentList$ until $CurrentList$ is replete or $FutureList$ is empty, therefore S_i should have be placed before in $CurrentList$ and removed from $FinalList$. Contradiction! ■

8.6 Global Properties

Lemma 11 *Once a session,* s_i *has been created, this session or another session created during the interval,* $N-1$, *will become stable after at most* $2(N-1)$ *rounds after the moment it has been created. (As long as no creator of a session crashes during this period.)*

Proof. According to Lemma 1, we know that the information spreads from one node to the whole network in at most $N-1$ rounds. And there can be two different scenarios for any arbitrary configuration (supposing that the 3 lists are empty):

I Suppose that in the interval of $N-1$, apart from the creation of s_i, a number of R sessions have been created, where $R > S$. Since R is greater than the size of the list, some sessions must be dismissed. Now, suppose that s_i is removed because it has lower id than any of the R sessions. Thus, s_i will never become stable, but there will be at least a session s_r that after $N-1$ rounds is located in the list of every correct node and after N extra rounds will become stable. This is due to the fact that every time the nodes send the list, they agree at least on s_r and therefore, they increase by one its stability counter, which after N rounds reaches its maximum value, N. This indicates that the session has become stable.

II Suppose that in the interval of $N-1$, apart from the creation of s_i, a number of R sessions have been created, where $R > S$. But this time, s_i is allocated in the list because of its ID. Consequently, after $N-1$ rounds, all nodes in the network will possess s_i in their lists. And after N extra rounds, its stability counter will reach value N, indicating that this session is stable.

■

Lemma 12 *If both lists, $Current$ and $Future$, are replete of stable sessions. The system can tolerate up to $2S-1$ failures, leading to a maximum period of $(2S-1)*(N-1)$ rounds without sending coded tokens.*

Proof. Suppose the creators, C_i, of the stable sessions, start crashing in the order their sessions are placed in the lists and with an interval of $N-1$ rounds. When the last session, s_{2S}, becomes the first element in the list, the node responsible for that session does not crash. For instance, the creator, c_1, of the first session, s_1, crashes. This will be noticed by the rest of the nodes within $N-1$ rounds. Once the rest of the nodes realize that the creator of s_1 is down, the session is removed from the list, $Current$. And the second session, s_2, becomes the first session in the list. But, it turns out that its creator, c_2, also crashes in that particular moment. Therefore, after $N-1$ rounds all nodes will now that c_2 is also down, and so on. Since, the maximum number of sessions that can be located in $Current$ and $Future$ sub-lists is $(2S)$, c_{2S} will need to wait $(2S-1)*(N-1)$ rounds until it can start sending random linear combinations of tokens. ■

9 Conclusions and Further Work

9.1 Conclusions

The *greedy-forward* algorithm given in [14], solves the k-token dissemination problem by means of network coding in a more efficient way than using a simple *random forwarding* approach. In this thesis report, we present the *session management problem* and two algorithms to solve it. By solving such a problem, we can also resolve k-token dissemination problem, providing a robust solution that can deal with certain failures, such as, crash-stop, crash-reboot and crash-recovery.

Two different algorithms have been designed during the realization of the project. The first algorithm provides a realistic and robust solution for the k-token dissemination and session management problems for when one session at a time is allowed in the system. The main advantage of using the first algorithm is that it is ideal for systems with memory constrains. However, there are some scenarios for which this algorithm becomes very inefficient. Whereas the second presented algorithm, which can handle S sessions at a time, needs more memory resources. But at the same time, besides the properties of the first algorithm, it also has some extra features that makes it more robust and to be able to perform efficiently in scenarios where the first proposed algorithm does not.

Moreover, even thought, no implementation or test of the algorithms has been carried out, the algorithms described in this work are accompanied by several lemmas and their respective formal proofs, which show the correctness of the algorithms themselves.

9.2 Further Work

Throughout the realization of this project, many problems and questions have arisen. These points can be further studied, in order to enhance the work done in this thesis. These are some of the open problems that I leave for future research:

- Estimating N. In this project we have assumed the N is known by all nodes, where N is an upper bound on the number of nodes in the network. It would be interesting to implement a method that could calculate an approximate number *n* of current nodes in the network in order to make the algorithm more efficient, since the more accurate the estimation is, the less number of packets are required to, for example, agree on the stability of a session and therefore, the algorithm will react faster to changes.

In [14], the authors propose an algorithm for the case of n-token dissemination, where the nodes ignore the value of n. It consists of starting guessing an upper bound $n = 2$, and then the nodes count the number of node IDs using token dissemination. If there is a failure, i.e. the number of received tokens from different nodes exceeds n, the n estimation is doubled and the algorithm restarted. This process is repeated every time a failure is detected. A similar technique can be used together with the proposed algorithm in order to make it more realistic and efficient.

Dolev et at.[11] give a solution to this problem, in a self-stabilizing group communication system for ad-hoc networks, in which the algorithm needs to know N in order to reach stabilization. The authors state that having a more accurate upper bound on n will ensure that the system will react in a faster way to addition/removal changes. They also indicate that estimating *n* will increase the number of short messages, which will lead to a trade-off between the time needed to estimate *n* and a special type of messages, called scouters.

In addition, Baldoni et al. [4] consider the problem of counting the number of nodes. The authors propose a more recent technique based on "energy transfer" to calculate the size of the network on anonymous dynamic network. They use the *energy-transfer* technique to deal with the dynamic environment and the absence of unique IDs. The solution proposed in [4] is leader based and where each node owns a fixed energy charge. A node discharges itself by exchanging at most half of its charge with its neighbours. Energy is not created or destroyed, i.e. after every round, the sum of energy contained in each node is the same. Nodes discharge themselves by exchanging at most half of their charge with their neighbours. On the other hand, the leader is responsible for absorbing all the energy, and thus, it does not transfer its energy to its neighbours. In order to calculate the number of nodes in the network, the leader measures the energy received.

- Extending the algorithm. The algorithm presented by Haeupler et al. and which has been used in the project as a starting point works well for a small size of message (b), but for very large $b \geq n^{1/3}$, as stated in [14], in the random-forwarding the nodes are not able to gather b^2/d tokens in a efficient way. Thus, Haeupler et al. propose another algorithm for such scenario. This algorithm, named *priority-forward*, also makes use of the first proposed algorithm(*greedy-forward*) but it uses a different approach for when no node gets b^2/d tokens after $O(n)$ rounds.

  It would be interesting to extend our proposed algorithm, in order to make it more efficient in cases where the size of the message is very large. Since *priority-forward* algorithm runs *greedy-forward* algorithm until no nodes gets b^2/d tokens, minor modifications will need to be done to our algorithm in order to design the mentioned extension.

- T-stable Token Dissemination. Haeupler et al. [14] also describe in their paper a more efficient algorithm for when the network is stable for a certain period of time. They define T-stable network, as a network that changes entirely only every T steps or rounds.

  This network model may seem less realistic, but in any case, it would be interesting to implement such algorithm and see which is the actual improvement(in practice) that network coding provides with respect to simple forwarding and compare it with the performance of the algorithms used for networks that are changing in every round.

- Implementing and Testing the algorithm. In the beginning of the project, when the goals were defined, one of the aims was to implement and test the algorithm, but due to lack of time, it could not be done. Hence, it would be interesting to deploy the algorithm in a network with a similar scenario to the one proposed in this thesis, and see which are the results obtained. This will give practical results that could corroborate the theoretical ones.

  A good scenario to implement and test the algorithm would be Wireless Sensor Networks(WSN). And, in particular, it is necessary to make use of an efficient MAC algorithm that can handle mobility. There already exists an adapted MAC algorithm that satisfies this requirement, it was implemented by Leone et al. and it was named as *Chameleon-Mac*

[22]. *Chameleon-Mac* is based on a self-stabilizing algorithm and employs a scheduled access strategy, known as *Time Division Multiplexing Access* (TDMA). The timeslots that a TDMA algorithm provides to the nodes will avoid that there are collisions and they will clearly define the rounds, which are an essential part of the algorithm. One new timeslot represents a new round in our algorithm.

A deeper analysis of *Chameleon-Mac* need to be carried out, in order to see whether the algorithm can be directly deployed or whether any modification need to be done with the purpose of having a good integration and a correct functioning of the algorithm.

References

[1] Marcos K. Aguilera, Carole Delporte-Gallet, Hugues Fauconnier, and Sam Toueg. Stable leader election. In Jennifer Welch, editor, *Distributed Computing*, volume 2180 of *Lecture Notes in Computer Science*, pages 108–122. Springer Berlin Heidelberg, 2001.

[2] R. Ahlswede, Ning Cai, S.-Y.R. Li, and R.W. Yeung. Network information flow. *Information Theory, IEEE Transactions on*, 46(4):1204–1216, 2000.

[3] A. Arora and M. Gouda. Distributed reset. *IEEE Trans. Comput.*, 43(9):1026–1038, September 1994.

[4] R. Baldoni, S. Bonomi, I. Chatzigiannakis, and G. Di Luna. Counting on anonymous dynamic networks through energy transfer. Technical Report 1, MIDLAB, 2013.

[5] S. Bhadra and S. Shakkottai. Looking at large networks: Coding vs. queueing. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, 2006.

[6] Ning Cai. *Network Coding Theory*. Now Publishers Inc, 2006.

[7] D. Conan, P. Sens, L. Arantes, and M. Bouillaguet. Failure, disconnection and partition detection in mobile environment. In *Network Computing and Applications, 2008. NCA '08. Seventh IEEE International Symposium on*, pages 119–127, 2008.

[8] Jean-Michel Couvreur, Nissim Francez, and Mohamed G. Gouda. Asynchronous unison (extended abstract). In *ICDCS'92*, pages 486–493, 1992.

[9] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, November 1974.

[10] Shlomi Dolev, Ronen I. Kat, and Elad M. Schiller. When consensus meets self-stabilization. *J. Comput. Syst. Sci.*, 76(8):884–900, December 2010.

[11] Shlomi Dolev, Elad Schiller, and Jennifer L. Welch. Random walk for self-stabilizing group communication in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(7):893–905, July 2006.

[12] Christos Gkantsidis and Mitch Goldberg. Avalanche: File Swarming with Network Coding. `http://research.microsoft.com/en-us/projects/avalanche/`. [Online; accessed 19-April-2013].

[13] Bernhard Haeupler. Analyzing network coding gossip made easy. *CoRR*, abs/1010.0558, 2010.

[14] Bernhard Haeupler and David R.Karger. Faster information dissemination in dynamic networks via network coding. *CoRR*, abs/1104.2527, 2011.

[15] Ted Herman and Chen Zhang. Best paper: stabilizing clock synchronization for wireless sensor networks. In *Proceedings of the 8th international conference on Stabilization, safety, and security of distributed systems*, SSS'06, pages 335–349, Berlin, Heidelberg, 2006. Springer-Verlag.

[16] Tracey Ho, R. Koetter, M. Medard, D.R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *Information Theory, 2003. Proceedings. IEEE International Symposium on*, pages 442–, 2003.

[17] I hong Hou, Yu en Tsai, Tarek F. Abdelzaher, and Indranil Gupta. Adapcode: Adaptive network coding for code updates. In *in Wireless Sensor Networks, in Proceedings of IEEE INFOCOM*, 2008.

[18] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft. Xors in the air: practical wireless network coding. *SIGCOMM Comput. Commun. Rev.*, 36(4):243–254, August 2006.

[19] MinJi Kim, Jason Cloud, Ali ParandehGheibi, Leonardo Urbina, Kerim Fouli, Douglas J. Leith, and Muriel Médard. Network coded tcp (ctcp). *CoRR*, abs/1212.2291, 2012.

[20] Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 513–522, New York, NY, USA, 2010. ACM.

[21] Leslie Lamport. Solved problems, unsolved problems and non-problems in concurrency. *SIGOPS Oper. Syst. Rev.*, 19(4):34–44, October 1985.

[22] Pierre Leone, Marina Papatriantafilou, EladM. Schiller, and Gongxi Zhu. Chameleon-mac: Adaptive and self- algorithms for media access control in mobile ad hoc networks. In Shlomi Dolev, Jorge Cobb, Michael Fischer, and Moti Yung, editors, *Stabilization, Safety, and Security of Distributed Systems*, volume 6366 of *Lecture Notes in Computer Science*, pages 468–488. Springer Berlin Heidelberg, 2010.

[23] Dolev Shlomi. *Self-Stabilization*. MIT Press, 2000.

[24] Andrew S. Tanenbaum. Network protocols. *ACM Comput. Surv.*, 13(4):453–489, December 1981.

[25] Gerard Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 2000.

A.1.4 Towards Resilient Real-Time Wireless Communications

"Towards Resilient Real-Time Wireless Communications", J. L. R. Souza and J. Rufino, Proceedings of the 25th Euromicro Conference on Real-Time Systems (ECRTS-WiP 2013), July 2013.

This page is intentionally left blank.

Towards resilient real-time wireless communications

Jeferson L. R. Souza and José Rufino
University of Lisboa - Faculty of Sciences
LaSIGE - Navigators Research Team
Email(s): jsouza@lasige.di.fc.ul.pt, ruf@di.fc.ul.pt

***Abstract*—The use of wireless networks on environments with real-time constrains requires timeliness and dependability guarantees to allow the execution of time-restricted networked operations. The current wireless standards and state-of-the-art solutions pay no or little attention to dependability aspects of communications, which are fundamental to secure any timeliness guarantee. This paper presents an innovative standard-compliant solution, dubbed *Mediator Layer*, which extends the MAC sublayer with additional components that can be incorporated within networked simulators and implemented in currently available hardware platforms. Thus, the *Mediator Layer* enhances and complements the services traditionally offered by wireless network standards, providing a set of fundamental abstractions useful for both system design and application programming.**

***Index Terms*—wireless sensor and actuator networks, real-time communications, dependability, timeliness, resilience.**

I. Introduction

The use of wireless networks on environments where communications may have real-time requirements is a current trend [1]. This trend is guided by the needs to reduce system size, weight, and power consumption (SWaP) without lessen timeliness and dependability guarantees. Industrial, vehicular, and aerospace environments are waiting for a solution to address an effective and efficient real-time support on wireless communications. For example, [2] discusses a set of pressing challenges on the use of wireless sensor networks (WSNs), more specifically wireless sensor and actuator networks (WSANs), in industrial automation, where dependability and real-time guarantees are a must.

So far, a lot of work has been done, proposing new medium access control (MAC) protocols [3]–[10], modifications on the existent standards [11]–[13], and abstract models [14] trying to enhance the reliability on wireless communications. However, all these works focus their analyses in the temporal aspects of the frame transmission service, paying no or little attention to the dependability aspects of communications, which are fundamental to secure any timeliness guarantee. An interesting conclusion draw in [2] is the necessity of improvements on the existent standards face to requirements of safe and secure networked operations.

Thus, this work-in-progress presents an innovative solution dubbed *Mediator Layer* [15], which is capable to enhance the dependability and timeliness of MAC sublayer services. In addition, a set of constructs useful to the programming of distributed applications, such as reliable communications, node failure detection, membership and clock synchronization may be offered immediately above the MAC sublayer. In this sense, our solution enhances and complements the services traditionally offered by wireless network standards, providing a set of fundamental abstractions useful for both system design and application programming. A prototype of the *Mediator Layer* is being integrated in the NS2 simulator [16] and in a commercial off-the-shelf (COTS) platform [17], using the IEEE 802.15.4 standard as a case study.

The presentation of our advances is organized as follows: Section II presents the system model. Sections III and IV describes our work-in-progress solution, including the incorporation of the *Mediator Layer* in the NS2 simulator and in the hardware platform. Finally, Section V presents some conclusions and future directions of this work.

This work was partially supported: by the EC, through project IST-FP7-STREP-288195 (KARYON); by FCT/DAAD, through the transnational cooperation project PROPHECY; and by FCT, through the project PTDC/EEI-SCR/3200/2012 (READAPT), the Multiannual Funding Program, and the Individual Doctoral Grant SFRH/BD/45270/2008.

II. System model

In this section we provide a brief description of our system model, which establishes a base foundation for our design and simulations. Our system model is formed by a set of wireless nodes $X = \{x_1, x_2, \ldots, x_n\}$, being $1 < n \leq \#A$, where A is the set of all wireless nodes using the same communication channel. A wireless node is a networked device capable to communicate with other wireless nodes. The set of nodes X itself defines a node relationship entity dubbed wireless network segment (WnS), which is established by all wireless nodes within $X \subseteq A$ that use a given communication channel and share a single hop communication range space.

For any WnS we use the following assumptions:

1) the communication range of X, i.e. its broadcast domain, is given by: $B_X = \bigcap_{j=1}^{n} B_D(x_j),\ \forall x_j \in X$, where $B_D(x_j)$ represents the communication range of node x_j;
2) $\forall x \in X$ can sense the transmissions of one another;
3) $\forall x \in A,\ x \in X \iff B_D(x) \bigcap B_X = B_X$ or, as a consequence of node mobility, $x \notin X \iff B_D(x) \bigcap B_X \neq B_X$;
4) $\exists x \in X$ which is the coordinator, being unique and with responsibility to manage the set X;
5) a network component (e.g. a node $x \in X$) either behaves correctly or crashes upon exceeding a given number

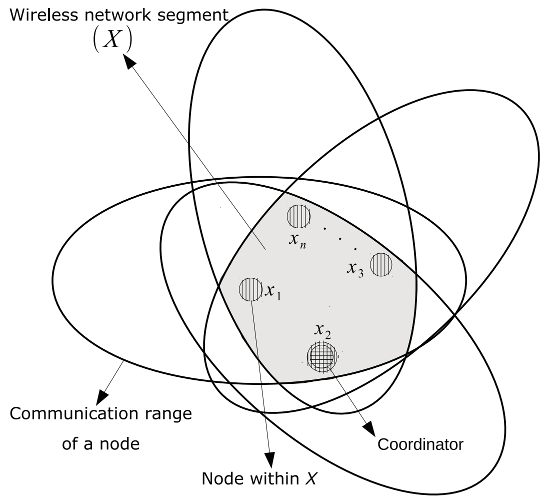
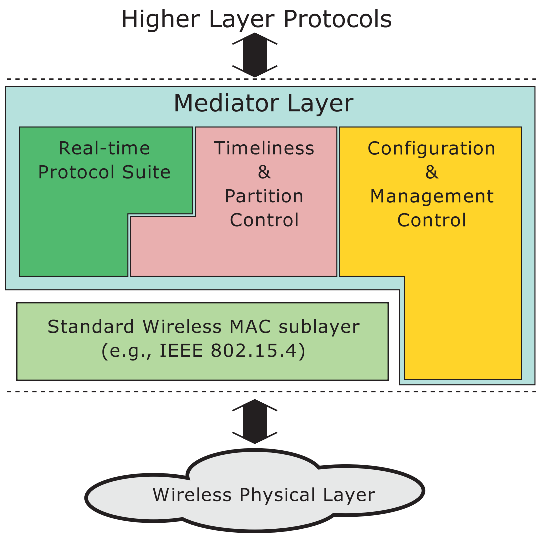


Fig. 1: Representation of a wireless network segment


Higher Layer Protocols
Mediator Layer
Real-time Protocol Suite
Timeliness & Partition Control
Configuration & Management Control
Standard Wireless MAC sublayer (e.g., IEEE 802.15.4)
Wireless Physical Layer
Fig. 2: The Mediator Layer and its components

of consecutive omissions (the component's *omission degree*, f_o) in a time interval of reference[1], $\mathcal{T}_{rd}$;

6) failure bursts never affect more than f_o transmissions in a time interval of reference, $\mathcal{T}_{rd}$;

7) omission failures may be inconsistent (i.e., not observed by all recipients).

Assumptions 1 to 3 define the physical relationship between nodes within the WnS. Our system model characterizes the relationship between nodes at MAC sublayer, where nodes must be in the communication range of each other to communicate, and are able to sense one another (assumption 2). Mobility may drive nodes away of the WnS (assumption 3).

In the context of network components, an omission is an error that destroys a data frame. Establishing a bound for the *omission degree* (assumptions 5 and 6) provides a general method for the detection of failed components. If each omission is detected and accounted for, the component fails once it exceeds the *omission degree bound*, k. The *omission degree* is thus a general measure of the reliability of network components with respect to accidental/intentional transient errors.

Figure 1 presents a graphical representation of a wireless network segment. In this figure we can see the communication range of each node within X, evidencing the intersection between all communication ranges of all nodes, which delimits the broadcast domain of X. One node within X assumes the role of coordinator (assumption 4). The management activities of the coordinator comprises the assignment of the current communication channel in use by the WnS, the allocation of guaranteed slots for frame transmissions, and so on.

[1]For instance, the duration of a given protocol execution. Note that this assumption is concerned with the total number of failures of possibly different wireless nodes.

III. Towards Resilient Real-Time Wireless Communications

Our approach to enhance the dependability and timeliness of wireless communications consists of an extensible component layer, dubbed *Mediator Layer*, build around a standard MAC sublayer. No modification is required to existing standards, being the *Mediator Layer* standard-compliant solution easily implemented in currently available COTS platforms. In this sense, the *Mediator Layer* approach significantly differs from other solutions described in the literature [3]–[14].

The *Mediator Layer* provides enhanced frame transmission and management services through a minimal set of fundamental components (draw in Fig. 2), which handle the actions required to secure dependability and timeliness in communications.

The *Real-Time Protocol Suite* (Fig. 2) component is responsible for handling frame transmissions, which can be data or management frames. Different protocols serving requests with different types of requisites, ranging from unreliable unicast to reliable broadcast, can be incorporated within this component. This augments the applicability of wireless networks to broader class of applications, including those with mixed-criticality requirements. Our intuition is that replacing a classical timeout-based protocol design approach with a combination of positive and negative confirmations (acknowledgements) contributes to reduce protocol worst case termination times, and thus service timeliness. Timeout-based techniques may still be used but only to detect node crash failures.

The *Timeliness and Partition Control* (Fig. 2) component deals with the temporal aspects related to the frame transmission service. Controlling and monitoring the timing of the actions within the *Mediator Layer* relies on an internal *Time Service* that ensures the temporal awareness of all networked operations, including the occurrence of temporary network partitions (i.e., network inaccessibility [18]) or even timeliness violations of a specific transmission protocol. An example is the monitoring and verification of a deadline associated to a data request, which may be disturbed by inaccessibility incidents. The dynamic control of inaccessibility periods [15] allows protocol execution to be aware of the real duration of inaccessibility incidents, and to (self-)adapt its execution to actual network operating conditions. In particularly, optimal timeout values can be used in the dimensioning of protocol timers, which contributes to enhance the timeliness guarantees.

The *Configuration and Management Control* (Fig. 2) component manages and controls the configuration of all parameters of the MAC sublayer and the internal parameters of the *Mediator Layer*, respecting application requirements, resource limitations, and environment restrictions. Such requirements, limitations, and restrictions should be defined in an profile that is utilised to adjust the aforementioned parameters. Autonomous methods may be utilized by the *Configuration and Management Control* components, making configuration procedures self-adaptive, self-managed, and self-controlled.

IV. Building *Mediator Layer* components

In order to illustrate how to improve the dependability and timeliness properties of a standard MAC sublayer, we select a set of key mechanisms that needs to be included in the *Mediator Layer* for that purpose.

The first mechanism we address introduces a minor extension in the standard frame check sequence (FCS) mechanism. The native FCS mechanism silently discards every frame received with errors, meaning omission errors will not be perceived, nor monitored nor accounted for. Conversely, the extension specified in Algorithm 1 provides a management indication of the status of the received frame to the *Mediator Layer*, even if the frame was received with errors. The management indication highlighted in line 15 signals: the instant the frame was received, represented by the variable $time\_stamp$, as obtained in line 6; the frame header, represented by the variable $frame\_header$, which is extracted in line 7; and the FCS error status represented by the variable fcs_error. This simple extension of the native standard mechanism enriches the monitoring capabilities of the *Mediator Layer*.

The second mechanism we address takes profit of the FCS extension mechanism to account for omission errors. The accounting of omission errors is expressed as a pseudo-code presented in Algorithm 2. The status of each frame received by a node is signalled through a MAC management indication in line 7. If a frame was received with errors, the number of consecutive omission errors, O_{degree}, is incremented by one (line 9). Otherwise, i.e., whenever a frame is received without errors, the value of O_{degree} is cleared (line 11). Should the omission degree bound, k, be exceeded, a management indication is provided (line 14). This is an indication that the communication channel in use has failed and that a channel switch action should be issued to MAC sublayer management entities.

Algorithm 1 Extending the FCS mechanism

```
 1: Initialization phase.
 2: $fcs\_error$ ←*false*;
 3: Begin.
 4: **loop**
 5:     **when** *Channel.indication(frame)* **do**
 6:         $\boxed{time\_stamp \leftarrow \textit{MLA.get.time()};}$
 7:         $\boxed{frame\_header \leftarrow \textit{MAC.get.header(frame)};}$
 8:         **if** *MAC.FCS.check(frame) is OK* **then**
 9:             $fcs\_error$ ← *false*;
10:             *MAC.indication(frame)*;
11:         **else**
12:             $fcs\_error$ ← *true*;
13:             *MAC.discard(frame)*;
14:         **end if**
15:         $\boxed{\textit{MAC.Mgmt.indication(time\_stamp, frame\_header, } fcs\_error\textit{)};}$
16:     **end when**
17: **end loop**
18: End.
```

Algorithm 2 Accounting local omission errors

```
 1: Initialization phase.
 2: $O_{degree} \leftarrow 0$;
 3: $k$ ← *The value of the omission degree bound depends on environment
    conditions. The default value utilized within the IEEE 802.15.4 standard
    is* $k = 3$.
 4: $current\_channel$ ← *It represents which is the current channel in use.*
 5: Begin.
 6: **loop**
 7:     **when** *MAC.Mgmt.indication(time_stamp, frame_header,* $fcs\_error$*)*
        **do**
 8:         **if** $fcs\_error = \text{true}$ **then**
 9:             $O_{degree} \leftarrow O_{degree} + 1$;
10:         **else**
11:             $O_{degree} \leftarrow 0$;
12:         **end if**
13:         **if** $O_{degree} > k$ **then**
14:             *MLA.Mgmt.indication(time_stamp,current_channel,*
                $O_{degree}$*_exceeds_k)*
15:         **end if**
16:     **end when**
17: **end loop**
18: End.
```

The real-time operation of nodes within the WnS is also monitored by a node failure detection service, which is part of the *Real-time Protocol Suite* component. The node failure detection is presented in Algorithm 3. This algorithm resides within every node of a WnS, establishing a distributed and decentralized node failure detection service.

Each node locally accounts omissions from other nodes of the WnS as follows: an indication representing the status of a received frame is detected (line 6). The source node index of the received frame is extracted from the $time\_stamp$ and $frame\_header$ variables (line 7). If a valid node index is found (e.g., the node source identifier in the frame header matches the foreseen time slot represented here by the $time\_stamp$ variable). If the frame was received with errors, the omission degree of this node is incremented by one (line 10), otherwise it is reseted to zero (line 12). When the omission degree of a given source node of a received frame exceeds k, a local indication is generated by the node failure detection service (line 15), which can be utilized by a membership

Algorithm 3 Node failure detection

* The detection of a node's crash is intentionally ommitted

```
 1: Initialization phase.
 2: $k \leftarrow 3$;
 3: $node_{index} \leftarrow$ It represents the index of a node.
 4: Begin.
 5: loop
 6:   when MAC.Mgmt.indication($time\_stamp, frame\_header, fcs\_error$) do
 7:     $node_{index} \leftarrow$ MLA.Mgmt.getNode($time\_stamp, frame\_header$);
 8:     if $node_{index}$ is valid then
 9:       if $fcs\_error$ = true then
10:         $O_{degree}[node_{index}] \leftarrow O_{degree}[node_{index}] + 1$;
11:       else
12:         $O_{degree}[node_{index}] \leftarrow 0$;
13:       end if
14:       if $O_{degree}[node_{index}] > k$ then
15:         MLA.FD.indication($time\_stamp, current\_channel,$
            $node_{index}, O_{degree}[node_{index}]\_exceeds\_k$)
16:       end if
17:     end if
18:   end when
19: end loop
20: End.
```

service to help the update of the view that represents the active nodes within the WnS.

As a proof-of-concept the *Mediator Layer* has been implemented and incorporated within both NS2 simulator [16] and a COTS platform [17], using the IEEE 802.15.4 standard as a case study. The results achieved so far have shown that these mechanisms allow to achieve optimal latencies with respect to the detection of channel failures (Algorithm 2) and node failures (Algorithm 3). Any violation of the omission degree bound is also detected as soon as it occurs.

This happens because frame omissions are monitored and detected at the lowest level of communications (Algorithm 1). The effectiveness of this approach is illustrated in Fig. 3. In Fig. 3 the blue color represents the number of frames (621) received without any error (i.e., indicated with or without the presence of *Mediator Layer*), while the orange color represents the number of frames (270) received with errors, and detected by *Mediator Layer*.

V. Conclusion and Future Work

This paper presented an extensible component layer dubbed *Mediator Layer*, which enhances the timeliness and dependability properties of wireless communications. A dependable and timely service at lowest level of the network protocol stack helps the higher level protocol designers to keep their solutions as simple as possible, using the easy-to-design building block approach enabled by the *Mediator Layer*. Our standard-compliant approach has been implemented and incorporated within the NS2 network simulator and a COTS platform.

Future directions include, but are not limited to: adding protocols to the *Real-Time Protocol Suite* component to cover the requirements imposed by real-time environments; finishing the implementation and incorporation of the *Mediator Layer* within the NS2 network simulator and the COTS platform; performing analyses and evaluation of the *Mediator Layer*

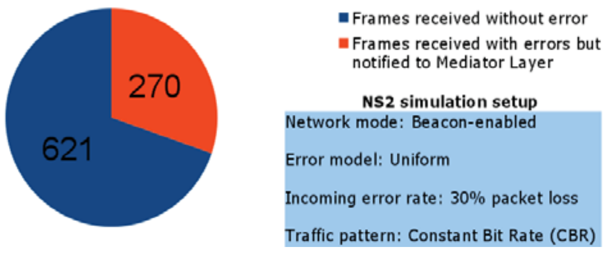


Fig. 3: Results obtained with the extension of the FCS mechanism implemented within NS2 simulator

approach in such both simulator and COTS platform face to different error conditions and temporal requirements of environments and applications; and defining relevant real-time metrics to evaluate the wireless communications with regard to application requirements and environment restrictions.

References

[1] T. Stone, R. Alena, J. Baldwin, and P. Wilson, "A viable COTS based wireless architecture for spacecraft avionics," in *IEEE Aerospace Conference*, 2012, pp. 1–11.

[2] J. Åkerberg, M. Gidlund, and M. Björkman, "Future research challenges in wireless sensor and actuator networks targeting industrial automation," in *9th IEEE INDIN*, July 2011.

[3] A. Sahoo and P. Baronia, "An energy efficient MAC in WSN to provide delay guarantee," in *15th IEEE LANMAN*, June 2007.

[4] I. Aad, P. Hofmann, L. Loyola, F. Riaz, and J. Widmer, "E-MAC: Self-organizing 802.11-compatible MAC with elastic real-time scheduling," in *IEEE MASS*, October 2007.

[5] E. E-López, J. V-Alonso, A. M-Sala, J. G-Haro, P. P-Mariño, and M. Delgado, "A WSN MAC protocol for real-time applications," *Personal Ubiquitous Computing Journal*, January 2008.

[6] P. Bartolomeu, J. Ferreira, and J. Fonseca, "Enforcing flexibility in real-time wireless communications: A bandjacking enabled protocol," in *IEEE ETFA*, September 2009.

[7] X.-Y. Shuai and Z.-C. Zhang, "Research of real-time wireless networks control system MAC protocol," *Journal of Networks*, April 2010.

[8] T. Zhou, H. Sharif, M. Hempel, P. Mahasukhon, W. Wang, and T. Ma, "A novel adaptive distributed cooperative relaying MAC protocol for vehicular networks," *IEEE Journal on Sel. Areas in Comm.*, Jan 2011.

[9] M. Sha, G. Hackmann, and C. Lu, "Arch: Practical channel hopping for reliable home-area sensor networks," in *17th IEEE RTAS*, April 2011.

[10] X. Zhu, S. Han, P.-C. Huang, A. Mok, and D. Chen, "MBStar: A real-time communication protocol for wireless body area networks," in *23rd ECRTS*, July 2011.

[11] Yu-Kai, Ai-Chun, and Hui-Nien, "An adaptive GTS allocation scheme for IEEE 802.15.4," *IEEE Trans. on Parallel and Distributed Systems*, May 2008.

[12] M. Hameed, H. Trsek, O. Graeser, and J. Jasperneite, "Performance investigation and optimization of IEEE 802.15.4 for industrial wireless sensor networks," in *IEEE ETFA*, September 2008.

[13] A. Koubâa, A. Cunha, M. Alves, and E. Tovar, "i-GAME: An implicit GTS allocation mechanism in IEEE 802.15.4, theory and practice," *Springer Real-Time Systems Journal*, August 2008.

[14] F. Kuhn, N. Lynch, and C. Newport, "The abstract MAC layer," in *23rd DISC*, September 2009.

[15] J. L. R. Souza and J. Rufino, "An approach to enhance the timeliness of wireless communications," in *5th UBICOMM*, Lisbon, 2011.

[16] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*. Springer, 2009.

[17] ATMEL, *ATMEL AVR2025: IEEE 802.15.4 MAC Software Package - User guide*, ATMEL Coorporation, May 2012.

[18] J. L. R. Souza and J. Rufino, "Characterization of inaccessibility in wireless networks - a case study on IEEE 802.15.4 standard," in *3th IFIP IESS*, September 2009.

A.1.5 Analysing and Reducing Network Inaccessibility in IEEE 802.15.4 Wireless Communications

“Analysing and Reducing Network Inaccessibility in IEEE 802.15.4 Wireless Communications”, J. L. R. Souza and J. Rufino. In Proceedings of the 38th IEEE Conference on Local Computer Networks (LCN 2013), October 2013.

This page is intentionally left blank.

Analysing and Reducing Network Inaccessibility in IEEE 802.15.4 Wireless Communications

Jeferson L. R. Souza and José Rufino
University of Lisboa - Faculty of Sciences
LaSIGE - Navigators Research Team
Email(s): jsouza@lasige.di.fc.ul.pt, ruf@di.fc.ul.pt

***Abstract*—Network inaccessibility is a temporal issue derived from the presence of faults affecting the communication services provided by the medium access control (MAC) sublayer. The occurrence of network inaccessibility represents temporary "communication blackouts", which prevent communications to be performed and may imply disruptions of network operation, therefore compromising the dependability and timeliness of communications. This paper uses an analytical model accounting for network inaccessibility periods in wireless sensor and actuator networks, presenting the IEEE 802.15.4 standard as a case study. The analytical model is then used to derive a set of simple, yet quite effective policies to reduce the durations of the periods of network inaccessibility. The effectiveness of these policies can be evaluated using a tool based on the analytical model, which is being integrated in the NS-2 simulator for validation. Reducing network inaccessibility is a crucial step to enable the use of wireless networking technologies in real-time settings.**

***Index Terms*—wireless sensor and actuator networks, real-time, timeliness, dependability, network inaccessibility.**

I. Introduction

There is a strong demand for the use of wireless sensor and actuator networks (WSANs) on settings with temporal restrictions, where real-time communications are fundamental. In many environments, such as in complex embedded systems aboard autonomous aerial and terrestrial vehicles, WSANs are the perfect communication technology to permit the balance between the needs of real-time networks and the reduction of system size, weight, and power consumption (SWaP), altogether without lessen timeliness and dependability guarantees [12].

A lot of work has been presented, proposing new protocols [1], [3], [4], [13]–[15], [20], [21], modifications on the existent standards [6], [9], [19], and abstract models [10] trying to enhance the real-time guarantees and reliability of wireless communications.

These works, built on analysis focused on timeliness, pay no or little attention to dependability aspects of communications. The fault model (when presented) only considers faults on data domain, disregarding the disruptive effect that faults may have on the medium access control (MAC) sublayer operation and its services. However, such faults may induce temporary "communication blackouts", which lead to the execution of additional procedures to reestablish normal MAC protocol operation. Meanwhile, the MAC protocol is prevented from providing service and the network is inaccessible. When a network inaccessibility incident occurs communications cannot be performed for a period of time. One key point is that the periods of network inaccessibility may have a duration much higher than the normal worst case network access delay. As a consequence, the overall timeliness and dependability properties of the system may be at risk, being compromised at communication service level.

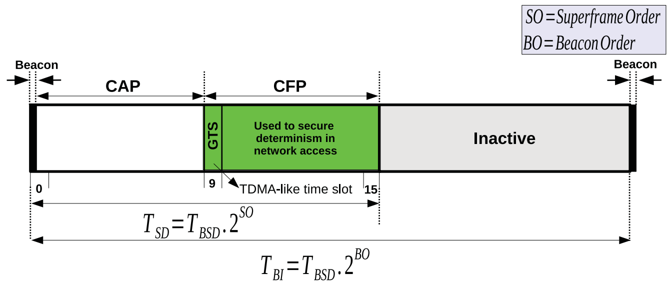
A solution to the problem of controlling network inaccessibility is needed to secure an effective and efficient real-time wireless communications support. Defining a strategy for network inaccessibility reduction is not only a significant but also an essential step towards that goal. Therefore, motivated by a pressing need to attenuate the negative effects caused by network inaccessibility, this paper presents and discusses an analytical model and a set of simple, yet quite effective policies to reduce the duration of network inaccessibility within IEEE 802.15.4 wireless communications [7].

To present our advances the paper is organized as follows: Section II presents an overview of the IEEE 802.15.4 standard. Section III introduces network inaccessibility and presents the analytical model characterising network inaccessibility on IEEE 802.15.4 wireless networks. Section V explains the policies defined to reduce the duration of network inaccessibility on IEEE 802.15.4 communications. Section VI briefly presents a tool for evaluating inaccessibility durations in IEEE 802.15.4 and its preliminary validation using the NS-2 simulator. Section VII analyses the impact of network inaccessibility reduction policies on the timeliness of the standard IEEE 802.15.4 MAC sublayer operation. Finally, section VIII draws the conclusion, and some future work.

II. IEEE 802.15.4 - Overview

The IEEE 802.15.4 [7] has two operation modes dubbed nonbeacon-enabled and beacon-enabled. This paper is focused on the beacon-enabled mode, designed to support traffic with temporal restrictions. In a beacon-enabled mode there is a

This work was partially supported: by the EC, through project IST-FP7-STREP-288195 (KARYON); by FCT/DAAD, through the transnational cooperation project PROPHECY; and by FCT, through the project PTDC/EEI-SCR/3200/2012 (READAPT), the Multiannual Funding Program, and the Individual Doctoral Grant SFRH/BD/45270/2008.



SO = Superframe Order
BO = Beacon Order
Beacon
Beacon
CAP
CFP
GTS
Used to secure determinism in network access
Inactive
0
9
TDMA-like time slot
15
$T_{SD} = T_{BSD} . 2^{SO}$
$T_{BI} = T_{BSD} . 2^{BO}$

Fig. 1: Superframe structure of the IEEE 802.15.4 in beacon-enabled mode

coordinator node that manages and controls network access. The default superframe structure [7], represented in Fig. 1, is utilised by the coordinator to control the access to the network. The duration of a superframe is calculated utilising a constant that defines the minimum (also known as base) superframe duration, $\mathcal{T}_{BSD}$, and a beacon order exponent, BO, which is utilised to determine the actual time interval between consecutive beacon frames, $\mathcal{T}_{BI}$, as given by:

$$\mathcal{T}_{BI} = \mathcal{T}_{BSD} \, . \, 2^{BO} \tag{1}$$

As illustrated by Fig. 1, the default superframe structure has a contention access period (CAP), where nodes compete in equal condition to access the network in a non-real-time manner; a contention free period (CFP), where nodes access the network within exclusive time slots (GTS, the Guaranteed Time Slots) supporting real-time traffic in a similar manner of time division multiple access (TDMA) approaches; and an optional inactive period (IP), where nodes may enter in a power-save mode. Several time slots may be allocated to a node, for exclusive and contention-free network access.

The CAP and CFP together represent the active portion of the superframe structure, which has a duration given by:

$$\mathcal{T}_{SD} = \mathcal{T}_{BSD} \, . \, 2^{SO} \tag{2}$$

where SO is the superframe order exponent that defines the duration of this active portion. If $SO = BO$ there is no IP within the superframe.

III. Analysing Network inaccessibility in IEEE 802.15.4 wireless communications

Network inaccessibility is characterised by a temporary lack of network access due to disturbances on MAC sublayer operation. Inaccessibility incidents need to be controlled to enforce real-time operation, meaning: one must ensure that such events have limited duration and rates; violation of such bounds leads to the permanent failure of the network.

The definition of an analytical model, thoroughly characterising network inaccessibility incidents and their durations for the IEEE 802.15.4 MAC protocol has been introduced in [18]. For the purpose of self-completeness, we summarise next some details of such analysis. For the relevant scenarios, we show how the corresponding periods of network inaccessibility are derived, being their worst case durations represented by the superscript (^{wc}).

The beacon frame controls the access to the network, and its reception is essential to maintain all the nodes synchronised within the different periods of the superframe structure. If a beacon frame is not correctly received, a network inaccessibility incident occurs. Thus, a **single beacon frame loss** occurs when only one beacon is lost:

$$\mathcal{T}_{ina \leftarrow sbfl} = \mathcal{T}_{BSD} . (2^{BO} + 1) \tag{3}$$

The value of $\mathcal{T}^{wc}_{ina \leftarrow sbfl}$ is equivalent to $\mathcal{T}_{BI}$ plus an extra $\mathcal{T}_{BSD}$ margin, accommodating the clock skew between a node and its coordinator. The **multiple beacon frame loss** occurs when multiple and consecutive beacons are lost:

$$\mathcal{T}^{wc}_{ina \leftarrow mbfl} = nrLost . \mathcal{T}_{BSD} . \left(2^{BO} + 1\right) \tag{4}$$

where a correct beacon frame is successfully received after the loss of *nrLost* beacons. The **synchronisation loss** is a special case of the *multiple beacon frame loss* scenario where after the loss of *nrLost* beacons, the next beacon is also lost. A node loses synchronisation with the network coordinator after a period given by:

$$\mathcal{T}_{ina \leftarrow nosync} = nrLost . \mathcal{T}_{BSD} . \left(2^{BO} + 1\right) \tag{5}$$

To recover from a synchronisation loss, two different strategies were identified in the standard specification [7]. Each individual node chooses the recovery strategy that it should use. We assume that if a data/control frame was received during the last beacon interval, the node assumes an *orphan* status; otherwise, a *re-association* procedure should be carried out. In both recovery strategies, the node looks for a coordinator in a given set of logical channels[1]. After the channel scan, a coordinator realignment or an association procedure is performed within the *orphan* and *re-association* scenarios, respectively. Thus, the worst case duration of network inaccessibility for the **orphan** scenario is given by:

$$\begin{aligned} \mathcal{T}^{wc}_{ina \leftarrow orphan} = \; & \mathcal{T}_{ina \leftarrow nosync} + \\ & \sum_{j=1}^{nrchannels} \left[\mathcal{T}^{wc}_{MAC}(Orphan) + nrWait \, . \, \mathcal{T}_{BSD} \right] \\ & + \mathcal{T}_{MLA}(Realign) + \mathcal{T}^{wc}_{MAC\_ack}(Realign) \end{aligned} \tag{6}$$

where: *nrchannels*, represents the number of logical channels to be scanned; *nrWait*, defines the waiting period for a beacon frame in each channel scan; $\mathcal{T}_{MAC\_ack}(frame)$ and $\mathcal{T}_{MAC}(frame)$ represent the delay from request to confirmation of a MAC frame transmission with and without acknowl-

[1] A logical channel is an abstract representation of a radio frequency (RF) channel utilised by the MAC layer to perform its network communications.

edgement; the reference to $\mathcal{T}_{MLA}(action)$ represents the time needed to perform the specified action at the MAC management layer. Without loss of generality, an uniform value of $\mathcal{T}_{MLA}(action) = \frac{1}{10}.\mathcal{T}_{BI}$ is assumed for the duration of each MAC management layer action.

In the **re-association** scenario, a node sends a beacon request ($Beacon\_R$) and waits for the reception of a beacon; upon beacon reception, recovery proceeds with an association ($Assoc\_R$) procedure and the extraction (Ext_R) of control information from the network coordinator:

$$\begin{aligned}\mathcal{T}^{wc}_{ina\leftarrow reAssoc} = \mathcal{T}_{ina\leftarrow nosync} + \\ \sum_{j=1}^{nrchannels} [\mathcal{T}^{wc}_{MAC}(Beacon\_R) + nrWait.\mathcal{T}_{BSD}] + \\ \mathcal{T}_{MLA}(Beacon) + \mathcal{T}^{wc}_{MAC\_ack}(Assoc\_R) + \\ \mathcal{T}_{MLA}(Assoc) + \mathcal{T}^{wc}_{MAC\_ack}(Ext\_R)\end{aligned} \tag{7}$$

Finally, a **coordinator conflict** occurs when more than one coordinator is active within the same network. By default, each network has an identifier, the *source identifier*, which identifies the network uniquely and is used by the coordinator in beacon transmissions. If some other (possibly old) coordinator enters the network operational space, e.g., after having been away from some period of time, the network may have two different coordinators transmitting beacons with the same *source identifier*. To solve such conflict, the current coordinator performs a search within a set of specified logical channels. After the scan in all logical channels, a fresh *source identifier* is selected and, if necessary, a MAC coordinator realignment command is broadcast:

$$\begin{aligned}\mathcal{T}^{wc}_{ina\leftarrow Conflict} = \mathcal{T}_{MLA}(Conflict) + \\ \sum_{j=1}^{nrchannels} [\mathcal{T}^{wc}_{MAC}(Beacon\_R) + nrWait.\mathcal{T}_{BSD}] \\ + \mathcal{T}_{MLA}(Realign) + \mathcal{T}^{wc}_{MAC}(Realign)\end{aligned} \tag{8}$$

IV. Network parametrisation for real-time operation

The first step towards real-time network operation may simply emerge from the fine-adjustment of a relevant set of network configuration parameters. This is formalised by the following proposition:

Proposition 1: Each node accesses the network in a bounded and known time interval of, at most, $\mathcal{T}_{ac}$.

The guarantees provided by this proposition depends on the network technology, its characteristics and, ultimately, on network configuration parameters. The value of $\mathcal{T}_{ac}$ simply accounts for the raw network access delay observed at MAC sublayer before starting a frame transmission; it does not account for the frame transmission time and it does not include any buffering/queueing effects nor any delays associated with possible frame retransmissions.

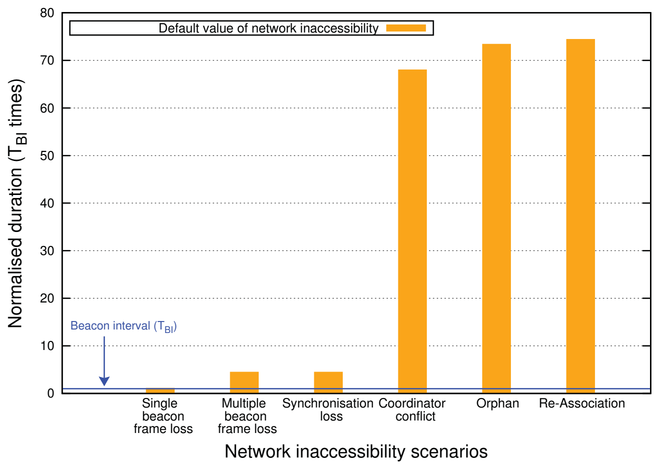


Fig. 2: The default values of IEEE 802.15.4 network inaccessibility durations normalised by, and compared with, $\mathcal{T}_{BI}$ ($\mathcal{T}_{BI} = 123ms$) ($nrWait = 32$; $nrChannels = 16$).

A. IEEE 802.15.4 networks

For the particular case of IEEE 802.15.4 networks operating in beacon-enabled mode, a bounded and known $\mathcal{T}_{ac}$ is secured given the contention-free network access provided by GTS within a period equal to $\mathcal{T}_{BI}$. Therefore:

$$\mathcal{T}_{ac} = \mathcal{T}_{BI} \tag{9}$$

For the remainder of our analysis we use as an example $\mathcal{T}_{BI} = 123ms$ ($BO = 3$; $\mathcal{T}_{BSD} = 15.36ms$), which can provide a reasonable beacon interval for periodic real-time transmissions, still allowing the use of reliable unicast data transmissions (with $SO = BO = 3$, i.e, no IP).

B. IEEE 802.15.4 network inaccessibility

The durations of network inaccessibility incidents defined by the analytical model of IEEE 802.15.4 discussed in Section III are inscribed in Fig. 2, for a standard network configuration ($nrWait = 32$; $nrChannels = 16$). The (real) value of $\mathcal{T}_{BI}$ is used to normalise the duration of network inaccessibility events, as also shown in Fig. 2. Network inaccessibility incidents have very different durations, with some of them much longer than $\mathcal{T}_{BI}$. Long and highly variable periods of network inaccessibility are a source of disruption and unpredictability in network operation.

V. Reducing network inaccessibility in IEEE 802.15.4 wireless communications

Reducing the network inaccessibility in IEEE 802.15.4 wireless communications is an essential step to enhance network communication properties such as dependability, timeliness, and predictability, which enforce real-time operation. The network characterisation in Section III is crucial to understand how to reduce (or even eliminate) the longest periods of network inaccessibility.

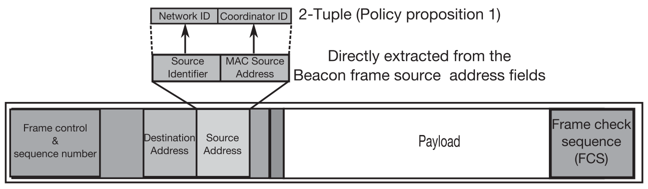

Network ID
Coordinator ID
2-Tuple (Policy proposition 1)
Source Identifier
MAC Source Address
Directly extracted from the Beacon frame source address fields
Frame control & sequence number
Destination Address
Source Address
Payload
Frame check sequence (FCS)

Fig. 3: Representation of a beacon frame.

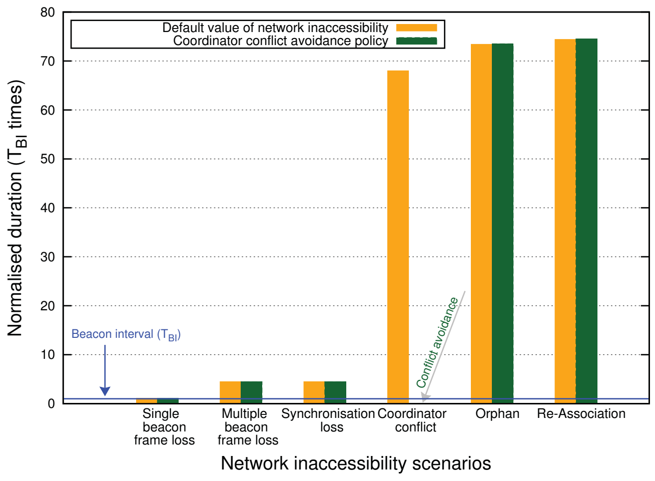


Fig. 4: Impact of the coordinator conflict avoidance policy in the IEEE 802.15.4 network inaccessibility

A. Coordinator conflict avoidance policy

The *coordinator conflict* scenario occurs when two or more coordinators transmit beacons with the same (unique) *source identifier*. A conflict resolution strategy should be triggered after the detection of such scenario (as specified in the IEEE 802.15.4 standard). The duration of network inaccessibility is characterised by the time spent to perform the coordinator conflict resolution. To avoid the coordinator conflict we establish the following proposition:

***Policy proposition** 1:* Each node must use a 2-Tuple $\langle networkID, coordinatorID\rangle$ as a unique compound network identifier, avoiding then coordinator conflicts.

The following check procedure is applied to each received beacon: *If the 2-Tuple $\langle networkID, coordinatorID\rangle$ inside the received beacon does not match the 2-Tuple $\langle networkID, coordinatorID\rangle$ of the network, the received beacon is discarded.*

Since the *coordinatorID* is directly extracted from the node MAC source address (Fig. 3) and this is unique for each node, the resulting 2-Tuple $\langle networkID, coordinatorID\rangle$ is also unique, for a given network coordinator. No coordinator conflict will ever occur (Fig. 4).

The 2-Tuple beacon check procedure extends and replaces the native IEEE 802.15.4 operation and can be made compatible with the standard specification. This procedure is not hard to implement in modern wireless communication platform [2]. Finally, since the 2-Tuple $\langle networkID, coordinatorID\rangle$ is directly extracted from existing fields in the standard beacon frame, no modification to the standard is required and no overhead is added to MAC sublayer operation.

The presence of malicious entities, which may cause an intentional coordinator conflict problem, are not addressed in this paper. Malicious entities need to be handled with additional techniques to overcome the hazards that they may cause, and will be addressed in a future work.

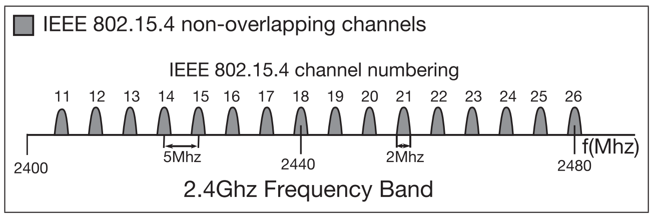


Fig. 5: IEEE 802.15.4 channels within 2.4Ghz frequency band

B. Channel utilisation awareness policy

To reduce the network inaccessibility periods resulting from the *orphan* and *re-association* scenarios, we design a policy that exploits the knowledge of channel utilisation by the network coordinator to reduce the time spent in logical channel scan operations.

In the channel utilisation awareness policy each node is “aware” of the number of logical channels available in the network to search for the presence of the network coordinator, being represented by the following proposition:

***Policy proposition** 2:* Each node is aware of the logical channel utilisation within its associated network, restricting the search for the network coordinator in some $C_a := \{c \mid c \in C \wedge C \subset A\}$, where C_a is the search channel set and A is the set of the available logical channels, being $0 < \#C_a < \#A$.

The nodes use a subset, C_a, of the available logical channels set, A, to confine its channel utilisation scope. Each node is able to search and find the network coordinator within that confined channel search space, reducing then the amount of time needed to find the network coordinator. Restricting the number of logical channels in the channel search space has no impact on network throughput, since only one logical channel is in use at a time. In fact, in the presence of noisy channels, selecting a restricted set of logical channels exhibiting lower error rates, may actually contribute to a potential increase of channel effective throughput, and to reduce the amount of energy utilized to complete a frame transmission successfully.

In the particular case of IEEE 802.15.4 networks there is no mutual channel interference since all the $\#A = 16$ channels are non-overlapping (Fig. 5). Thus, we can choose an arbitrary number of logical channels to include in subset C_a. Figure 6 illustrates the impact of our channel utilisation awareness policy in a IEEE 802.15.4 network, where the value of $\#C_a$ is successively reduced to half, until an optimal $\#C_a = 2$ value is reached. A value of $\#C_a = 2$ minimises the duration of network inaccessibility for the *orphan* and *re-association* sce-

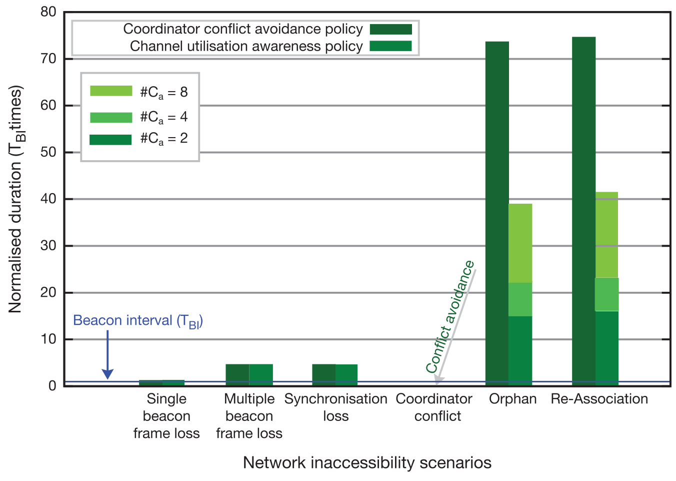


Fig. 6: Impact of the channel utilisation awareness policy in the IEEE 802.15.4 network inaccessibility

narios, while preserves the dependability property of channel diversity.

C. Network dependability awareness policy

The frame check sequence (FCS) is a fundamental mechanism to verify the integrity of a received frame, and therefore to detect accidental errors with an appropriate coverage. When a frame is received with errors, such frame is automatically discarded and nothing is signalised upward (FCS default operation). Only the reception of frames without errors are signalised. The lack of a management notification for such discarded frames prevents the MAC management entities to detect and account for omission errors.

Algorithm 1 presents an extension to the FCS default operation, which introduces a management signalisation to notify the status of the received frame, even if the frame contains errors and must be discarded. Two fundamental additions were proposed: the extraction of the frame header (line 6); and the notification of the received frame status to the MAC management entities (line 14), being such status represented by the fcs_error variable. The information within frame header (e.g., source address) may be utilised to identify if the sender is the owner of a *time_slot*, in case of transmissions within CFP.

The improvements proposed in Algorithm 1 are simple, efficient, and not hard to implement off-the-shelf using modern commercially available wireless communication platforms [2].

The notification of the status of the received frame, which is proposed in Algorithm 1, also allows to built accurate error detection and accountability solutions. In particular, an error that destroys a frame is transformed into an omission. Thus, Algorithm 2 presents a simple mechanism to account for channel omissions. The number of consecutive omissions observed by a node within the current logical channel is represented by O_d, the omission degree, in line 2. The value of O_d is cleared every time a frame is received without

Algorithm 1 Extending frame check sequence (FCS) mechanism

```
 1: Initialisation phase.
 2: $fcs\_error \leftarrow false$;
 3: Begin.
 4: loop
 5:     when Channel.indication(frame) do
 6:         $\boxed{frame\_header \leftarrow MAC.get.header(frame);}$
 7:         if MAC.FCS.check(frame) is OK then
 8:             $fcs\_error \leftarrow false$;
 9:             MAC.indication(frame);
10:         else
11:             $fcs\_error \leftarrow true$;
12:             MAC.frame.discard(frame);
13:         end if
14:         $\boxed{MAC.Mgmt.indication(time\_slot,\ frame\_header,\ fcs\_error);}$
15:     end when
16: end loop
17: End.
```

Algorithm 2 Omission degree monitoring

```
 1: Initialisation phase.
 2: $O_d \leftarrow 0$;
 3: $k \leftarrow$ The value of the omission degree bound, k, is dependent of the MAC
    layer characteristics and of the network environment. The IEEE 802.15.4
    standard indirectly defines $k \leftarrow 3$;
 4: Begin.
 5: loop
 6:     when MAC.Mgmt.indication(time_slot, frame_header, fcs_error) do
 7:         if $fcs\_error$ is true then
 8:             $O_d \leftarrow O_d + 1$;
 9:         else if $fcs\_error$ is false then
10:             $O_d \leftarrow 0$;
11:         end if
12:     end when
13:     if $O_d > k$ then
14:         $\boxed{MLA.Mgmt.indication(logical\_channel,\ O_d\_exceeds\_k);}$
15:     end if
16: end loop
17: End.
```

errors (line 10). When a frame is received, and an error is detected, the procedure increments O_d (line 8). If O_d exceeds an omission degree bound, k, the MAC management entities are notified (line 14). This may be an indication of a heavily disturbed logical channel or it may be a result of the underestimation of the omission degree bound. In any case, the logical channel should be considered failed. Considering only accidental transient faults, the omission degree of a logical channel can be bounded by the following property: *in a known time interval, omission failures may occur in at most k transmissions*. The value of omission degree bound depends on the network error characteristics and on the environment conditions [5]. The IEEE 802.15.4 standard indirectly defines a fixed value of $k = 3$ in its error handling mechanisms. Having the ability to determine the true omission degree bound of a given logical channel is a worthwhile feature, due to the nature of wireless communications, highly susceptible to multiple external disturbances such as signal attenuation, noise and electromagnetic interferences from other signal sources, and multipath propagation interference due to obstacles in the

IEEE 802.15.4 Parameter	IEEE 802.15.4 Standard Configuration	Dependable Adaptation
nrLost	4	$k+1$
nrWait	32	$(k+1).2^{BO}$

TABLE I: Network parametrisation in function of dependability metrics

communication path.

Our network dependability awareness policy is now formulated by the following proposition:

***Policy proposition** 3:* Each node is aware of the dependability characteristics of the network, described by a set of relevant metrics.

The omission degree bound is one of such dependability metrics, but others may be introduced. For example, slight modifications to Algorithm 2 will allow to assess: the average value of the omission degree, the number of omission degree bound violations within a given period and other statistics.

To illustrate how the dependability parameters can be used to improve the characteristics of wireless communications, we use the omission degree bound k to dynamically define some MAC protocol parameters relevant for IEEE 802.15.4 operation, as specified in Table I, thus opening room for the use of (self-)adaptation techniques, e.g., to cope with varying environment conditions. This may be advantageous for decreasing the network inaccessibility durations associated to the *multiple beacon frame loss* and *synchronisation loss* scenarios.

D. Logical channel diversity policy

The violation of the channel omission degree bound, locally-perceived by each node, should be interpreted as a failure indication. To restore communication one must resort to the following propositions:

***Policy proposition** 4:* There are multiple and redundant logical channels, though only one is active at a time.

***Policy proposition** 5:* Every frame is transmitted only in the active logical channel.

***Policy proposition** 6:* In the presence of faults which may lead a logical channel to an incorrect state, a node may switch to a different logical channel.

In particular, one must take advantage of the use of redundant logical channels, specifying the following procedure: *when a node (including the network coordinator) detects that a given logical channel O_d exceeds k, a node switches to another logical channel.* To avoid the occurrence of a permanent physical partitioning of the network, the same channel switching sequence is used by all nodes, defining a deterministic order utilised to switch from one logical channel to another. Furthermore, we assume: *each node must transmit at least one (heartbeat) frame during its allocated GTS to signal node liveness.* As we are also interested to

Algorithm 3 Logical channel diversity procedure - COORDINATOR

```
1: Initialisation phase.
2: nrAssocNodes ← 0;
3: channel_idle_status ← true;
4: Begin.
5: loop
6:     when MAC.Mgmt.request(Beacon) do
7:         nrAssocNodes ← MLA.Mgmt.get(NR_ASSOC_NODES);
8:         if channel_idle_status is true ∧ nrAssocNodes > 0 then
9:             MLA.Mgmt.request(Change_Channel);
10:            MLA.Mgmt.request(RESET_NR_ASSOC_NODES);
11:        end if
12:        channel_idle_status ← true;
13:    end when
14:    when MAC.indication(frame) do
15:        channel_idle_status ← false;
16:    end when;
17:    when MLA.Mgmt.indication(logical_channel, O_d_exceeds_k) do
18:        MLA.Mgmt.request(Change_Channel);
19:        MLA.Mgmt.request(RESET_NR_ASSOC_NODES);
20:    end when
21: end loop
22: End.
```

reduce network inaccessibility durations in benefit of a real-time network operation, only nodes with allocated GTS are monitored.

Upon channel switch it may happen that a node detects no traffic activity because it is the only node in that logical channel. The standard MAC protocol of non network coordinator nodes has mechanisms to detect such situations, signalled to MAC management entities through a *synchronisation loss* indication. The MAC protocol of the network coordinator does not have such capability by default. Thus, we enhance the coordinator to detect channel idleness, as specified in Algorithm 3, taking advantage of node liveness signalisation within GTS slots.

Algorithm 3 describes the execution of the logical channel diversity policy in the coordinator node. When the network coordinator is started the variable utilised to store the number of associated nodes, *nrAssocNodes*, and the channel idle status, *channel_idle_status*, are initialised with 0 and $true$, respectively (lines 2 and 3). A channel switch operation is triggered by two different situations. The first channel switching scenario, described by lines 17 to 20, is a direct consequence of a logical channel failure indication, as provided by the signalling that the value of O_d for the current logical channel has exceeded k (line 17); after logical channel switching (line 18) the list of nodes associated with the network coordinator is cleared (line 19).

The second channel switching situation is more complex and involves the results of monitoring logic channel activity during the last beacon interval. The logic channel monitoring actions are in fact quite simple, being described by lines 14 to 16: upon reception of a correct frame indication (line 14), the channel idle status variable is set to $false$ (line 15). The network coordinator monitors logical channel traffic between any two consecutive beacon transmissions: if a frame is

Algorithm 4 Logical channel diversity procedure - NON COORDINATOR

```
1:  Initialisation phase.
2:  $receivedFrame \leftarrow false$;
3:  Begin.
4:  loop
5:      when MLA.Mgmt.indication(logical_channel, $O_d$_exceeds_k) do
6:          MLA.Mgmt.request(Change_Channel);
7:      end when
8:      when MAC.Mgmt.indication(SYNC_LOSS) do
9:          $receivedFrame \leftarrow$ MLA.Mgmt.get(FRAME_FROM_COORD);
10:         if receivedFrame is true then
11:             MLA.Mgmt.request(ORPHAN,$\#C_a = 1$)
12:         else
13:             MLA.Mgmt.request(RE_ASSOCIATION,$\#C_a = 2$);
14:         end if
15:     end when
16: end loop
17: End.
```

correctly received, there is no reason, *per se*, to perform a channel switch; if no traffic at all is correctly received within that period and the network coordinator has, at least, one node associated to it (i.e., $nrAssociated > 0$), the logical channel is considered idle (line 8) and the coordinator switches to the next logical channel using a pre-defined channel hopping sequence (line 9); the list of nodes associated with the network coordinator is cleared (line 10).

The boundaries of logical channel monitoring intervals are defined by the periodic issuing of beacon transmission requests, as specified by the management action at line 6. Each time a new logical channel monitoring interval is started (line 6): the logical channel status is evaluated with respect to logical channel idleness (line 8); the value of the channel idle status is set to *true* (line 12); it will remain with that value until a frame is correctly received from the logical channel.

Algorithm 3 should be combined with low-level node failure detection mechanisms to ensure stability in the presence of node crash failures. The logic channel switch procedure (lines 9 and 18) assumes switching to a correct channel, meaning that the value of O_d is cleared ($O_d = 0$). The network coordinator also resets the list of its associated nodes (lines 10 and 19) to avoid a false channel idleness detection within the newly selected logical channel.

Algorithm 4 describes the execution of the logical channel diversity policy at nodes other than the network coordinator. We dubbed such nodes as non-coordinator nodes for the sake of simplicity. When a channel failure indication is received (line 5), a non coordinator node performs a switch operation to other logical channel (line 6), utilising the same pre-defined sequence used by the network coordinator. Provided the network coordinator operates in the same logical channel, and that beacon frames are received by the node, no further action is required to restore communication.

However, it may happen that beacon frames are not received by node. Consequently the node continues inaccessible and may lose synchronisation with the network coordinator (line 8). If some other frame has been received within that period

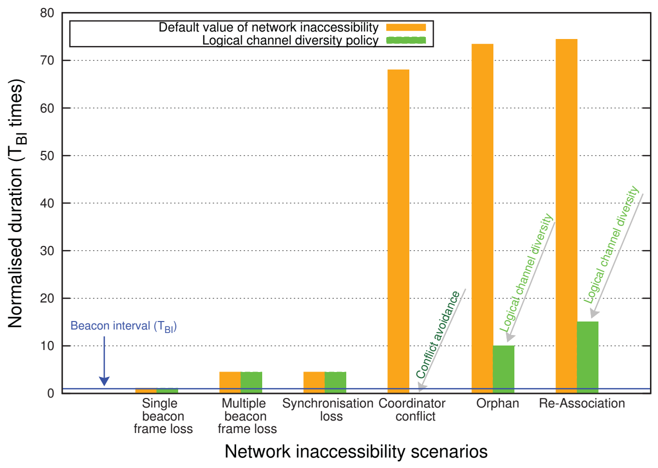
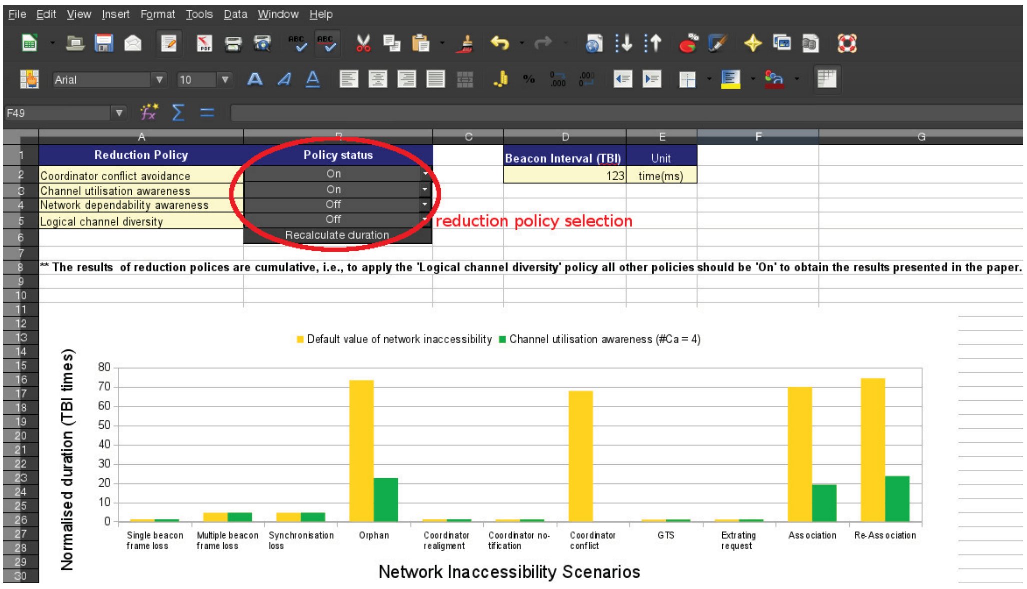


Fig. 7: Impact of the logical channel diversity policy in the IEEE 802.15.4 network inaccessibility ($\#C_a = 1$ and $\#C_a = 2$ for the *orphan* and *re-association* scenarios, respectively).

from the network coordinator (line 9), that is a clear indication the network coordinator remains active in the current logical channel and therefore the node declares itself as an orphan node. Since both the network coordinator and the non-coordinator node are on the same logical channel, the node performs an orphan procedure only on this logical channel, being the cardinality of the channel search set $\#C_a = 1$ (line 11). It results in a quick synchronisation re-establishment between the network coordinator and the non-coordinator node. The orphan procedure updates the node information stored by the network coordinator, alerting it about the remaining presence of the node in the same logical channel.

Otherwise, a *re-association* is performed through the execution of the re-association procedure (line 13). The *re-association* procedure is optimised to be performed within only two logical channels, the new logical channel and the previous one (i.e., $\#C_a = 2$). The use of only two logical channels is justified by: (*a*) the network coordinator remains in the previous logical channel with other non-coordinator nodes; or (*b*) the coordinator detects an idle period and switches to the new logical channel, an action that is faster than the detection of a loss of synchronization. While a channel idleness detection has a duration of $\mathcal{T}_{BI}$, the time required to detect the loss of node synchronisation is, at least, $nrLost = k+1$ times greater than $\mathcal{T}_{BI}$, as we can see in equation 5 presented in section III. It is worthwhile mentioning that the channel scan process, implicit in the *re-association* procedure, may imply a new logical channel change, upon detection of the network coordinator.

The contributions of this policy, and the general impact of our policies to reduce network inaccessibility in IEEE 802.15.4 wireless communications, are presented in Fig. 7.



Reduction Policy	Policy status		Beacon Interval (TBI)	Unit
Coordinator conflict avoidance	On		123	time(ms)
Channel utilisation awareness	On			
Network dependability awareness	Off			
Logical channel diversity	Off			
	Recalculate duration			

reduction policy selection

** The results of reduction polices are cumulative, i.e., to apply the 'Logical channel diversity' policy all other policies should be 'On' to obtain the results presented in the paper.

File Edit View Insert Format Tools Data Window Help
Arial
10
F49
Default value of network inaccessibility
Channel utilisation awareness (#Ca = 4)
Normalised duration (TBI times)
Single beacon frame loss
Multiple beacon frame loss
Synchronisation loss
Orphan
Coordinator realigment
Coordinator notification
Coordinator conflict
GTS
Extrating request
Association
Re-Association
Network Inaccessibility Scenarios

Fig. 8: Use of the network inaccessibility evaluation tool to study the impact of some inaccessibility reduction policies

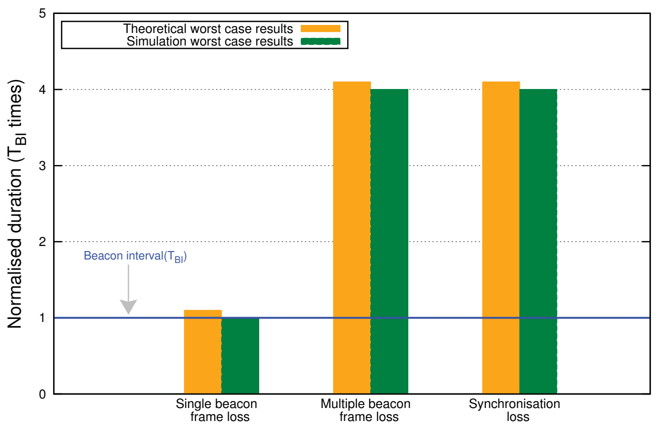


Fig. 9: Preliminary validation of network inaccessibility analysis using the NS-2 simulator and beacon-related scenarios

VI. Network inaccessibility analysis: tool, results, and validation

We designed and developed a tool to evaluate the duration of network inaccessibility scenarios as draw from the IEEE 802.15.4 standard. The analysis tool was built on the LibreOffice suite [11].

The tool is a spreadsheet with enhanced LibreOffice macros, which define mechanisms to calculate and verify parameter values, due its specified restrictions. It is also an open source tool available under a GNU General Public License (GPL) version 3, which can be downloaded at: http://www.karyon-project.eu/wp-content/uploads/2012/10/Inaccessibility_IEEE802.15.4_Beacon-enabled-Karyon.ods.

There are different tabs, each one designed to represent constants, parameters, and configurations allowed to be performed on a standard compliant IEEE 802.15.4 network.

The duration of network inaccessibility scenarios are evaluated and visualised as values in milliseconds (ms), or as normalised values by $\mathcal{T}_{BI}$ units of time. Figure 8 presents an example of the results obtained from the tool; the set of reduction policies to be used draws from its selection as shown in Fig. 8. The screen capture of Fig. 8 also presents the complete set of network inaccessibility scenarios as presented in [18].

Additionally, we have being working to incorporate the analysis of network inaccessibility on the IEEE 802.15.4 NS-2 module. We already have some preliminary results [16], as illustrated by Fig. 9. This preliminary validation compares a fundamental set of beacon loss scenarios, asserting that our theoretical analysis presents the worst case durations face to simulations performed on NS-2. The incorporation of the remaining scenarios on NS-2 simulator, and therefore the proposed reduction policies requires substantial engineering work to complete and complement the IEEE 802.15.4 NS-2 module. This engineering work needs the implementation of essential management operations to simulate IEEE 802.15.4 networks, and the network inaccessibility durations in total compliance with the IEEE 802.15.4 standard.

VII. Relation with the standards

On the other hand, the IEEE 802.15.4 specification has been recently enhanced with amendment IEEE 802.15.4e [8], proposing TDMA like schemes to control the access to the network. This amendment aims to enhance IEEE 802.15.4 network operation for the industrial markets, including the utilisation of periodic channel hopping using pre-defined sequences. The operation of these new protocol variants may benefit from the frame monitoring functions for enhanced dependability introduced in this paper. Conversely, the channel diversity policy can be combined to the now standardised utilisation of channel hopping in IEEE 802.15.4 settings. In practice, all the reduction policies presented in this paper can be easily integrated, and controlled, by a standard compliant solution dubbed *Mediator Layer* [17]. The use of the *Mediator Layer* approach enables and promotes a low level control of communications, which used with the network inaccessibility reduction policies can enhance the dependability and timeliness of the wireless communication standards.

VIII. Conclusion and future work

This paper presented a set of highly effective policies to reduce the negative effects of network inaccessibility on IEEE 802.15.4 wireless networks. The analytical model presented in this paper has shown the limitations of wireless networks to support real-time operation. For example, a standard IEEE 802.15.4 can be affected by network inaccessibility incidents with durations as high as $74.5 \times \mathcal{T}_{BI}$, being $\mathcal{T}_{BI}$ the beacon period. Such long network inaccessibility periods prevents a real-time operation of the network.

However, complemented with frame and channel monitoring mechanisms, a standard IEEE 802.15.4 platform can integrate a set of network inaccessibility control policies that proved to be highly effective in the reduction of the duration of inaccessibility incidents down to $15 \times \mathcal{T}_{BI}$. This is a first step towards solving the difficult problem of enforcing real-time behaviour over wireless networks.

Future research directions of this work includes the study and assessment of new techniques, which exploit multiple communication channels to enhance the reliability of communications; the study of network inaccessibility and the network operation in the presence of malicious attacks; the incorporation of the effects of network inaccessibility in the timeliness model of wireless communications, defining relevant real-time QoS metrics to evaluate if communication network operation is compliant with the level of requirements needed by given applications.

References

[1] I. Aad, P. Hofmann, L. Loyola, F. Riaz, and J. Widmer, “E-MAC: Self-organizing 802.11-compatible MAC with elastic real-time scheduling,” in *IEEE International Conference on Mobile Adhoc and Sensor Systems MASS*, October 2007.

[2] ATMEL, *ATMEL AVR2025: IEEE 802.15.4 MAC Software Package - User guide*, ATMEL Coorporation, May 2012.

[3] P. Bartolomeu, J. Ferreira, and J. Fonseca, “Enforcing flexibility in real-time wireless communications: A bandjacking enabled protocol,” in *IEEE 14th International Conference on Emerging Technologies & Factory Automation (ETFA)*, September 2009.

[4] E. E-López, J. V-Alonso, A. M-Sala, J. G-Haro, P. P-Mariño, and M. Delgado, “A wireless sensor networks MAC protocol for real-time applications,” *Personal Ubiquitous Computing Journal*, January 2008.

[5] D. Eckhardt and P. Steenkiste, “Measurement and analysis of the error characteristics of an in-building wireless network,” in *Annual Conference of the Special Interest Group on Data Communication (SIGCOMM)*, 1996.

[6] M. Hameed, H. Trsek, O. Graeser, and J. Jasperneite, “Performance investigation and optimization of IEEE 802.15.4 for industrial wireless sensor networks,” in *IEEE 13th International Conference on Emerging Technologies & Factory Automation (ETFA)*, September 2008.

[7] IEEE 802.15.4, “Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs) - IEEE standard 802.15.4,” 2011.

[8] ——, “Part 15.4: Low-rate wireless personal area networks (WPANs) - amendment 1: MAC sublayer,” 2012.

[9] A. Koubâa, A. Cunha, M. Alves, and E. Tovar, “i-GAME: An implicit GTS allocation mechanism in IEEE 802.15.4, theory and practice,” *Springer Real-Time Systems Journal*, August 2008.

[10] F. Kuhn, N. Lynch, and C. Newport, “The abstract MAC layer,” in *23rd International Symposium on Distributed Computing (DISC)*, September 2009.

[11] LibreOffice, *LibreOffice - The Document Foundation*, LibreOffice, January 2013, avaiable in http://www.libreoffice.org/. Last access- January 30, 2013.

[12] J. Åkerberg, M. Gidlund, and M. Björkman, “Future research challenges in wireless sensor and actuator networks targeting industrial automation,” in *9th IEEE Internation Conference on Industrial Informatics (INDIN)*, July 2011.

[13] A. Sahoo and P. Baronia, “An energy efficient MAC in WSN to provide delay guarantee,” in *15th IEEE Workshop on Local & Metropolitan Area Networks LANMAN*, June 2007.

[14] M. Sha, G. Hackmann, and C. Lu, “ARCH: Practical channel hopping for reliable home-area sensor networks,” in *IEEE 17th Real-Time and Embedded Technology and Application Symposium (RTAS)*, April 2011.

[15] X.-Y. Shuai and Z.-C. Zhang, “Research of real-time wireless networks control system MAC protocol,” *Journal of Networks*, April 2010.

[16] J. L. R. Souza, A. Guerreiro, and J. Rufino, “Characterizing inaccessibility in IEEE 802.15.4 through theoretical models and simulation tools,” in *4th Simposio de Informática (INFORUM)*, September 2012.

[17] J. L. R. Souza and J. Rufino, “Towards resilient real-time wireless communications,” in *25th Euromicro Conference on Real-Time Systems (ECRTS-WiP)*, July 2013.

[18] ——, “Characterization of inaccessibility in wireless networks - a case study on IEEE 802.15.4 standard,” in *IFIP 3th International Embedded System Simposium IESS*, September 2009.

[19] Yu-Kai, Ai-Chun, and Hui-Nien, “An adaptive GTS allocation scheme for IEEE 802.15.4,” *IEEE Transactions on Parallel and Distributed Systems*, May 2008.

[20] T. Zhou, H. Sharif, M. Hempel, P. Mahasukhon, W. Wang, and T. Ma, “A novel adaptive distributed cooperative relaying MAC protocol for vehicular networks,” *IEEE Journal on Selected Areas in Communications*, January 2011.

[21] X. Zhu, S. Han, P.-C. Huang, A. Mok, and D. Chen, “MBStar: A real-time communication protocol for wireless body area networks,” in *23rd Euromicro Conference on Real-Time Systems (ECRTS)*, July 2011.

This page is intentionally left blank.

A.1.6 Improving NS-2 Network Simulator for IEEE 802.15.4 standard operation

“Improving NS-2 Network Simulator for IEEE 802.15.4 standard operation”, A. Guerreiro, J. L. R. Souza, J. Rufino, Improving NS-2 Network Simulator for IEEE 802.15.4 standard operation, In 5th Simpósio de Informática (INFORUM), Évora, Portugal, September 2013.

This page is intentionally left blank.

Improving NS-2 Network Simulator for IEEE 802.15.4 standard operation [⋆]

André Guerreiro, Jeferson L. R. Souza, and José Rufino

University of Lisbon - Faculty of Sciences
Large-Scale Informatics System Lab. (LaSIGE)
aguerreiro@lasige.di.fc.ul.pt, jsouza@lasige.di.fc.ul.pt, ruf@di.fc.ul.pt

Abstract The IEEE 802.15.4 standard was designed to support the specification of wireless sensor networks (WSNs) and wireless sensor and actuator networks (WSANs), which the utilization is emerging within environments with real-time requirements such as industrial and aerospace. The network simulator NS-2 supports the test, simulation and evaluation of such type of networks, although the real-time support offered by the standard is not yet available in the NS-2 release. This paper presents improvements in the IEEE 802.15.4 NS-2 module to provide a better support for the emulation of networks with real-time requirements, through the incorporation of the contention free period (CFP) and of guaranteed time slot (GTS) defined within the IEEE 802.15.4 module present in the NS-2. Additionally, we also complement this module with IEEE 802.15.4 standard management operations not implemented in the official NS-2 release.

Keywords: NS-2 simulator; wireless sensor and actuator networks; wireless communications; real-time.

1 Introduction

Wireless networking technology has experienced a thriving development in recent years. Due to their unique features such as reduced size, weight, cost, power consumption and mainly the absence of cables, there is a huge interest in developing applications that use wireless sensor networks (WSNs) and wireless sensor and actuator networks (WSANs) in different sectors such as natural resources monitoring [13], aerospace [18], vehicular [9], and industrial [8]. Most of these environments have real-time communication constraints, which implies that WSNs and WSANs must be capable to provide support on real-time communication services.

Several studies have been focused in evaluating wireless technologies for supporting reliable and real-time communication services [15,20,21,18], although

[⋆] This work was partially supported: by the EC, through project IST-FP7-STREP-288195 (KARYON); by FCT/DAAD, through the transnational cooperation project PROPHECY; and by FCT, through the project PTDC/EEI-SCR/3200/2012 (READAPT), the Multiannual Funding Program, and the Individual Doctoral Grant SFRH/BD/45270/2008.

dependability and safety of wireless communications are not yet addressed properly [14]. A recent study presents a *Mediator Layer*[16] approach, which is capable to enhance the dependability and timeliness of medium access control (MAC) sublayer. This study represents a motivation for the current research on the test and evaluation of WSNs and WSANs through network simulators, where we take the IEEE 802.15.4 network standard as a case study.

The use of network simulators is a suitable way to test and evaluate network behaviours using several environmental conditions that can be represented by different environment models and setups. There are several network simulators available [11], some with commercial license, such as OMNeT++ [4] and OPNET [5], and others with open source or academic license, like Prowler [6], TOSSIM [12], NS-3 [3] and NS-2 [1]. NS-3 is the next generation and the evolution of the NS-2 simulator, but does not yet support the IEEE 802.15.4 standard. In the presence of such limitation in the NS-3 we choose the NS-2, which is a widely accepted and used network simulation tool, being open source and modular, supporting the simulation of WSNs and WSANs through the IEEE 802.15.4 standard [10].

However the NS-2 IEEE 802.15.4 module [19] does not have a native support for features that address real-time aspects of communications, such as emulation of a contention free period (CFP) where time slots can be allocated for exclusive access to the network. This paper presents improvements in the IEEE 802.15.4 NS-2 module to provide a better support for the test, simulation and evaluation of IEEE 802.15.4 networks with real-time requirements. We include all the management functions needed to support the use of guaranteed time slots (GTS) for frame transmissions, adapting and extending an implementation of a CFP module proposed by [7]. It is our objective to overcome the existing limitation which, natively, only allow contention-based communications in the 802.15.4 NS-2 module. We evaluate and validate our implementation through test cases that uses different network loads, and performance metrics such as delivery ratio, latency, and energy consumption, allowing a better characterization of IEEE 802.15.4 networks in the support of real-time communications.

To present our contributions, this paper is organized as follows: Section 2 presents an overview of the IEEE 802.15.4 standard. Section 3 presents an overview of the NS-2 and its IEEE 802.15.4 module. Section 4 describes our enhancements in the IEEE 802.15.4 NS-2 module. Section 5 presents, compares, and discusses the results obtained from different network load patterns and GTS utilization through the simulation work. Finally, section 6 draws some conclusions and future research directions.

2 IEEE 802.15.4 overview

The IEEE 802.15.4 specification [17] is a standard that allows the creation of wireless networks, being more specifically oriented for the creation of WSNs and WSANs. Each IEEE 802.15.4 network has a special node dubbed network coordinator, which defines a set of characteristics of the network such as addressing,

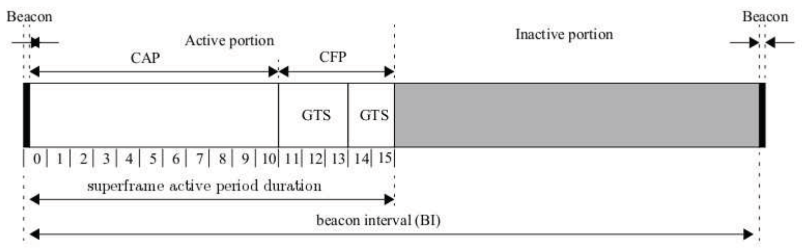


Figure 1: Superframe structure [17]

supported channels, and operation mode. The network can operate either in a beacon-enabled mode or in a nonbeacon-enabled mode. In the beaconless mode, the protocol is essentially a simple Carrier Sense Multiple Access with collision avoidance (CSMA-CA) protocol. Since most of the unique features of IEEE 802.15.4 are in the beacon-enabled mode, like support for communications with real-time restrictions we will focus our attention on this mode. In the beacon-enabled mode the network coordinator coordinates the access to the network by periodically transmitting a special frame dubbed Beacon, which delimits the structure dubbed superframe that specifies the intrinsic rules to perform such access. The period that specifies the consecutive beacon transmissions is dubbed beacon interval (BI).

The superframe structure depicted in Fig. 1 may comprise two periods: a mandatory active period, and an optional inactive period. Each active period is divided into a contention access period (CAP), and an optional contention free period (CFP). The CAP was designed for general purpose traffic, using a contention-based approach in the access of the network. The CFP was designed to support real-time traffic, being divided in transmission windows dubbed guaranteed time slots (GTSs) that uses an exclusive and contention-free approach in the access of the network. Once a given GTS slot is allocated to a node, only this node can transmit in this time interval. Finally, the inactive period was designed to power-saving purposes, where all nodes use such period to save the energy spent in the listen process.

3 IEEE 802.15.4 in NS-2 Simulator

The network simulator NS-2 is a discrete event simulator developed in a collaborative effort by many institutions, containing contributions from different researchers [2]. As a discrete-event simulator, all actions in NS-2 are associated with events rather than time. NS-2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up

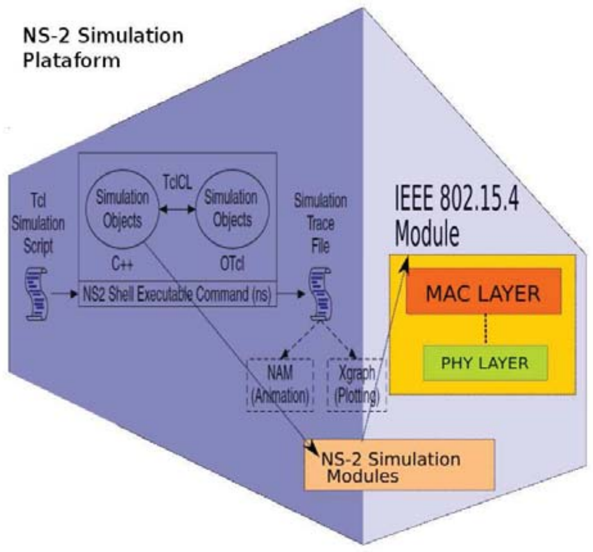


Figure 2: NS-2 802.15.4 module architecture.

simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TclCL as illustrated in the rightmost part of Fig. 2. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a node handle) is just a string in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It is possible to define its own procedures and variables to facilitate the interaction. The member procedures and variables in the OTcl domain are called instance procedures. The basic architecture is represented in Fig. 2.

The IEEE 802.15.4 NS-2 module is provided in the form of methods of each layer specified in the IEEE 802.15.4 standard [10]. The module came with different functionalities, and support different network topologies (star and point-to-point), two types of operation (beacon and non-beacon enabled), and basic MAC management actions such as Association, Channel Scan, energy model, etc. However, the communication during CFP is not implemented in a modular way in the current IEEE 802.15.4 NS-2 module. The absence of the GTS mechanism is a major drawback once is fundamental for real-time WSAN applications, allowing a node to operate on the channel within a portion of the superframe that is dedicated exclusively to it. **IEEE 802.15.4 MAC:** The MAC sub-layer handles all access to the physical radio channel and is responsible for tasks such as network synchronization based on beacons, network association and disassociation, data security, network access, and handling and maintaining the GTS mechanism. The current IEEE 802.15.4 module within the NS-2 simulator has some differences in the behaviour regarding the implementation of the standard MAC management actions. Actions needed for the support of real-time data transmissions such as GTS allocation and deallocation are not implemented. Additionally, other management actions such as disassociation and network ID conflict notification, and all auxiliary mechanisms needed to support the execution of such actions are not implemented and should be developed and incorporated in the IEEE 802.15.4 module, enhancing the compliance with the IEEE 802.15.4 standard.

4 Improvements in the 802.15.4 NS-2 Simulator module

In order to achieve a better real-time support from the 802.15.4 simulation module in NS-2, we extend the existent module to provide the GTS mechanism for network nodes. Adapting the CFP implementation in [7], to the latest version of the simulator used in this study, revealed some issues and incompatibilities. The referenced implementation proposed in [7] prevents the use of MAC management commands such as Orphan Notification and Coordinator Realignment. In our implementation this was corrected and the GTS can be activated from the NS-2 script without affecting other MAC services. A description of the implemented GTS mechanism and enhanced MAC management operations is given below.

4.1 CFP and GTS implementation in NS-2

Data can be transmitted in three ways, *Direct transmission* which means that data is sent during the CAP. *Indirect transmission* is only available for coordinators, data is placed on the indirect transmission queue and is sent during the CAP when polled. And finally *GTS transmission*, which requires that a node has to use a GTS slot to transmit its data. To allow this, a GTS slot is allocated to the node for the specified data transmission. Although a data transmission

Algorithm 1 Transmission Data Frame using GTS

```
1: Begin.
2: $MAC.Data.Send.Request(data)$;
3: when allocated GTS is reached do
4:    $MAC.Data.transmit(data)$;
5: end when
6: End.
```

request can occur at any-time in a superframe, a data transmission request using a GTS is required to transmit the data only during the allocated GTS. Therefore, in our implementation described in Algorithm:1, it is checked if the data

Algorithm 2 Coordinator processing a GTS request command

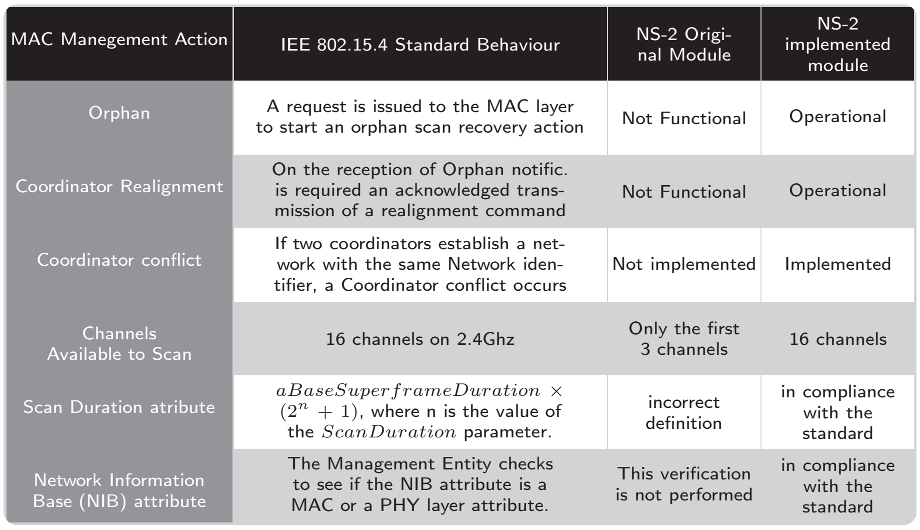
```
1: Begin.
2: $MAC.Mgmt.GTS.Request(node_{addr}, nr\_slots)$;
3: if $nr\_slots$ are available then
4:     $MAC.Mgmt.GTS.allocate(node_{addr}, nr\_slots)$;
5:     $MAC.Mgmt.GTS.updateGTSList(node_{addr})$;
6: else
7:     $MAC.Mgmt.GTS.Response(slots\_not\_available)$;
8: end if
9: End.
```

transmission request using a GTS is in the allocated GTS duration or not, as represent on line 3. Since the procedure to check the remaining GTS time is also implemented, multiple data can be transmitted during a GTS, which complies with the IEEE 802.15.4 standard. When a coordinator receives a GTS request command from a node, willing to transmit data, Algorithm:2 is executed by the coordinator. After checking if the node GTS slot is valid (line 3), which means the $node_{addr}$ is already known by the coordinator and nr_slots are available, the allocation is made (line 4). If the operation is successfully concluded, the final CAP slot is updated as in line 5, and the updated beacon is sent. If all the GTS slots are occupied at the time, an information regarding $slots\_not\_available$ is sent to the node, as described in line 7. The information from the GTS allocation or deallocation is delivered in the next beacon frame to the nodes that sent the GTS request command, letting them know the result of the requesting process.

4.2 Enhancing MAC management actions according to the IEEE 802.15.4 standard

The NS-2 simulator module has some differences in the behaviour regarding the implementation of IEEE 802.15.4 standard MAC management actions. Taking this in consideration we added the functions presented in Fig. 3. Some operations are implemented in the original module but they are not fully functional. We corrected them and implemented other additional and needed operations. The Coordinator conflict is one of those such operations. Coordinator conflict occurs when more than one coordinator is active within the same network. By default, each network has an identifier, the source identifier, which identifies the network uniquely and is used by the coordinator in beacon transmissions. If some other (possibly old) coordinator enters the network operational space, e.g., after having been away from some period of time, the network may have two different coordinators transmitting beacons with the same source identifier.

The resolution in turn will request the MAC layer to perform an active scan. This scan is realized in all currently used logical channels. If the protocol management entities decide that the node was orphaned, a request is issued to the MAC layer to start an orphan scan recovery action, over a specified set of logical channels. For each logical channel: a MAC orphan notification command is sent; as reply, a MAC realignment command from the previously associated coordinator, is awaited for during a given period. Once such MAC command is received the node terminates the scan and the network becomes accessible. At the coordinator the need to assist MAC layer management actions starts when a MAC orphan notification command is received. Upon processing by protocol management entities, the acknowledged transmission of a MAC realignment command is requested. Relatively to the channel scan process carried out by the different nodes, they should be able to scan all channels available to either associate to a network or to establish one. However, since we aim to simulate a network operating in 2.4 Ghz we removed the limitation of scanning only the first 3 channels. This limitation was removed and the scan of all the 16 channels defined by the standard is now allowed. The duration of each scan was also incorrect, once its parametrization was not set in compliance with the standard. Again, the issue was corrected.



MAC Manegement Action	IEE 802.15.4 Standard Behaviour	NS-2 Original Module	NS-2 implemented module
Orphan	A request is issued to the MAC layer to start an orphan scan recovery action	Not Functional	Operational
Coordinator Realignment	On the reception of Orphan notific. is required an acknowledged transmission of a realignment command	Not Functional	Operational
Coordinator conflict	If two coordinators establish a network with the same Network identifier, a Coordinator conflict occurs	Not implemented	Implemented
Channels Available to Scan	16 channels on 2.4Ghz	Only the first 3 channels	16 channels
Scan Duration atribute	$aBaseSuperframeDuration \times (2^n + 1)$, where n is the value of the $ScanDuration$ parameter.	incorrect definition	in compliance with the standard
Network Information Base (NIB) attribute	The Management Entity checks to see if the NIB attribute is a MAC or a PHY layer attribute.	This verification is not performed	in compliance with the standard

Figure 3: NS-2 IEEE 802.15.4 Module behaviour comparison

5 Results

We now describe our evaluation metrics, the simulation setup and finally the results achieved in improving the IEEE 802.15.4 MAC NS-2 module.

5.1 Evaluation Metrics

The evaluation metrics that we used are applied to the network, taking into consideration all the nodes involved in the simulation. The metrics are the following:

– *Data frame delivery Ratio (DFDR)*, which is the ratio between the total number of frames received in MAC sub-layer and the total number of data frame transmit requests during the simulation period. In our simulation we consider the data frames transmit requests issued by all the nodes but the coordinator.

$$DFDR = \frac{Total\ Data\ frames\ received \times 100}{Total\ Data\ frames\ transmit\ requests} \tag{1}$$

– *Latency*, which represents the transfer time of a data frame to a one-hop neighbour. For each individual data frame transfer the frame transfer latency represents the interval between the data frame reception instant (T_{rxData}) and the instant the corresponding data frame transmit request is issued (T_{txData}). This frame includes the data frame processing and queueing time at the nodes, the data frame transmission time and the back off interval (if applicable). The average latency can then be calculated over all successful end-to-end transmissions within the simulation run.

$$AverageLatency = \frac{\sum_{allreceivedframes}(T_{rxData} - T_{txData})}{Total\ number\ of\ received\ frames} \tag{2}$$

On the other hand, the worst case value is given by:

$$WorstCaseLatency = \max_{allreceivedframes}(T_{rxData} - T_{txData}) \tag{3}$$

– *Energy*, the energy model present in NS-2 is used to calculate the amount of energy consumed by the nodes during the simulation time.

$$Energy\ Used\ per\ Node = \frac{Total\ Energy\ Used}{Number\ of\ Nodes} \tag{4}$$

5.2 Simulation Setup

The characteristics of the simulation setup scenario are shown in Table 1. The star topology network was chosen in this simulation. The network was simulated with seven nodes, where one of these nodes, in the center, was the coordinator. All other nodes are in the radio transmission range of the coordinator. Additionally all nodes are in a single broadcast domain, which means that all the nodes are within the range of each other. To evaluate the network behaviour, the six remaining nodes constantly transmit data frames to the coordinator during CAP or CFP. The traffic generator is set to produce Constant Bit Rate traffic (CBR), which means data frames are transmitted at a constant rate from the nodes to the coordinator. The interval between each data transmission request is successively set to 1, 0.1, 0.01 and 0.001 seconds. Given the packet size of 70 bytes, this means the network load is monotonically increased, adopting the values of 15, 40, 70, 95 kilobytes. The MAC management actions required for node association with its coordinator and the GTS allocation times (if required) are excluded from the evaluation scope in the present simulation run.

Table 1: NS-2 Simulation Parameters

Simulation Parameters	
NS-2 Version	2.35 updated with GTS features
Network Topology	Star Topology
Nodes	7
Traffic	Constant Bit Rate (CBR)
Reception range	15m
Carrier Sense range	15m
Packet Size	70 bytes
CAP Transmission Type	Direct, using CSMA/CA
CFP Transmission Type	GTS transmission
Transmission/Reception Power	30mW
Beacon	Enabled
Beacon Order	3
Superframe Order	3
Maximum CSMA/CA Attempts	4
Simulation Time	600 seconds

5.3 Simulation Results

Figure 4 represents the delivery ratio on the network, providing a comparison of the results achieved for transmissions requests issued during the CAP and CFP periods. During CFP, nodes use the allocated GTS and get direct and exclusive network access, which allows to achieve about 100% delivery rate. In CAP the delivery ratio drops in function of the increase in the network load. This is explained by the occurrence of collisions during CAP, or due to the number of nodes attempting to access the medium. In the CSMA/CA protocol a data frame transmit request is dropped, if the number of transmission attempts exceed a given threshold defined by the Maximum CSMA/CA attempts (Table 1).

Figure 5 shows the latency comparison between a data frame transmission using CFP and CAP. While the latency remains almost constant when data frames are transmitted during CFP, using allocated GTS, the latency highly increases while using CAP. The constancy achieved in data frame transfer times during CFP is a sign of determinism and predictability and shows in Figure 5 in two ways: an (almost) constant worst-case data frame transmission latency; the optimal value of this latency, which does not exceed 0.002936 seconds. This is due to nodes during CFP get exclusive network access, meaning nodes do not have to check if the media is idle and no collisions occur for those nodes. These results show the importance of the GTS mechanism in applications with real-time requirements on which deterministic data frame transmission times are mandatory. Additionally, the data frame transmission latency increases in CAP, up to the worst-case value of 0.010512 seconds, given the worst-case network load in the simulation setup. Finally, Figure 6 represents the average energy consumption by the nodes during the simulation period. It noticed that the

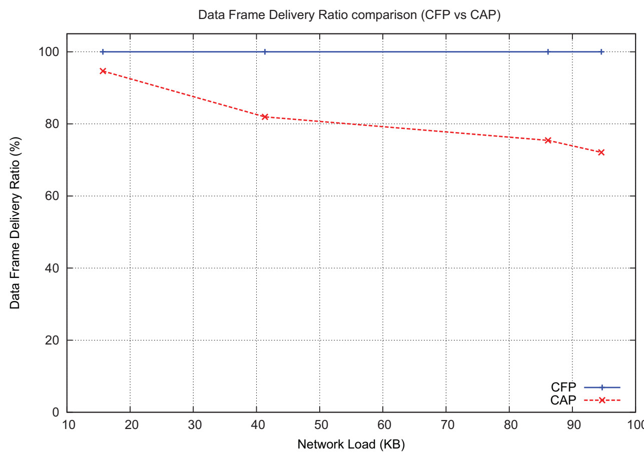


Figure 4: Data Frame Delivery Ratio comparison between transmission during CAP and CFP

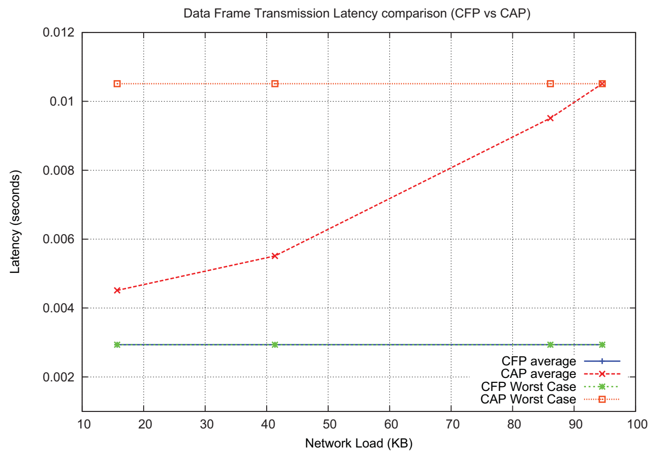


Figure 5: Data frame transmission Latency comparison between transmission during CAP and CFP

energy consumption increases when using CFP in comparison with CAP as result of required beacon frame reception tracking by the node, an action that obliges the node to switch-on its transceiver during the active period of every superframe instance. Contention-based access is more efficient under light network loads, whereas contention-free access becomes preferable when the background network load increases.

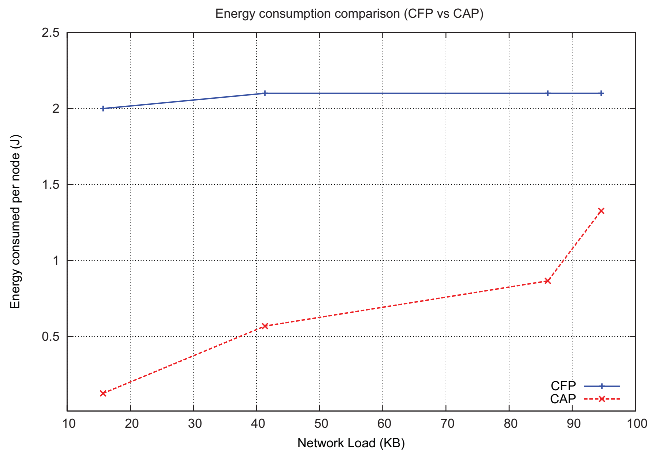


Figure 6: Energy consumption per node for data transmission during CAP and CFP

6 Conclusion and future directions

In this paper the current IEEE 802.15.4 module implemented in NS-2 is modified and extended to include the use of the GTS mechanisms based on the standard. So, the operations of the GTS allocation, use and deallocation are implemented. The addition of unimplemented MAC operations enhanced the simulation module so that is in accordance to the standard.

Based on NS-2 simulations, we evaluate the performance of various features in the IEEE 802.15.4 MAC. We find that data transmission during the CAP reduces energy cost due to idle listening in the backoff period but increases the collision at higher rate and larger number of sources. While the use of GTS in the CFP can allow dedicated bandwidth to a device to ensure low latency, the device need to track the beacon frames in this mode, which increases the energy cost. The addition of available channels to scan during association revealed an increase of the association time an energy cost, but made the NS-2 more compliant to the standard. Having better tools allow us to better understand the temporal aspects, in this case, of IEEE 802.15.4 networks. Thus, the knowledge of the network performance allows a better analysis and definition of a timeliness model, representing a first though crucial step for achieving an effective support to real-time operation in IEEE 802.15.4 networks.

This greatly assists the IEEE 802.15.4 standard related research. To benefit the research community, our NS-2 implementation of this protocol is publicly available online at: `http://www.karyon-project.eu/wp-content/uploads/2012/10/ns-2-2.35-with-gts.tar.gz`.

References

1. Ns-2: The network simulator version 2. http://www.isi.edu/nsnam/ns/
2. Ns-2: The network simulator version 2 contributed code. http://nsnam.isi.edu/nsnam/index.php/Contributed_Code/
3. Ns-3: The network simulator version 3. http://www.nsnam.org/
4. Omnet++ discrete event simulation system. http://www.omnetpp.org/
5. Opnet:application and network performance tool. http://www.opnet.com
6. Prowler: Probabilistic wireless network simulator. http://www.isis.vanderbilt.edu/Projects/nest/prowler/
7. Choi, W., Lee, S.: Implementation of the IEEE 802.15.4 module with CFP in NS-2. Telecommunication Systems (Aug 2011), http://link.springer.com/10.1007/s11235-011-9548-7
8. Han, S., Zhu, X., Mok, A., Chen, D., Nixon, M.: Reliable and real-time communication in industrial wireless mesh networks. In: 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). pp. 3–12 (2011)
9. Hartenstein, H., Laberteaux, K.: A tutorial survey on vehicular ad hoc networks. IEEE Communications Magazine 46(6), 164 –171 (June 2008)
10. IEEE 802.15.4: Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs) - IEEE standard 802.15.4. IEEE P802.15 Working Group (2011), Revision of IEEE Standard 802.15.4-2006
11. Korkalainen, M., Sallinen, M., Kärkkäinen, N., Tukeva, P.: Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications. 2009 Fifth International Conference on Networking and Services (2009)
12. Levis, P., Lee, N.: TOSSIM : A Simulator for TinyOS Networks pp. 1–17 (2003)
13. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless sensor networks for habitat monitoring. Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications - WSNA '02 p. 88 (2002)
14. Åkerberg, J., Gidlund, M., Björkman, M.: Future research challenges in wireless sensor and actuator networks targeting industrial automation. In: 9th IEEE International Conference on Industrial Informatics (INDIN) (July 2011)
15. Shuai, X.Y., Zhang, Z.C.: Research of real-time wireless networks control system MAC protocol. Journal of Networks (April 2010)
16. Souza, J.L.R., Rufino, J.: Towards resilient real-time wireless communications. In: 25th Euromicro Conference on Real-Time Systems (ECRTS-WiP) (July 2013)
17. Standard, I., Society, I.C.: IEEE Standard for Local and metropolitan area networks, Part 15 . 4 : Low-Rate Wireless Personal Area Networks (LR-WPANs) IEEE Computer Society S ponsored by the (2011)
18. Stone, T., Alena, R., Baldwin, J., Wilson, P.: A viable COTS based wireless architecture for spacecraft avionics. In: IEEE Aerospace Conference. pp. 1–11 (2012)
19. Zheng, J., Lee, M.J.: A Comprehensive Performance Study of IEEE 802.15.4, chap. 4, pp. 218–237. IEEE Press, Wiley Interscience (June 2006)
20. Zhou, T., Sharif, H., Hempel, M., Mahasukhon, P., Wang, W., Ma, T.: A novel adaptive distributed cooperative relaying MAC protocol for vehicular networks. IEEE Journal on Sel. Areas in Comm. (Jan 2011)
21. Zhu, X., Han, S., Huang, P.C., Mok, A., Chen, D.: MBStar: A real-time communication protocol for wireless body area networks. In: 23rd ECRTS (July 2011)

A.1.7 Improving NS-2 Network Simulator To Evaluate IEEE 802.15.4 Wireless Networks Under Error Conditions

“Improving NS-2 Network Simulator To Evaluate IEEE 802.15.4 Wireless Networks Under Error Conditions”, A. Guerreiro, J. L. R. Souza, J. Rufino. To appear in 3th International Conference on Sensor Networks (SENSORNETS), Lisbon, Portugal, Jan. 2014.

This page is intentionally left blank.

Improving NS-2 Network Simulator to evaluate IEEE 802.15.4 wireless networks under error conditions

André Guerreiro, Jeferson L. R. Souza and José Rufino
University of Lisbon - Faculty of Sciences
Large-Scale Informatics System Lab. (LaSIGE)
aguerreiro@lasige.di.fc.ul.pt, jsouza@lasige.di.fc.ul.pt, ruf@di.fc.ul.pt

Keywords: NS-2 simulator, wireless communications, network inaccessibility, timeliness, dependability

Abstract: The behaviour of wireless networks in the presence of error conditions is still being studied by the research community. Improvements on the evaluation methods and tools are crucial to acquire a better knowledge, and understanding of the network operation under such conditions. This paper presents enhancements on the network simulator NS-2 to support the evaluation of the IEEE 802.15.4 standard, used as a case study. We are specially interested to evaluate the temporal behaviour of the network operation under errors conditions, considering the applicability of the IEEE 802.15.4 standard in safety-critical environments such as industrial and vehicular.

1 INTRODUCTION

The applicability of wireless technologies on environments with temporal restrictions has been attracting interest of the real-time research community in the last decade (Åkerberg et al., 2011; Stone et al., 2012). The main advantages offered by wireless networks are: the reduction of size, weight, and power (SWaP) consumption; the ability to have mobile entities; and the possibility to establish networking communications where the use of wires is extremely difficult or impractical (Kandhalu and Rajkumar, 2012).

There are many studies in wireless communications addressing the temporal behaviour of communication services at the lowest level of the protocol stack (Han et al., 2011; Shuai and Zhang, 2010; Hou and Bergmann, 2010). These studies pay little or no attention to the dependability aspects of medium access control (MAC) sublayer and its services, which are essential to assure the timeliness and resilience of the network when operating under error conditions.

This paper evaluate the network simulator NS-2 to identify its limitations, proposing enhancements to provide a better knowledge and understanding of wireless network operation under such conditions. The NS-2 was chosen due to their native support to simulate wireless networks based on IEEE 802.15.4 standard, which is used as case study.

Our research achievements are organised as follows: Section 2 describes our system model, which comprises the assumptions utilised through the paper; Section 3 presents a brief overview of the IEEE 802.15.4 standard; Section 4 addresses the main temporal issues of the network operation under error conditions; Section 5 presents a brief overview of the NS-2 simulator, including its limitations; Section 6 presents the improvements in the IEEE 802.15.4 NS-2 module, including a fault injector that complements the existent NS-2 mechanisms, and a new component to perform the temporal analysis of the network operation; Section 7 presents the simulation setup of the Inaccessibility Scenarios and a simulation script description; Section 8 presents the results obtained in the simulation of IEEE 802.15.4 networks, allowing an enhanced temporal evaluation of such networks; Finally, Section 9 presents some conclusions and future directions of this work.

2 SYSTEM MODEL

Our system model is formed by a set of wireless nodes[1] $X = \{x_1, x_2, \ldots, x_n\}$, being $1 < n \leq \#A$, where A is the set of all wireless nodes using the same communication channel. The set X itself represents a network entity dubbed wireless network segment (WnS), as depicted in Figure 1. A WnS establishes a wire-

[1]A wireless node is a networked device capable to communicate with other nodes

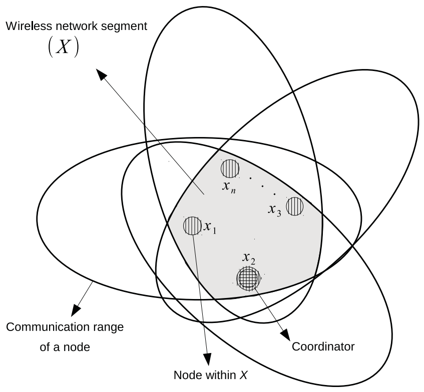


Figure 1: The graphical representation of a wireless network segment.

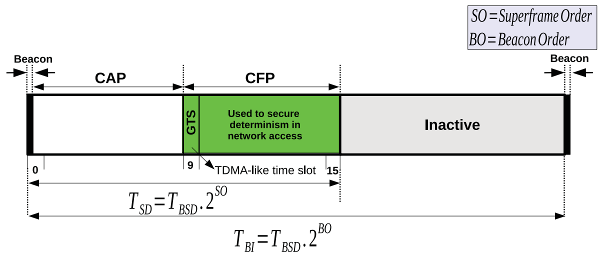


Figure 2: Superframe structure (Standard and Society, 2011)

less network where each node can sense one another within one-hop of distance, being more complex networks composed by more than one WnS. For simplification purposes, our analyses assume a network with one WnS, being its behaviour supported by the following assumptions:

1. The communication range of X, i.e. its broadcast domain, is given by: $B_X = \bigcap_{j=1}^{n} B_D(x)$, $\forall x \in X$, where $B_D(x)$ represents the communication range of a node x;
2. $\forall x \in A,\ x \in X \iff B_D(x) \bigcap B_X = B_X$ or, as a consequence of node mobility, $x \notin X \iff B_D(x) \bigcap B_X \neq B_X$;
3. $\forall x \in X$ can sense the transmissions of one another;
4. $\exists x \in X$ which is the coordinator, being unique and with responsibility to manage the set;
5. A network component (e.g. a node $x \in X$) either behaves correctly or crashes upon exceeding a given number of consecutive omissions (the component's *omission degree*, f_o) in a time interval of reference[2], $\mathcal{T}_{rd}$;
6. failure bursts never affect more than f_o transmissions in a time interval of reference, $\mathcal{T}_{rd}$;
7. omission failures may be inconsistent (i.e., not observed by all recipients).

For a given WnS, assumptions 1, 2, and 3 define the physical relationship between nodes, assumption 4 defines the existence of a coordinator, and assumptions 5, 6, and 7 define how the occurrence of communication errors are modelled and handled within the WnS. All communications and relations between nodes are established at MAC level, which are reinforced by assumption 3. As a consequence of mobility, nodes may be driven away of a given WnS (assumption 2). All communication errors within WnS are transformed into omissions (assumption 5), and in the context of network components an omission is an error that destroys a data or control frame.

[2]For instance, the duration of a given protocol execution. Note that this assumption is concerned with the total number of failures of possibly different nodes.

3 IEEE 802.15.4 OVERVIEW

IEEE 802.15.4 is a standard for wireless sensor and actuator networks (WSANs), which support two operating modes: beacon-enabled and non beacon-enabled. In this paper we only address the beacon-enabled mode, which supports traffic with temporal restrictions. The network coordinator controls the network access through the superframe structure depicted in Figure 2.

This superframe structure comprises an active period, and optionally, an inactive period. In the active period there are a contention access period (CAP) and a contention free period (CFP). CAP is used to transmit traffic without any temporal guarantee and in a best effort approach. In CFP nodes can allocate time slots to transmit traffic with temporal restrictions (i.e., time division multiple access (TDMA) approach), where such slots are dubbed guarantee time slots (GTSs). The inactive period is used for power saving purposes (when needed).

As depicted in Figure 2, the duration of CAP and CFP are defined by two parameters: the beacon order (*BO*); and the superframe order (*SO*). The value of *BO* defines the superframe duration (i.e., the beacon interval, $\mathcal{T}_{BI}$), and the value of *SO* the duration

Scenario	Equation
Single Beacon Frame Loss	$\mathcal{T}^{wc}_{ina\leftarrow sbfl} = \mathcal{T}_{BSD} \,.\, (2^{BO} + 1)$
Multiple Beacon Frame Loss	$\mathcal{T}^{wc}_{ina\leftarrow mbfl} = \mathcal{T}_{BSD} \,.\, \left(2^{BO} + 1\right) \,.\, nrLost$
Synchronisation Loss	$\mathcal{T}_{ina\leftarrow nosync} = \mathcal{T}_{BSD} \,.\, \left(2^{BO} + 1\right) \,.\, nrLost$
Orphan Node	$\mathcal{T}^{wc}_{ina\leftarrow orphan} = \mathcal{T}_{ina\leftarrow nosync} + \mathcal{T}_{MLA}(Orphan) + \sum_{j=1}^{nrchannels} \left(\mathcal{T}^{wc}_{MAC}(Orphan) + nrWait \,.\, \mathcal{T}_{BSD}\right) + \mathcal{T}^{wc}_{MAC\_ack}(Realign)$
Coordinating Realignment	$\mathcal{T}^{wc}_{ina\leftarrow realign} = \mathcal{T}_{MLA}(Realign) + \mathcal{T}^{wc}_{MAC\_ack}(Realign)$
Coordinator Conflict Detection	$\mathcal{T}^{wc}_{ina\leftarrow C\_Detection} = \mathcal{T}^{wc}_{MAC\_ack}(C\_Conflict)$
Coordinator Conflict Resolution	$\mathcal{T}^{wc}_{ina\leftarrow C\_Resolution} = \mathcal{T}_{MLA}(Conflict) + \sum_{j=1}^{nrchannels} \left[\mathcal{T}^{wc}_{MAC}(Beacon\_R) + nrWait.\mathcal{T}_{BSD}\right] + \mathcal{T}_{MLA}(Realign) + \mathcal{T}^{wc}_{MAC}(Realign)$
Extract Request	$\mathcal{T}^{wc}_{ina\leftarrow extReq} = \mathcal{T}^{wc}_{MAC_{ack}}(ExtReq) + \mathcal{T}_{wait}$
GTS request	$\mathcal{T}^{wc}_{ina\leftarrow GTS} = \mathcal{T}^{wc}_{MAC\_ack}(GTS)$
Association	$\mathcal{T}^{wc}_{ina\leftarrow assoc} = \sum_{j=1}^{nrchannels} \left[\mathcal{T}^{wc}_{MAC}(Beacon\_R) + nrWait.\mathcal{T}_{BSD}\right] + \mathcal{T}_{MLA}(Beacon) + \mathcal{T}^{wc}_{ina\leftarrow extReq} + \mathcal{T}_{MLA}(AssocReq) + \mathcal{T}^{wc}_{MAC\_ack}(AssocReq)$
Re-Association	$\mathcal{T}^{wc}_{ina\leftarrow reAssoc} = \mathcal{T}_{ina\leftarrow nosync} + \mathcal{T}^{wc}_{ina\leftarrow assoc}$

Table 1: Easy-to-use formulas defining the durations of periods of network inaccessibility

of the active period (CAP+CFP). The duration of the beacon interval is $\mathcal{T}_{BI} = \mathcal{T}_{BSD} \,.\, 2^{BO}$, where $\mathcal{T}_{BSD}$ is the base value of $\mathcal{T}_{BI}$ when $BO = 0$, as defined within the IEEE 802.15.4 standard. The real length of CFP depends on the number of GTSs actually allocated. There is no inactive period when $BO = SO$, being the duration of the active period equal to $\mathcal{T}_{BI}$.

4 CHARACTERISING IEEE 802.15.4 NETWORK OPERATION UNDER ERROR CONDITIONS

The utilisation of IEEE 802.15.4 WSANs is emerging in environments such as industrial and vehicular, where some networking communications must respect restrict temporal constrains. Wireless communications may be affected by different sources of interferences, such as electromagnetic waves, obstacles on the communication path, or even by the mobility of nodes. Communication errors may occur as a consequence of such interferences, disturbing the communication services and the network operation itself.

The occurrence of such communication errors may affect two different types of operations, which are related to transmit data traffic and to control and maintain the network operation. The literature presents different works (Wang et al., 2012; Saifullah et al., 2011), which are only focused on the characterisation and presence of errors on data transmissions, disregarding the negative effects of such conditions in the MAC management operations.

In the context of MAC management operations, an already known severe consequence of communication errors is dubbed network inaccessibility (Souza and Rufino, 2009). A network inaccessibility period is characterised by the occurrence of ”blackouts“ within networking communications, where the network remains inaccessible by a temporary period of time. A research study performed by (Souza and Rufino, 2009) presents formulas to specify the duration of network inaccessibility for the IEEE 802.15.4 standard. Those communication ”blackouts“ may have a huge impact on the timeliness and dependability of the whole networking system, where a better evaluation may suggest the incompatibility of the guarantees offered by the communication service, and the temporal requirements of the target environment.

In Table 1 we present a summary of the worst case duration (represented by the superscript (^{wc})) for each network inaccessibility scenario. As an example, we will briefly explain the characterisation of the most evident network inaccessibility scenarios, which are related with the loss of beacon frames. Three different inaccessibility scenarios may occur if such frames are not received correctly.

A single beacon frame loss occurs when only one beacon is lost. The duration of such scenario is equal to $\mathcal{T}_{BI} + \mathcal{T}_{BSD}$, where $\mathcal{T}_{BSD}$ is utilised as a temporal compensation to accommodate possible clock deviations between network nodes. The loss of multiple and consecutive beacons characterises the occurrence of the multiple beacon frame loss scenario, where a correct beacon is received after the loss of the previous *nrLost* beacons. The synchronisation loss is a special case of the multiple beacon frame loss scenario where after the loss of *nrLost* beacons the next beacon is also lost. The duration of both multiple beacon frame loss and synchronisation loss is a multiple of the single beacon frame loss, which is $nrLost \,.\, (\mathcal{T}_{BI} + \mathcal{T}_{BSD})$. For simplification purposes we replace the $(\mathcal{T}_{BSD}.2^{BO})$ by $\mathcal{T}_{BI}$, as indicated in section 3. The complete network inaccessibility characterisation for the IEEE 802.15.4 is present in (Souza and Rufino, 2009).

5 NS-2 SIMULATOR OVERVIEW

The NS-2 is a discrete-event simulation tool, widely used to study the dynamics of communication networks. It is developed in a collaborative effort by many institutions, containing contributions from different researchers.The simulation library and network protocols are written using the *C* and *C++* languages. The simulation environment is described and modified using the *OTcl* script language, without the necessity to recompile the whole NS-2 source code.

Every action in NS-2 is associated with events rather than time. An event comprises an execution time, a set of tasks, and a reference to the next event. These events are connected to each other, and form a chain of events on the simulation time line. The sequential execution of this chain of events is controlled and managed by a scheduler component, the brain and execution engine of the NS-2. It is possible to define its own procedures and variables to facilitate the interaction. The member procedures and variables in the *OTcl* domain are called instance procedures.

The IEEE 802.15.4 NS-2 module is provided in the form of methods of each layer specified in the IEEE 802.15.4 standard. The module came with different functionalities, and support different network topologies (star and point-to-point), two types of operation (beacon and non-beacon enabled), and basic MAC management actions such as Association, Channel Scan, energy model, etc.

6 ENHANCING NS-2 SIMULATOR

The evaluation of the network operation under error conditions needs components capable to inject faults in the simulation, which cause the network inaccessibility scenarios described previously. The NS-2 already provides components to perform such fault injection, but these components using an error model not capable to affect specific MAC frames, utilised by the IEEE 802.15.4 to control the access to the network.

To overcame the current error model limitation, we complement the existing NS-2 components with a new fault injector component, which is capable to generate faults in specific MAC frames. We also incorporate in NS-2 a a temporal analysis component,

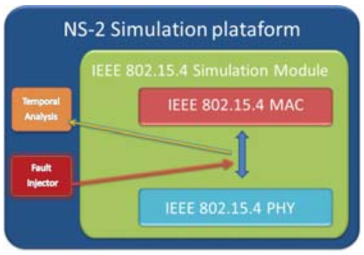


Figure 3: New Features in 802.15.4 module

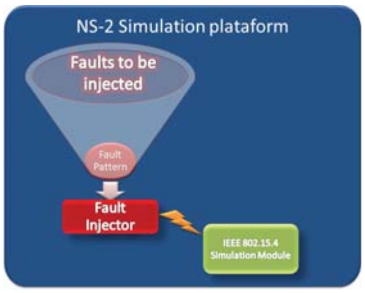


Figure 4: Fault Injector scheme

which is needed to account and measure the effects of faults generated by the fault injector component. These two components are independent of the type of network, being separated from the IEEE 802.15.4 module, as represented in the figure 3.

6.1 Fault Injector Component

Our fault injector is capable to use a fault pattern to inject errors during the simulation. The criteria to define the fault pattern is totally configurable, allowing the definition of deterministic or probabilistic fault patterns. An illustration of the fault injection scheme is shown in the figure 4.

A fault pattern can be defined to generate transmission errors randomly in time (random noise or interference) or be localized in specific time intervals (deterministic noise). On both of these patterns, the fault injector can be customized regarding the type of frame to affect, the rate and the duration of the fault injection.

Patterns with long duration are discouraged for deterministic error models, since such long duration may cause a permanent inaccessibility to the network access. For example, if we are corrupting a beacon frame injecting deterministic faults successively over a long period we may cause the loss of synchronization by the node and consequently this becoming unable to access the network again. However this type of pattern is beyond the scope of this work that is to analyse accidental faults where such pattern does not happen.

Algorithm 1 Fault Injector - A random function

```
1: Begin.
2: $randomTime \leftarrow randomGenerator(seed)$;
3: $NewRandomEvent \leftarrow$ faultInjector(frameToCorrupt);
4: $Scheduler.schedule$(NewRandomEvent,randomTime);
5: $CorruptNode.Update()$;
6: End.
```

To perform the random noise or interference is possible to simulate aleatory errors on the network communication, injecting faults between the MAC and the PHY. A random function implemented in the fault injector allows inserting random corruption events in the NS-2 scheduler as described in Algorithm:1. In case of random noise the instant when the corruption occurs is totally aleatory, and is generated through a seed given by argument as described in line 2. A new event is created and the action associated with it is a frame corruption performed by the fault injector as indicated in line 3. Finally the *NewRandomEvent* which will perform the corruption is inserted in the NS-2 scheduler and executed at the defined instant as in line 4. An information about the corruption occurred in a specific node is recorded as described in line 5.

The fault injector achieves the frame corruption as described in Algorithm:2, accessing the command header of the frame as represented in line 5, and changing a bit in the frame content, implying the drop of these frames in the MAC level of the receiving nodes. When the frame is received if the fault injector is active, we can decide if a specific frame is affected or any frame that a node receives will be corrupted. The parameter $frameToCorrupt$ represented on line 3 is previously defined and if desired all the received frames can be affected defining the $frameToCorrupt$ to a specific value. An information about the corruption occurred in a specific node is recorded as described in line 6. This information is used for a better control of the simulation events. The fault injection may be performed in the coordinator, which implies, depending on the type of frame affected, that the whole network may be inaccessible, in the specific case of affecting a MAC control frame. In case we decide to affect a MAC control frame, affecting specific network points, the fault injection can be performed for example at non-coordinator nodes tracking the reception of beacon frames. In the specific case, when we perform corruption in a MAC control frame such as the beacon in the coordinator, none of the nodes receives the beacon and therefore the whole network will be inaccessible. Otherwise, when the corruption is performed in the nodes that should receive beacon frames, only the node that has

Algorithm 2 Fault Injector Mechanism

```
1: Begin.
2: $MAC.Receive(frame)$;
3: **if** $frame = frameToCorrupt$ **then**
4:     **for** selected Fault Pattern **do**
5:         $CommandHeader(frame)->error() = 1$;
6:         $CorruptNode.Update()$;
7:     **end for**
8: **end if**
9: End.
```

the fault injector component activated, i.e. beacon corruptions occurring, cannot access the medium and becomes inaccessible. The corruption of the frames can be disabled, through the deactivation of the fault injector on the *tcl* script, and the normal behaviour of the network restored at any time.

6.2 Temporal Analysis Component

The temporal analysis component was designed to measure the temporal aspects of received frames, from their duration to the effects of frames received with errors during the network operation. In this paper, we instrumented the temporal analysis component to measure network inaccessibility events. As the occurrence of network inaccessibility is related with the MAC control frames (e.g., beacons), this component was configured to monitor and capture information about such kind of frames.

Let us present an example to characterise how the temporal analysis component works. We use the beacon frame as the frame to be monitored in such example. When the received beacon is corrupted, the temporal analysis component starts a timer to account the related network inaccessibility period. When a new beacon is received successfully, the timer is stopped and the duration of such period is registered.

When the simulation is finished, the temporal analysis component generates a report, regarding all events captured during the simulation. The report is utilised as input to a *gnuplot* script, which transform the raw data within the report to a graphical representation of the captured events.

7 SIMULATING INACCESSIBILITY SCENARIOS

The simulation is defined in an OTcl script (Listing:1) and is carried out in an one-hop star topology, where all the nodes are within the range of each other.

```
1 Event at 0.0   node_(0)
      startWnSCoordinator $beaconOrder
      $superFrameOrder”;
2 Event at 20.0 node_(1) & node_(2)
      startDevice”
3 Event at 20.0 node_(1)
      enableTemporalAccount $Scenario”;
4 Event at 30.0 node_(0)
      startBeaconTransmission
      $beaconOrder $superFrameOrder”
5 Event at 30.0 node_(1) GTS On”
6 Event at 30.0 node_(1) Start Fault
      Injection $Beacon $Rounds”
7 Event at $stopTime ”stop”
```

Listing 1: Example of NS-2 Simulation Script

In the script (Listing:1) we define that the first node to start was the coordinator, specifying the values of its *BO* and *SO* in line 1. After the WnS is established we start the nodes in line 2. Our temporal analysis component is enabled on line 3, given the selected scenario. The periodic beacon transmission is initiated at the coordinator on line 4, taking the *BO* and *SO* as arguments. At line 5 we enable the GTS transmission for the node(1), which means that each time this node have data to transmit will use the GTS mechanism. Finally, at line 6 we start our fault injector to, in this example, corrupt beacon frames for a certain number of rounds.

To simulate a network operation under error conditions, implying the occurrence of network inaccessibility, we configure our fault injector component to generate deterministic faults. For each addressed scenario we set our fault injector to corrupt a specific frame at a given number of times, on a chosen node. The fault injector can corrupt one of each frame type present in the Table: 2.

To achieve the **Single Beacon Frame Loss (SBFL)** scenario we executed the following schedule of Events:

```
Event at 30.0 node_(1) Start Fault
    Injection $Beacon $SBFL”
```

In this scenario the beacon frame will be corrupted *SBFL* number of times (i.e., only one time) at the Node(1), after 30 simulation seconds.

The **Multiple Beacon Frame Loss (MBFL)** occurs when we change the number of corrupting rounds on the fault injector depending on the value that *MBFL* assumes in order to achieve the loss of *nrLost* beacons. The **synchronization loss** is a special case of the MBFL scenario where after the loss of *nrLost* beacons the next beacon is also lost.

Frame type value	Command frame ID	Standard Reference
0		Beacon
1		Data
2		Ack
3		MAC Control Frame
	01	Association Request
	02	Association Response
	03	Disassociation
	04	Data Request
	05	Coordinator conflict
	06	Orphan
	07	Beacon request
	08	Coordinator realignment
	09	GTS request

Table 2: MAC frame types

```
Event at 30.0 node_(1) Start Fault
    Injection $Beacon $MBFL”
```

The **Orphan notification and Coordinator realignment** are achieved when the fault injector corrupts *NOSYNC* beacon frames, corresponding to the current scenario, and the node lose the synchronization. The Orphan notification is observed on the node and the Coordinator realignment is transmitted by the coordinator on response.

```
Event at 30.0 node_(1) Start Fault
    Injection $Beacon $NOSYNC”
```

So that **Coordinator Conflict Detection** can occur, this event has to be forced on the simulator. Once every time a node becomes a coordinator it assumes its ID as the *networkID*, so a coordinator conflict is impossible because every coordinator assumes a distinct ID. To force that event we oblige the coordinator to use the same identifier with the following line.

```
Event at 0.0  node_(1) Coordinator
    Conflict 1”
Event at 0.0  node_(0) Coordinator
    Conflict 1”
```

When the **GTS mechanism** is previously activated from the script, and the node has data to transmit, a GTS Request will occur. This request will be send to the coordinator by the node to perform an allocation of a GTS slot for exclusive transmission time.

```
Event at 30.0 node_(1) GTS On”
```

Simulation Parameters	
NS-2 Version	2.35 updated with GTS, Fault Injector, and Temporal Analysis features
Network Topology	Star Topology
Nodes	7
Traffic	Constant Bit Rate (CBR)
Reception range	15m
Carrier Sense range	15m
Packet Size	8, 67, 127 kbytes
CAP Transmission Type	Direct, using CSMA/CA
CFP Transmission Type	GTS transmission
Transmission/Reception Power	30mW
Beacon	Enabled
Beacon Order	3
Superframe Order	3
Maximum CSMA/CA Attempts	4
Simulation Time	600 seconds

Table 3: Simulation Parameters

8 RESULTS

8.1 Simulation Setup

The network was simulated with seven nodes, where one of these nodes, in the center, was the coordinator. All other nodes are in the radio transmission range of the coordinator, and in the range of each other. A $BO = 3$ was utilised to specify the superframe duration within simulations.

The characteristics of the simulation setup scenario are shown in Table 3. To evaluate the network behaviour under error conditions we applied different error patterns on the simulation through fault injection.

The fault injection can be performed using three different durations: short, medium, or long. The short had the duration of a normal frame transmission, the medium had the duration of the transmission of 3 frames, and the long had the duration of half a beacon.

The traffic generator is set to produce Constant Bit Rate traffic (CBR), which means data frames are transmitted at a constant rate from the nodes to the coordinator. The payload of the sent data was also varied, being the smallest payload of 8 kilobytes, the medium of 67 kilobytes, and the large of 127 kilobytes. The characteristics of the simulation scenario are shown in Table 3.

8.2 Simulation Results

After the environment setup, the simulation was performed to obtain the best and worst case duration of the inaccessibility scenarios.

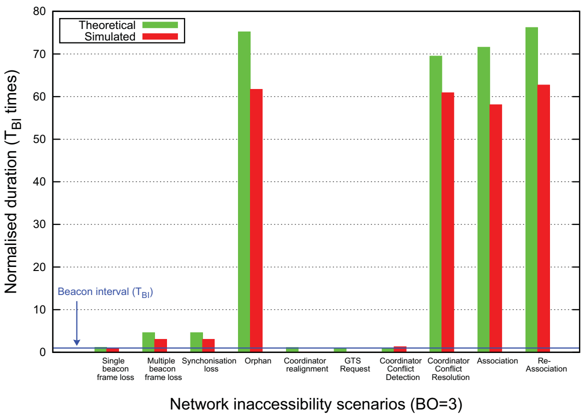


Figure 5: Normalized Inaccessibility Scenarios comparison between Theoretical and Simulated worst case (BO=SO=3 and $\mathcal{T}_{BI} = 0.123$s)

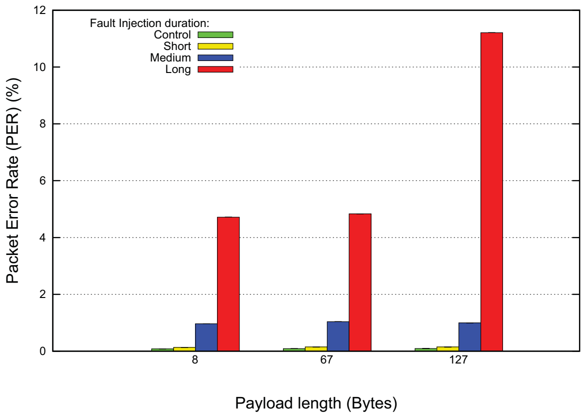


Figure 6: Packet Error Rate comparison between different error patterns

Figure 5 shows the graphic that represents the duration of each network inaccessibility scenario, comparing the results of the previous theoretical study in (Souza and Rufino, 2009) and the obtained simulation results. The results presented in Figure 5 clearly shows that periods of inaccessibility may have a huge duration, which represents a non-negligible impact for networks with temporal restricted traffic.

In the Figure 6 we can observe the Packet Error Rate (PER) between different error conditions. As expected, longer periods of fault injections implies a higher PER. In comparison with the Control frame, a frame that was transmitted without errors, we can see an increasing PER related to the higher periods of fault injection.

The result with the greatest impact is related with the bigger payload, achieves a PER of more than 10%.

An important result of this study is that the influence of network errors, causing periods of inaccessibility in the network, cannot be overlooked if predictability and real-time operation is a system requirement, under the risk of jeopardize the safety and timeliness of the entire system. The effects of network inaccessibility incidents should be controlled by the definition of strategies for the reduction of the periods of inaccessibility, which is achieved in (Souza and Rufino, 2013).

9 CONCLUSION AND FUTURE DIRECTIONS

The paper addressed the behaviour of IEEE 802.15.4 networks in the presence of network errors, leading to periods of network inaccessibility.

Significant improvements and modifications in the NS-2 simulator IEEE 802.15.4 module were presented, which includes the specification of two additional components capable to inject and measure the effects of errors during the network operation. The presence and use of these two component were essential to perform the simulation and evaluation of all network inaccessibility scenarios.

The results obtained by our simulations evidence the relevant temporal aspects of the IEEE 802.15.4 beacon-enabled networks operating under error conditions. Such results can be utilised in the specification of a robust timeliness model of the network, in order to achieve an effective support to real-time operation in IEEE 802.15.4 networks.

Acknowledgements

This work was partially supported: by the EC, through project IST-FP7-STREP-288195 (KARYON); by FCT/DAAD, through the transnational cooperation project PROPHECY; and by FCT, through the project PTDC/EEI-SCR/3200/2012 (READAPT), the Multiannual Funding Program, and the Individual Doctoral Grant SFRH/BD/45270/2008.

REFERENCES

Han, S., Zhu, X., Mok, A., Chen, D., and Nixon, M. (2011). Reliable and Real-Time Communication in Industrial Wireless Mesh Networks. *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*.

Hou, L. and Bergmann, N. (2010). System Requirements for Industrial Wireless Sensor Networks The University of Queensland.

Kandhalu, A. and Rajkumar, R. (2012). Qos-based resource allocation for next-generation spacecraft networks. In *IEEE 33rd Real-Time Systems Symposium (RTSS)*.

Åkerberg, J., Gidlund, M., and Björkman, M. (2011). Future research challenges in wireless sensor and actuator networks targeting industrial automation. In *9th IEEE Internation Conference on Industrial Informatics (INDIN)*.

Saifullah, A., Xu, Y., Lu, C., and Chen, Y. (2011). End-to-end delay analysis for fixed priority scheduling in wirelesshart networks. In *17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*.

Shuai, X. and Zhang, Z. (2010). Research of Real-Time Wireless Networks Control System MAC Protocol. *Journal of Networks*.

Souza, J. L. R. and Rufino, J. (2009). Characterization of inaccessibility in wireless networks - a case study on IEEE 802.15.4 standard. In *IFIP 3th International Embedded System Simposium IESS*.

Souza, J. L. R. and Rufino, J. (2013). Analysing and reducing network inaccessibility in IEEE 802.15.4 wireless communications. In *38th IEEE Conference on Local Computer Networks (LCN)*.

Standard, I. and Society, I. C. (2011). *IEEE Standard for Local and metropolitan area networks, Part 15 . 4 : Low-Rate Wireless Personal Area Networks (LR-WPANs) IEEE Computer Society S ponsored by the*.

Stone, T., Alena, R., Baldwin, J., and Wilson, P. (2012). A viable COTS based wireless architecture for spacecraft avionics. In *IEEE Aerospace Conference*, pages 1–11.

Wang, J., Dong, W., Cao, Z., and Liu, Y. (2012). On the delay performance analysis in a large-scale wireless sensor network. In *IEEE 33rdReal-Time Systems Symposium (RTSS)*.

A.2 Predictability and Adaptation in Middleware for Advanced Safety-Critical Control

A.2.1 Fighting Uncertainty in Highly Dynamic Wireless Sensor Networks with Probabilistic Models

"Fighting Uncertainty in Highly Dynamic Wireless Sensor Networks with Probabilistic Models", L. Marques and A. Casimiro, in 32nd International Symposium on Reliable Distributed Systems (SRDS 2013), Braga, Portugal, Sept. 2013.

This page is intentionally left blank.

Fighting Uncertainty in Highly Dynamic Wireless Sensor Networks with Probabilistic Models

Luís Marques
lmarques@lasige.di.fc.ul.pt
FC/UL*

António Casimiro
casim@di.fc.ul.pt
FC/UL

***Abstract*—Real-time operation in Wireless Sensor Networks (WSNs) is conditioned not only by the current technological level (e.g., limited computing power) but also inherently by the target problem itself: WSNs are required to operate in very open and uncertain environments, subject to external radio interferences, highly dynamic network load, etc.**

Current WSN solutions either provide only best-effort real-time guarantees or make (generally implicit) assumptions on the dynamics of the open environment. These assumptions, in turn, are either very relaxed (i.e., compatible only with undemanding real-time requirements) or very hard to justify.

When dealing with WSNs supporting highly dynamic applications and operating environments (e.g., media streaming, robot control, vehicle coordination, etc.) this problem cannot be ignored. Accordingly, we argue for, and show the efficacy of, using probabilistic models to characterize dynamic WSN QoS, which is the first step to tackle the problem head on. Using our network monitoring technique, we demonstrate that it is possible to meet probabilistic real-time objectives.

***Keywords*-Wireless Sensor Networks; real-time; dependability; adaptation; 802.15.4; non-parametric; lightweight; QoS;**

I. Introduction

Wireless Sensor Networks (WSNs) are used to gather and process information about the state of physical entities in their environment, and may also include control functions. In typical networks the sensor nodes have very limited energy, and therefore the only feasible applications are those where nodes can spend most of their time in low-power sleep modes. These are generally monitoring applications that only infrequently need to send sensor data updates, with transmission periods ranging somewhere between tens of seconds to multiple hours, and correspondingly undemanding deadlines. For these applications it may well be realistically possible to offer even hard real-time guarantees. As long as the WSN has enough redundancy, node faults can be tolerated, while packet transmissions can be retried many times (possibly over redundant network paths) until they succeed, generally well before a timing failure would occur.

We thus understand that, for the currently typical WSN applications, any limitations in real-time guarantees are more a consequence of specific energy budgets than any "intrinsic" inability to offer the desired guarantees. That is, the problem does not derive so much from the open environment the networks operate in, and the uncertainty it implies, but at most from the inability to deal with that uncertainty at a reasonable energy cost. Since we have very loose deadlines and large amounts of redundancy, we should expect that, as long as we were willing to expend energy and we used appropriate techniques, we could recover from faults, be they "internal" (e.g., sensor faults) or "external" (e.g., intermittent wireless interferences), and thus be able to meet even hard real-time objectives.

That is not true for highly dynamic WSNs, such as those employed in teams of cooperating robots sensing their environment. These WSNs or applications thereof involve monitoring and control of very dynamic phenomena in environments where conditions are uncertain and dynamic enough, such that it may not be possible to recover from faults, or from adverse changes in operating conditions, before a timing failure would occur.

For instance, in most WSNs it would likely be unrealistic to assume that a wireless link cannot vary abruptly in quality, or even that it always provides some minimal quality over short timespans. For applications with tight deadlines, such as multimedia streaming, monitoring and control of mobile robots, or coordination of vehicles, this is problematic for a hard real-time design: we cannot dependably design these applications while assuming some static QoS from the network (for some predefined set of resources), and we previously assumed not to have time to recover from faults or QoS changes without incurring in timing failures.

Despite being generally ignored, this problem — that it is not possible to provide hard real-time guarantees in WSNs, in the general case — has been previously recognized in the research community, namely in a recent keynote [1]. But what exactly is the source of the problem? And why can it not be solved by matching the obstacles to real-time operation with the adequate resources and techniques?

The previously referred keynote cited RFC 1925 to wittily motivate the problem: "With sufficient thrust, pigs fly just fine. However, this is not necessarily a good idea." Indeed, there is nothing inherently impossible in meeting hard real-time requirements in WSNs, even demanding ones. Yet, the typically open nature of the environment brings with it the

* This work was partially supported by the EU through the KARYON project (FP7-288195) and the FCT through the Multiannual Funding Program.

possibility of an almost unbounded amount of faults, which in hard real-time designs would have to be dealt with an equally impressive amount of redundancy. For applications with distant deadlines this can probably be achieved with current WSNs, through the exploitation of the large temporal redundancy — at an energy cost, as discussed —, but otherwise this is not practical.

So, our problem is the conjugation of uncertainty from the open environment with real-time requirements that (if strictly interpreted) are incompatible with that uncertainty, under realistic assumptions and acceptable resource usages. Past work has mostly ignored this problem because it is hard to model. Environment uncertainty is (almost by definition) hard to quantify, and its effects are very variable at runtime. Simulations tend to have fairly simplified models, in particular with regards to radio propagation (e.g., circular and symmetric radio coverage models) and external interferences. Therefore, a simulation to show that certain hard real-time requirements can be met under quite specific operating conditions does not address the problem of ascertaining how realistic those conditions are in the first place.

There are two possible solutions. One is to better understand and bound the (currently uncertain) possible operational conditions of these networks. Since the expected trend of WSNs is in the direction of being more general purpose and pervasive, this is probably not a very realistic option. The other solution is to embrace the uncertainty, abandon static environment and fault models, and to use runtime monitoring and adaptation to adapt to actual conditions. This implies that we have to abandon hard real-time models.

In a previous work we explored the possibility of modeling uncertain environments probabilistically, to allow providing probabilistic real-time guarantees, and introduced a technique based on non-parametric statistics [2]. This technique was devised to be computationally lightweight, as would be required for *real-time* monitoring and adaptation in WSNs, but we did not demonstrate its adequacy to WSNs. We only provided a preliminary evaluation of its effectiveness using traces of wired and wireless IP-based networks.

For the adaptation process to be dependable it must be built upon a dependable monitoring technique, which provides an accurate characterization of the current state of the network. Indeed, evaluating an adaptation technique directly can hide problems, as incorrect estimations of the network QoS at time t can be masked by (costlier) adaptations at time $t+1$. Therefore, in this paper our objective is not detailing how to build an adaptation mechanism but to demonstrate that we can accurately monitor the network QoS.

We show, using a combination of a real network and simulations, that dynamic WSN QoS can indeed be modeled probabilistically. Our monitoring technique, although simple and lightweight, provides a useful assessment of current network latencies, in the form "a latency L can be expected to be met with some probability P", thus providing a dependable foundation upon which to build adaptation mechanisms.

More precisely, the contributions of this paper are:

- Showing that, even in simple and well controlled scenarios, WSNs are significantly affected by interferences, whose bounds are uncertain;
- Demonstrating that our monitoring technique can provide probabilistic estimates of the current network latencies, that are accurate under typical network and environment dynamics (always within ± 6 percentage points of the target probabilities, and generally much closer).

To the best of our knowledge, no previous works have demonstrated a monitoring technique that is sufficiently lightweight to be employed at runtime by the WSN itself and yet dependable enough to support real-time adaptation, so our contribution is of practical interest in addition to theoretical.

The paper is organized as follows. In the following section we better define the problem of achieving (probabilistic) real-time operation in WSNs, and give a quick overview of the applied statistical technique. Then, Section III presents the evaluation scenarios. In Section IV we unveil and discuss the results. In Section V we present related work and Section VI concludes the paper.

II. Probabilistic Real-Time Model and Evaluation Method

WSNs are used to sense and collect the state of physical entities. Applications rely on the WSN to create an internal representation of the state of such entities, to be used in monitoring and control functions. Different applications have different requirements regarding how faithful such representation must be, compared with the true state of the sensed entity.

One of the fundamental factors that affect a reliable perception of the physical environment is the delay between sensing a physical entity and the use of that information by the application. This delay is particularly dependent on the one-way network latency, measured between the sensing node and the sink node, to which the state is propagated. Unfortunately, due to the open nature of the environment over which these networks operate, it is not possible to fully characterize (a priori) the bounds of this latency. As such, we rely instead on a runtime characterization of the network latencies.

We consider that the WSN sends update/event messages at defined intervals, pertaining to the state of a monitored physical entity, which the application uses to create or freshen a corresponding internal representation. According to the dynamics of the monitored physical entity and the acceptable margin of error, the application assumes a deadline until which it is supposed to receive the message. If no message is received until the deadline a timing failure (at the network level) or fault (at the application level) is considered to have occurred.

The aim of the adaptation process is to limit the occurrence of timing failures. While, due to the open nature of the operating environment, it is not possible to guarantee that timing failures will never occur, their probability of occurrence can be managed. For instance, if the application can change its mode of operation (application-level adaptation) so that it will tolerate a higher margin of error, then the deadline may be extended, increasing the time during which updates may be received and, therefore, tending to lower the probability of timing failures. Another possibility is to change the mode of operation of the WSN itself (network-level adaptation). For instance, by sending duplicate updates through redundant network routes the probability of timely delivery of an update is likely to increase, since radio interferences and other perturbations are often geographically circumscribed. These adaptations generally imply trade-offs; in these examples, lower Qualities of Service provided by the application (due to the increased margin of error) or increased energy and bandwidth consumption (due to the duplicate traffic). Therefore, it is up to the application to define the best trade-off, by indicating the target probability of timing failure.

We achieve probabilistic real-time operation if we dependably maintain an acceptable probability of timing failure. We refer to the expected probability (or observed frequency) of meeting deadlines as the *coverage*, since it denotes how well the application's assumption of timely updates is being covered, and to our objective of achieving probabilistic real-time operation as one of maintaining *coverage stability*.

In this paper we analyze how well such coverage stability can be achieved in 802.15.4-based WSNs without using network-level adaptation. We do this by continually monitoring network conditions and testing how well the estimated conditions hold (which is equivalent to application-level adaptation) despite the environment dynamics. This tells us how accurate our monitoring is, and thus allows us to evaluate if we have a dependable monitoring solution upon which to build complex adaptation mechanisms.

Network Monitoring

We consider as the monitoring target the network's one-way latency that occurs between the beginning of sending the update/event message and its reception at the sink. One such interval is exemplified in Figure 1, identified as "message latency". It differs from the "packet latency" in the case when packets are lost. Newer updates are considered to supersede old ones, which are discarded if received out of order.

The message latency can be measured if a notion of global time is assumed (there are several clock synchronization algorithms optimized for WSNs [3][4]) and the messages

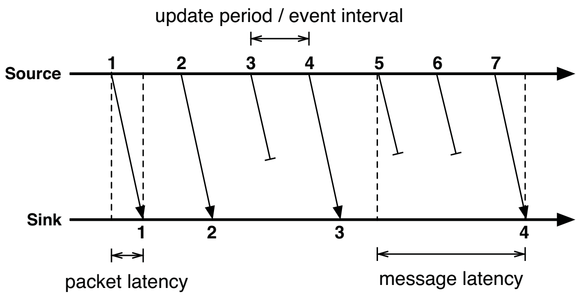


Figure 1: Timing variables (example)

are timestamped before being sent. Otherwise it may be estimated by other methods; for instance by periodically measuring round-trip times.

At the sink the latencies are computed (against the local synchronized clock) and collected, up to a given sample size n, with old sample values being discarded.

Adaptation

Our monitoring evaluation is equivalent to an application adapting to the perceived network conditions (but not the network). At each instant we have some assumed latency, which we expect to be met with some chosen probability (see the evaluation subsection). When we receive a new message we test if the assumed latency was met or not. We also make a new estimate of the latency for the chosen probability, and update our assumption (i.e., the application would adapt to the new latency). At the end of the trial we compute the average coverage, by averaging the number of met deadlines over the total number of messages.

Probabilistic analysis

The non-parametric analysis upon which the adaptation is based is simple and efficient. A copy of the sample is ordered. The first value (the 1^{st} order statistic) is the lowest latency observed, and if it were used as a deadline we would expect a high number of timing failures, since most subsequent latencies would likely be higher. Conversely, the last value of the ordered sample (the n^{th} order statistic) is the highest observed latency, and therefore most subsequent latencies would likely be lower. The previously proposed non-parametric adaptation algorithm [2] very efficiently chooses the latency value given by the order statistic k which best matches the target coverage C, based on the following equation:

$$k = C * (n + 1) \tag{1}$$

This method was chosen because it is very efficient, as required to be executed continually at runtime by the WSN nodes themselves, yet provides an accurate probabilistic view of the latencies that can be used as the basis for complex adaptation mechanisms.

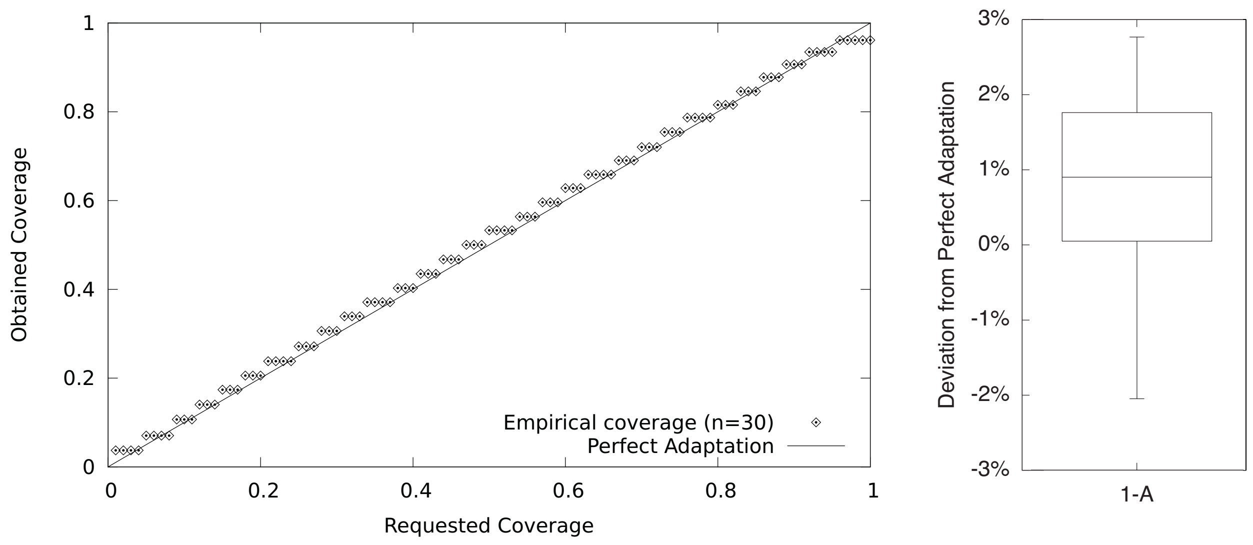


Figure 2: Example of summarizing coverage deviations

Evaluation

According to the relied upon statistical properties, if the WSN never changed between the instant at which the latencies were characterized and the deadline instant, then the observed frequency of timing failures would tend toward the expected value (the mean), with negligible deviations for long-running applications. In this paper we measure the observed deviations from the target coverages to evaluate how robust our monitoring technique is under typical WSN dynamics (i.e., how well our stability assumption holds [2]).

Because the impact of latency variations can depend on the probability with which we want to meet deadlines, we test a wide range of coverages. For instance, if the observed latencies are bimodal and they change in such a way that only one of the two modes varies, then that change will have no impact on adaptations based only on the other mode. As such, for a full characterization of the impact of environment dynamics, we test all coverages between 1% and 99%, in percentage point increments.

To illustrate the deviations between the requested and the achieved coverage for a single scenario we can create a graphic plotting all tested target coverages and the associated empirically obtained coverages. As an alternative, we can summarize the deviations in the form of a box plot graphic, which condenses the information and facilitates comparison of alternative scenarios. Figure 2 exemplifies this process. In this paper we used such box plots[1].

In our example, we observe that the obtained coverages are generally slightly higher than the requested coverages (i.e., they are above the perfect adaptation line). Accordingly, the box plot summarizes this by showing that most (about 75%) of the population (i.e., most of the tested coverages) is above the 0% deviation from perfect adaptation. No outlier (represented as a dot) is present in the box plot because no observed coverage deviated enough (about 3 percentage points from the lower or upper quartile, in this example).

III. Evaluation Scenarios

In this paper we present results for a small real network, as well as various scenarios which were simulated with the omnet++ simulator using the inetmanet package. All networks are based on a 802.15.4 PHY/MAC stack (which is a good trade-off between supporting highly dynamic applications and still conserving some energy). These simulation scenarios are structured as three base scenarios (1, 2, 3), with variants (e.g., 1–A, 1–B, etc). In both the real network and in the simulations we used only a non-beacon mode, with CSMA/CA, to make the network dynamics more easily visible and because we believe that not relying on a centralized beacon would more appropriately reflect the kind of large-scale networks that are envisioned in the literature for the future [5]. We chose the end-to-end latency as the variable used for monitoring/adaptation. Nodes send an acknowledgement (ACK) upon successful frame reception, as specified in the 802.15.4 standard. Periods of 600 seconds of activity were analyzed.

The use of a real network allows us both to (1) show the impact of the uncertain interferences, and (2) to evaluate the monitoring accuracy under real (if simple) conditions, while the simulations allow the evaluation of more complex scenarios. The real network and the simulation base scenarios 1 and 2 were designed to be (as much as possible) compa-

[1]The whiskers are defined to represent, as is common, the most extreme values which still lie within 1.5 times the interquartile range from lower and upper quartile (or the minimum and maximum values, if no outliers are present)

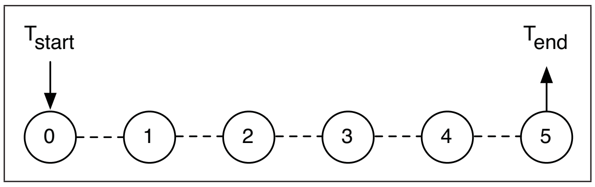


Figure 3: Simulation base Scenario 1 (1–A & 1–B)

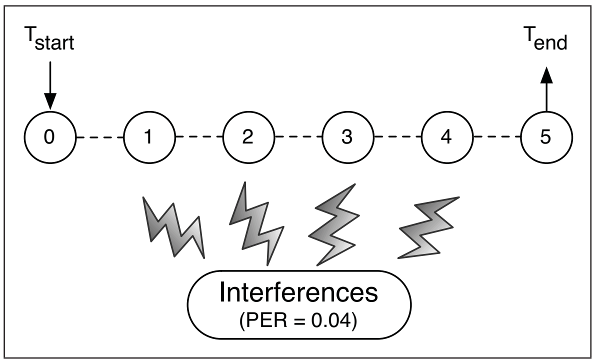


Figure 4: Simulation base scenario 2

rable. This is helpful, since by comparing real results with simulation results we can better understand how realistic and meaningful the simulation results really are.

The simulation base scenarios 1 and 2 and the real network scenario follow a similar structure: all nodes are arranged in a logical line, with nodes only communicating with their immediate neighbors. The first node produces the updates, the last node is the sink to which the updates must arrive, and the intermediate nodes forward these updates.

In the simulation base scenario 1 (see Figure 3), neither movement nor any kind of interferences are introduced, so the variations in end-to-end latency are expected to reflect factors which are intrinsic to 802.15.4, such as the CSMA/CA exponential back-off. The variants 1–A and 1–B have 6 nodes, to allow comparisons with the real network, while the variant 1–C tests the impact of varying the number of nodes. We also test different traffic patterns and different sample sizes. The traffic patterns chosen were a Constant Bit Rate (CBR), to model updates at periodic intervals (e.g., media streaming), and a Poisson / Exponential pattern, to model the detection of natural events, which do not occur at fixed periods (e.g., moving obstacles). Short update intervals were used, reflecting our focus on highly dynamic WSNs and applications.

The simulation base scenario 2 (see Figure 4) introduces a source of interferences, producing a packet error rate of 4%, which is intended to mimic the existence of inferences in real networks operating in open environments.

The real network (see Figure 5) was built using six RCB128RFA1 V6.3.1 evaluation modules from dresden elektronik. These boards contain an AVR ATmega128RFA1

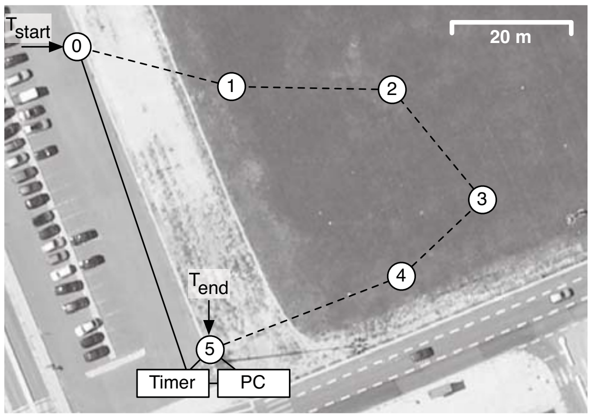


Figure 5: Real network scenario

chip, which integrates an ATmega1281 microcontroller with an AT86RF231 transceiver, compatible with 802.15.4 at 2.4 GHz. The nodes were configured to use the minimum transmission power, by setting the PX_PWR3:0 bits of the PHY_TX_PWR register to one, resulting in a -16.5 dBm power output (0.0224 mW).

As much as possible, we tried the real network to be conceptually equivalent to a (logical) linear network. Two major factors decided the actual geographical distribution of the nodes, which is illustrated in Figure 5.

One factor was that, instead of using a clock synchronization algorithm, we measured the end-to-end latency through wired electronic signals, so we could not have the source and the destination nodes too far apart. Such setup allowed us to better isolate any variation in the latencies, and thus be sure our results reflected actual network and environment dynamics. This was performed by having the source node (node 0) and the sink node (node 5) signal in microcontroller pins when the transmission started and when the update message was received. These signals were timed with real-time equipment, with 16 μs of accuracy, and forwarded to a PC for storage and analysis. The sink node also connected to the PC to share the sequence number of the messages received, so that in cases of multiple consecutive lost messages no ambiguity would arise.

Another factor was the land relief. Although the terrain was fairly flat, we observed that nodes had to be placed at different distances among each other to achieve equivalent signal quality. We tried to obtain a node disposition which best increased signal quality among neighbor nodes and decreased interference among unconnected nodes, which differed significantly from the planned arrangement. We observed that nodes which should ideally not be connected nor interfere among each other had to be within distances where sometimes (but rarely) was still possible to communicate, in order to achieve good link quality between the connected

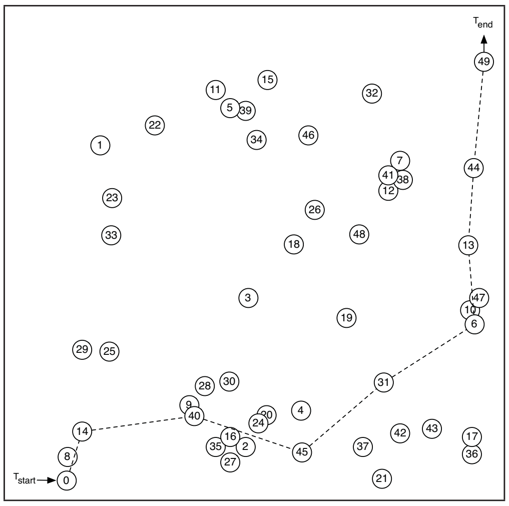


Figure 6: Simulation base scenario 3 (Example)

nodes. This also exemplifies how uncertain the properties of real WSN deployment environments really are.

The real network scenario has similarities with both the simulation base scenarios 1 and 2. While we performed our evaluation in a geographical area and during a time of the day where low 2.4 GHz WLAN activity was present, there is always some amount of interferences. These include thermal noise, microwave ovens, 802.11 WLANs and the non-neighbor network nodes. Therefore, we would expect results from the real network to be somewhere between those of simulation scenarios 1 and 2.

The simulation base scenario 3 departs from the other scenarios to implement a more complex network, as one would expect to find in a real WSN (see Figure 6). It is less artificial, and does not isolate as well the dynamics of the 802.15.4 physical and MAC layers, but allows studying the impact of internal network dynamics, such as node mobility.

In the simulation scenario 3–A the nodes are randomly dispersed in a 50×50 meter area, except for the source and sink nodes which always start at opposing edges. The traffic is routed through the intermediate nodes using the Ad hoc On-Demand Distance Vector Routing (AODV) protocol. The variant 3–B changes the traffic pattern from constant to random with an exponential probability distribution, while the variant 3–C introduces movement of the nodes and 3–D further adds four competing traffic flows. Scenario 3–D is therefore the most complex, and is a good test to evaluate whether the monitoring is accurate, even in the presence of highly challenging dynamics.

These scenarios and the analyzed variants are summarized in Table I. Other variants (e.g., different update intervals or sample sizes) were explored, but only those which provided important insights on the limitations of monitoring effectiveness are here presented and discussed.

Scenario	# Nodes	Update Interval	Sample Size (n)
Real Netw.	6	Constant (0.05 s)	20, 30, 60, 100
1–A	6	Constant (0.05 s)	20, 30, 60, 100
1–B	6	Exp. ($\lambda^{-1} = 0.05$)	30
1–C	5–30	Constant (0.05 s)	30
2	6	Constant (0.05 s)	30, 100
3–A	50	Constant (0.05 s)	30
3–B	50	Exp. ($\lambda^{-1} = 0.05$)	30, 100
3–C (M)	50	Exp. ($\lambda^{-1} = 0.05$)	30
3–D (M+C)	50	Exp. ($\lambda^{-1} = 0.05$)	30, 100

Key
(M): movement of nodes
(C): competing message flows

Table I: Overview of evaluation scenarios

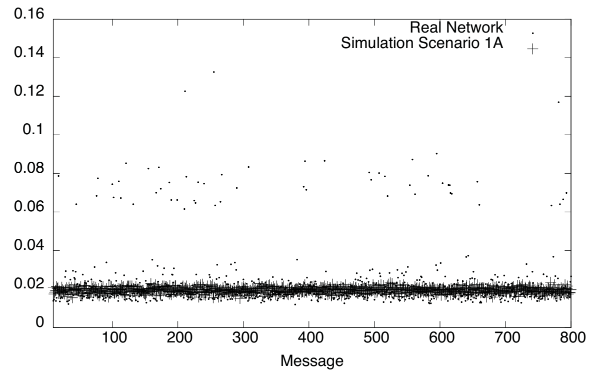


Figure 7: Latencies: real network vs 1–A

IV. Results

Figures 7 and 8 compare the end-to-end message latency of the simulations versus the empirical results of the real network. The latencies of the first 800 messages of each are plotted.

As we can see in Figure 7, in both the simulation and the real network most latencies are distributed around 20 ms, with some random dispersion. The dispersion is due to mechanisms such as the 802.15.4 CSMA/CA during the contention access period (CAP) and the MAC-level transmission retries — up to 7 retries are allowed by the 802.15.4 specification, the default of 3 maximum retries was used in both the real network and the simulations. The simulation has a slightly smaller dispersion (as might be expected, since the simulation does not reflect all realistic sources of interference), but is otherwise very similar (the same scenario simulated using ns-2 showed greater divergence). Because no channel interferences were introduced, almost no messages were lost during the whole simulation period, despite the high update data rate. As such, the simulation does not exhibit latencies around 70 ms (1 lost update) and 120 ms (2 lost updates) like the real network does.

The simulation scenario 2 more closely matches the empirical results of the real network, as is visible in Figure 8. Due to the introduction of interferences, packets were

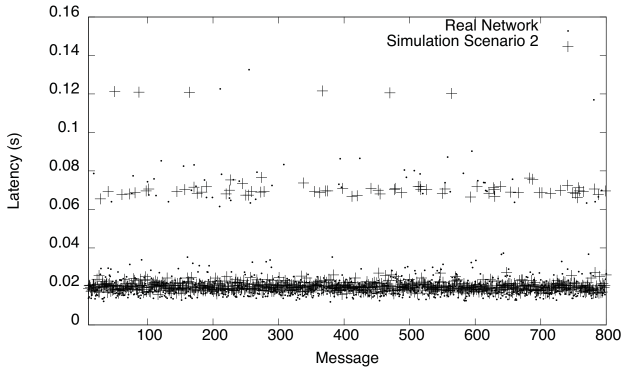


Figure 8: Latencies: real network vs 2

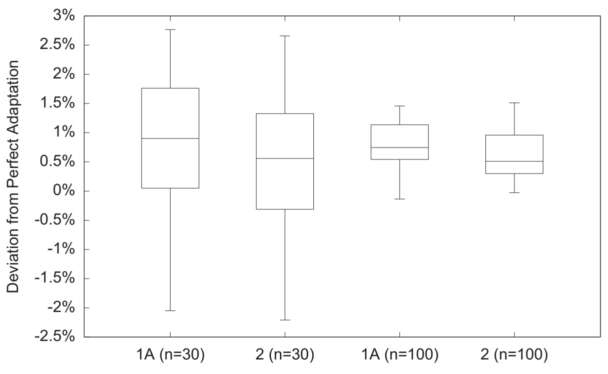


Figure 9: Scenario 1–A vs scenario 2

dropped, creating a probabilistic pattern of three different latency bands, caused by application-level "retries" (superseding updates), with a progressively lower occurrence of latencies in each band as the number of retries increases.

There are three important conclusions regarding these results. The first is that, despite the real network being very simple and small, and despite our efforts to reduce all sources of interference as much as possible, we needed to include an explicit source of interferences in the simulation to accurately replicate the empirical results. This stresses the real impact of the open environment and of complex radio interference patterns, which are very hard to characterize and bound for all realistic scenarios that a real network might face. The second is that our dispersion of latencies is indeed quite random, and thus compatible with the assumptions of our probabilistic approach. The third is that the simulation results agree well with those of the real network, which establishes a baseline of trust in the results from the other simulations we present.

We can make a first assessment regarding the monitoring effectiveness in small and linear simulated networks by inspecting Figure 9. We can note that the coverage (i.e., the inverse of the observed frequency of timing failures) never exceeded the target by more than 3 percentage points, nor ever underachieved it by more than 2.5. Thus, as preliminary finding we could expect that: (1) even without using network-level adaptation to meet QoS targets, our monitoring solution would be accurate enough to support application-level adaptations that should come within about 3 percentage points of the best performance/timeliness trade-off for even the most adverse coverage; (2) that when also using network-level adaptation we would only have to expend a small amount of resources to compensate for the slightly inaccurate QoS estimates.

We also notice that despite scenario 2 introducing interferences which are not present in scenario 1–A, the monitoring effectiveness is not worse in scenario 2, and even improved slightly (the median is closer to the target).

Another preliminary conclusion that can be taken from Figure 9 is that larger sample sizes (increased n) tend to lead to more accurate latency estimates, since the box plots for the n = 100 have a smaller amplitude than those for n = 30, reflecting more accurate monitoring.

From these results we can tentatively conclude that increasing the complexity of scenarios (i.e., introducing factors of temporal uncertainty) does not in itself create an impediment to obtaining accurate monitoring. On the contrary, we observe that the results may even improve. The explanation for this outcome may be understood by analogy to the central limit theorem (CLT) from probability theory. In the CLT, as we add an increasing number of independent random variables (with finite mean and variance) the average starts to approximate a normal distribution, even if the variables were not normally distributed. Likewise, as we add various factors which affect the latency in unpredictable ways the result may start to be, overall, more predictable.

A similar effect can be seen in Figure 10, which compares the scenario 1–A (a constant rate update stream) with scenario 1–B (updates at exponentially distributed intervals). While the amplitude of the box plots for these scenarios is similar, the median deviation from the target coverage for the scenario 1–B is much closer to ideal of 0 percentage points.

Figure 11 more comprehensively explores the effect of increasing the sample size, showing the summarized results of scenario 1–A. We see that by gradually increasing the sample size we gradually improve the monitoring accuracy, although with progressively smaller gains. This diminishing of returns is not unexpected, since the dynamics of the WSN can endanger the freshness of the older sample values, counterbalancing their benefit for assessing the current state of the network's QoS.

Figure 12 presents a similar analysis, but performed using the latencies of the real network. The most important conclusion that should be taken from Figure 12 is that the monitoring is highly effective under a real scenario, with

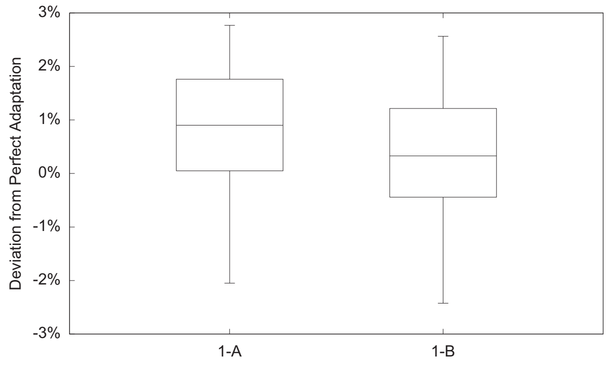


Figure 10: Scenario 1–A vs 1–B

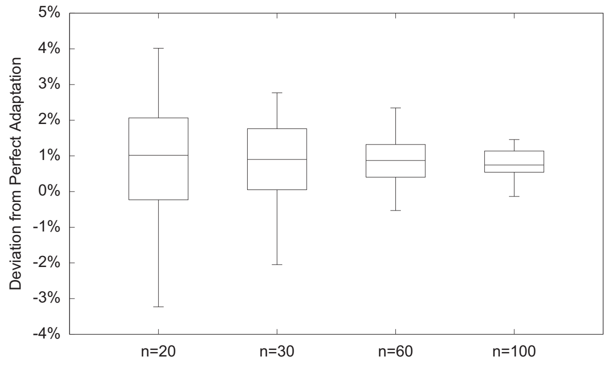


Figure 11: Scenario 1–A — Adaptation effectiveness at different sample sizes (simulated)

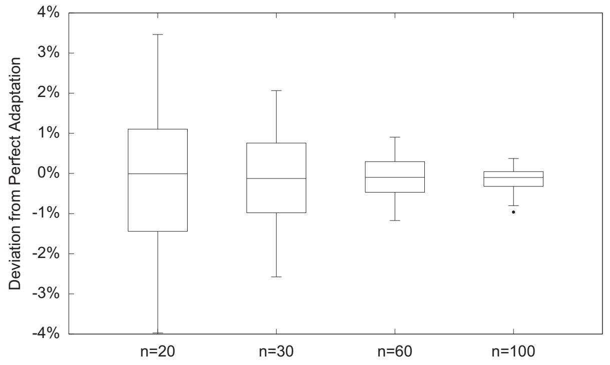


Figure 12: Real Network — Adaptation effectiveness at different sample sizes

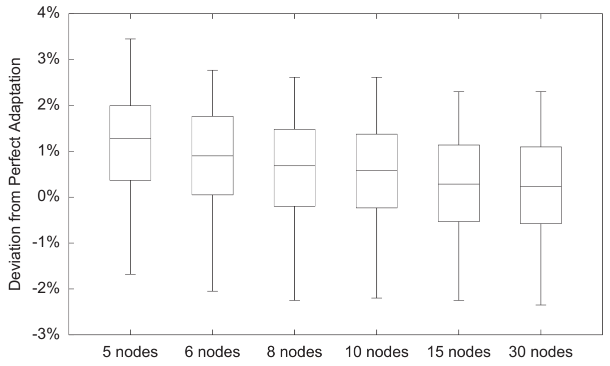


Figure 13: Scenario 1–C — Effect of number of hops in adaptation effectiveness (simulated)

small measured deviations from the target coverage. First, even for the small sample sizes of $n = 20$ and $n = 30$ we obtain good results, even better than those obtained under simulation with scenario 1–A. Second, by increasing the sample size we are able to come very close to the target coverages; for $n = 100$ we always come within less than 1 percentage point, which implies that our monitoring strategy estimated very accurate latencies for all target coverages. Third, there is slightly less of an effect of diminishing returns, compared with simulation scenario 1–A. This is most likely due to the increased randomness of the real network, compared with the simulation.

Another important consideration when exploring the dependability of probabilistic QoS models is the existence of scale effects. To address this question, Figure 13 scrutinizes the impact to the adaptation of the number of hops that must be traversed between the update/event producing node and the sink node. The number of nodes of a linear network (scenario 1–C) is increased between 5 and 30, corresponding to 4 and 29 hops.

We see in Figure 13 that not only does the adaptation not degrade with the increase in the number of hops that need to be transversed but that the median deviation actually improves. That is, while the average latency will increase as the number of hops rises we can still effectively estimate probabilistic bounds for dynamic latencies. This result strongly supports our hypothesis that the proposed monitoring strategy can dependably support adaptation mechanisms for probabilistic real-time operation in real networks, under a variety of different circumstances; from small networks to large networks.

While the simplicity of the simulated scenarios 1 and 2, as well of the real network, provide good testing grounds for a variety of different experiments (which highlight particular properties and important corner cases), they do not reflect the complexity of realistic larger-scale networks. The base simulation scenario 3 addresses the effectiveness of monitoring in more complex and realistic networks.

The four concrete scenarios derived from the base scenario 3 address increasingly complex situations. First we start with static nodes which communicate at a constant rate (scenario 3–A). Then we change to a probabilistic interval model (scenario 3–B). To this we add the movement of network nodes (including the source and sink nodes — scenario 3–C) and, finally, we incorporate competing message

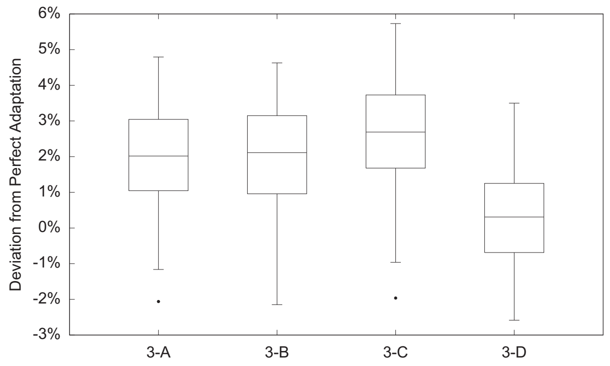


Figure 14: Scenarios 3–A, 3–B, 3–C and 3–D

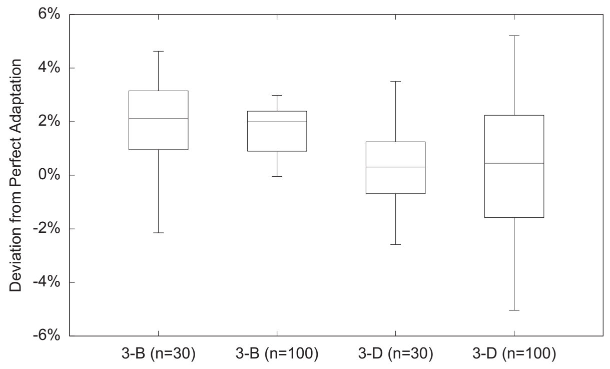


Figure 15: Scenario 3 — Adaptation effectiveness at different sample sizes

flows, which will further disrupt the latencies pertaining to the measured flow of event update messages (scenario 3–D). Figure 14 presents the adaptation results for all of these scenarios.

In general, we observe that even under complex scenarios, and with a small sample size (n = 30), our monitoring strategy is still fairly accurate, as all of the simulated scenarios never deviate more than +6 or -3 percentage points from the target coverage, and in most cases quite less.

More particularly, we notice that, as happened in the real network and in the simulated base scenario 1, the transition from a constant (scenario 3–A) to a random message interval (scenario 3–B) did not disrupt the monitoring. Comparing the box plots for scenarios 3–B and 3–C we witness a very slight decline in monitoring accuracy, as movement is introduced in the network. Contrarily, introducing competing messages flows on top of node movement, as was done in scenario 3–C, again improved the monitoring accuracy, with lower deviations from the target coverage.

Finally, we present and discuss the effect of increasing the sample size in the more realistic base scenario 3. Figure 15 shows the results.

We see that when no movement was present (scenario 3–B) the increase in sample size produced more consistent monitoring results, with less variation in the deviations. On the other hand, when in the presence of movement (scenario 3–C) the increased sample size worsened the monitoring effectiveness. This makes sense, and also helps explain the behaviors exhibited in Figure 15. Due to the dynamics created by the node movement, older sample values no longer accurately reflect the current state of the network, making for less accurate estimates of the latency that matches the chosen coverage.

V. Related Work

One of our long-term research objectives is to improve the dependability of real-time WSNs. Therefore, our work is in general related to real-time and QoS provision in the context of WSNs, but more specifically with mechanisms and approaches to deal with the uncertainties affecting the timeliness of communication.

Several works address the problem of providing QoS and real-time guarantees in WSNs, which can be tackled at different levels, including the network architecture, sensor node hardware, and protocols at the different layers of the communication stack (MAC, network, transport, application) [6]. One particularly relevant and important approach for dependable and real-time operation is to exploit the intrinsic redundancy existing in large-scale WSNs. While the deployment of many nodes can be a cause of interferences and increased uncertainty, it also allows for multi-path protocols to be considered [7]. A different approach is to differentiate classes of traffic, assigning higher priorities to urgent or real-time data. For instance, the RAP architecture [8] takes into account routing distances when assigning priorities and routing packets, to reduce deadline misses for packets far-away from their destination.

One of the fundamental problems for predictability and dependability is the interference that derives from the crowded electromagnetic spectrum [9]. In this paper we considered several scenarios involving interference, to observe the behavior of communication from a temporal perspective, and to investigate the possibility of correctly characterizing this behavior with our probabilistic analysis approach. The work presented in [10] also considers the impact of interference processes in WSNs. However, their goal is to characterize the interference itself rather then its effects on timeliness, and to propose solutions to reduce or avoid this interference, and thus achieve a better overall behavior. The detection of radio interferences, as done in [11], can be relevant from a communication reliability perspective, while the possibility of assessing the wireless link quality [12] can, in general, be useful for achieving dependable adaptation.

There is competing work in probabilistic models of network QoS [13] but not with a focus on being lightweight (and thus allowing runtime monitoring from the network nodes themselves), with estimations taking several seconds on a modern computer.

Finally, experimentally observing, measuring and characterizing the behavior of WSNs is not easy, as it usually involves the need of special setups and measurement equipment [14]. Our approach was to use additional measurement equipment (a dedicated network node, plus a PC) to perform these measurements. On the other hand, experiments are usually done for specific application environments (e.g., industrial ones [15]) or using existing testbeds (e.g., one that was previously used to validate a routing protocol [16]), whereas our objective in this case was to create scenarios that would allow us to compare the results of a real setting with those of simulated systems.

VI. Concluding Remarks

Dependably achieving real-time operation in wireless and open environments is a hard problem. The nature of open environments makes hard real-time guarantees either extremely pessimistic or unreasonably expensive, and further makes this a hard issue to research, since it is not clear what typical bounds can be safely assumed. Indeed, our results showed that in real networks, even in simple and well controlled scenarios, interferences have a significant impact on the results.

We argued that an approach based on runtime monitoring and adaptation can overcome this limitation, and that establishing the dependability of a monitoring technique such as ours is an essential step to take before building complex adaptation mechanisms, which might hide inaccurate monitoring estimates, at a steep cost.

While in an open environment we cannot assume any strict bounds of the network and environment dynamics, we saw that under typical conditions our monitoring approach accurately estimates the current network conditions, providing the latencies that should be assumed to meet deadlines with the desired probability. In particular, we saw that the estimates allowed us to always come within ± 6 percentage points of the desired timeliness probabilities, and generally much closer. This monitoring approach thus provides a solid base upon which to build dependable adaptation techniques.

References

[1] Decotignie, J.D.: The real-time and wireless sensor networks: are they compatible? In: 24th Euromicro Conference on Real-Time Systems, ECRTS 2012. (2012) Keynote address.

[2] Marques, L., Casimiro, A.: Lightweight dependable adaptation for wireless sensor networks. In: Proceedings of the 30th IEEE International Symposium on Reliable Distributed Systems Workshops, 4th International Workshop on Dependable Network Computing and Mobile Systems (DNCMS 2011), Madrid, Spain (oct 2011) 26–35

[3] Li, Q., Rus, D.: Global clock synchronization in sensor networks. IEEE Transactions on Computers **55**(2) (2006)

[4] Yoon, S., Veerarittiphan, C., Sichitiu, M.L.: Tiny-sync: Tight time synchronization for wireless sensor networks. ACM Trans. Sen. Netw. **3**(2) (June 2007)

[5] Corke, P., Wark, T., Jurdak, R., Hu, W., Valencia, P., Moore, D.: Environmental wireless sensor networks. Proceedings of the IEEE **98**(11) (nov. 2010) 1903 –1917

[6] Akyildiz, I.F., Melodia, T., Chowdhury, K.R.: A survey on wireless multimedia sensor networks. Computer Networks **51** (2007) 921–960

[7] Li, S., Neelisetti, R., Liu, C.: Efficient multi-path protocol for wireless sensor networks. International Journal of Wireless and Mobile Networks (IJWMN) (Jan 2010)

[8] Lu, C., Blum, B., Abdelzaher, T., Stankovic, J., He, T.: Rap: A real-time communication architecture for large-scale wireless sensor networks. In Real-Time Technology and Applications Symposium (2002)

[9] Zhou, G., Stankovic, J.A., Son, S.H.: Crowded spectrum in wireless sensor networks. In: Proceedings of Third Workshop on Embedded Networked Sensors (EmNets. (Jan 2006)

[10] Liang, C.J.M.: Interference characterization and mitigation in large-scale wireless sensor networks (2011)

[11] Zhou, G., He, T., Stankovic, J.A., Abdelzaher, T.: Rid: Radio interference detection in wireless sensor networks. In: in INFOCOM. (2005)

[12] Baccour, N., Koubía, A., Ben Jamía, M., do Rosário, D., Youssef, H., Alves, M., Becker, L.B.: Radiale: A framework for designing and assessing link quality estimators in wireless sensor networks. Ad Hoc Netw. **9** (September 2011) 1165–1185

[13] Wang, Y., Vuran, M.C., Goddard, S.: Cross-layer analysis of the end-to-end delay distribution in wireless sensor networks. IEEE/ACM Trans. Netw. **20**(1) (February 2012) 305–318

[14] Ageev, A., Macii, D., Petri, D.: Experimental characterization of communication latencies in wireless sensor networks. In: Proceedings of the 16th IMEKO TC-4 International Symposiu ”Exploring New Frontiers of Instrumentation and Methods for Electrical and Electronic Measurements” and 13th Workshop on ADC Modelling and Testing, IMEKO, A&T (2008) 258–263

[15] Bertocco, M., Gamba, G., Sona, A., Vitturi, S.: Experimental characterization of wireless sensor networks for industrial applications. Instrumentation and Measurement, IEEE Transactions on **57**(8) (aug. 2008) 1537 –1546

[16] Pavkovic, B., Theoleyre, F., Barthel, D., Duda, A.: Experimental analysis and characterization of a wireless sensor network environment. In: Proceedings of the 7th ACM workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks. PE-WASUN ’10, New York, NY, USA, ACM (2010) 25–32

A.3 Vehicular Coordination Algorithms

A.3.1 Vehicular Coordination with a Safety Kernel in the Gulliver Test-Bed

"Vehicular Coordination with a Safety Kernel in the Gulliver Test-Bed". O. M. Ponce; T. Petig, and E. M. Schiller, Technical Report. 2013.

This page is intentionally left blank.

Vehicular Coordination with a Safety Kernel in the Gulliver Test-bed

Oscar Morales Ponce Thomas Petig Elad M. Schiller

1 Introduction

The vehicle coordination problem requires making sure that vehicles will not have accidents, for example, when crossing intersections, or run into objects, such as road fences. Current driver-less vehicle implementations that operate on the road, e.g., Google car, do not consider cooperative path plans. As a result, the risk exclusion of causing severe damage with sufficient certainty requires longer headways. The *headway* is the distance, expressed as the time, that the vehicle requires to perform safety operations. Recent developments in the automotive domain, as well as in communication networks, follow a more structured approach regarding information dissemination about road hazards, vehicle whereabouts and cooperative driving. We follow this structured- and cooperative-driving approach and consider a system in which accidents are avoided via periodic agreement on cooperative service level and recalculation of the path plans.

Vehicles use the communication network for requesting the information such as localization, speed, intention, of each other vehicle. Once all vehicles receive the information, each determines safe paths for all level of services that the vehicle support before agreeing on a cooperative one. These paths ensure a safe driving for any coherent, yet possibly changing, service level. For example, in the case of cooperative functionalities such as adaptive cruise control and vehicle platooning, the plan considers the safe distance headways for each of the possible cooperative service level. Note that the cooperative service level may change during the plan term and, in turn, cause the vehicles to adjust accordingly their safe distance headways and vehicle separation.

Cooperative driving is not only be dependent on the expected vehicle paths and the local sensory information but also the data validity. Thus, these functionalities are based on periodic information exchange with nearby vehicles, and validity assessment before deciding on the cooperative service level. One may consider systems in which vehicles have individual service levels that based on their currently received information validity. The safety analysis in this case must keep track of all the possible service level combinations of nearby vehicles and timely decide on the joint and safe driving behavior. We use a cooperative service level evaluator in a way that guarantees safe transition between two different cooperative service levels, i.e., autonomous and cooperative. The system lets all nearby vehicles coherently refer to the cooperative service level.

Changes to the data validity level are not the only reason why the cooperative service level may change. The system ability to agree on the cooperative service level is based on inter vehicle communications and thus it is prone to faults. During the recovery period, the cooperative service level evaluator may decide on a cooperative service level that is lower than any local service level.

2 System Description

We consider a system that includes n autonomous and cooperative vehicles, *members*. All vehicles are identical and move on an available map of the environment. Each vehicle is an autonomous entity with the capabilities of *sensing*, i.e. ability to perceive some parameters of the environment, *communication* - ability to receive and transmit information to other vehicles, *computation* - ability to process the obtained data, and *mobility* - ability to control its motion speed and steering within the environment. We assume the availability of a map that defines a road network and that the vehicles have the capability of following the lanes and changing lanes. Further, vehicles have unique identifiers, which we simply refer to as $id \in members$.

We consider a synchronous system with *rounds* of exactly $ROUND$ time units. Throughout the document, we refer to $round_{now}$ as the current round. Indeed, we assume that each vehicle has access to a synchronized clock with a known synchrony bound of $LOCAL\_BOUND\_SYNCHRONY$ milliseconds. Moreover, we consider clock synchronization algorithms among the vehicles with a known synchrony bound of $GLOBAL\_SYNCHRONY\_BOUND$ milliseconds. We choose the $ROUND$ value according to the synchrony bounds and the time it takes to perform key communication operation such as agreeing on the cooperative service level and the refresh rate of the local dynamic map.

We illustrate the system software components and their interaction in Figure 1. The primary component is the safety kernel, which monitor that data validity flows (Section 3). Our simplified version of the Local Dynamic Map (LDM) provides location information to the components that are responsible for driving management whether the system employs autonomous or cooperative driving.

3 Interface with the Safety Kernel

The Safety Kernel is a set of timely-behaved safety-related components that are responsible to perform monitoring and management tasks that ensure the required functional safety goals, see KARYON's deliver D4.2 (first report on safety kernel definition V1.0). We next describe the data validity flow and interaction with the safety kernel. We detail and interaction with the safety kernel after listing interface primitives in the Figure 2.

One of the most related kernel components is the local service level evaluator. It receives the data validity from both onboard and remote sensors about the location and behavior of nearby vehicles, as well as other functional components that need to report about failures, such as the validity of mechanisms for cooperative service level evaluating. The safety kernel supports these operations using the primitives $WRITE\_VALIDITY\_DATA$ and $READ\_VALIDITY\_DATA$ for allowing components to send their validity data to be checked and evaluated by the safety kernel, and respectively, for the safety kernel to receive this information. The output of the local service level evaluator is the input of the cooperative service level evaluator. The safety kernel supports these operations using the primitives $WRITE\_LOCAL\_MAXIMUM\_LOS$ and $READ\_LOCAL\_MAXIMUM\_LOS$ for allowing safety kernel to send, and respectively, receive its local service level to the cooperative service level evaluator. The output of the cooperative service level evaluator then is sent to the safety manager. The safety kernel supports these operations using the primitives $WRITE\_COOPERATIVE\_LOS$ and $READ\_COOPERATIVE\_LOS$ for allowing safety kernel to send, and respectively, receive its cooperative service level to the cooperative service level evaluator. The safety manager periodically adjusts the performance level of functional components by making available the effective service level of all cooperative functionalities. It also reconfigures and selects the respective components, whenever required. The safety kernel supports these operations using the primitives $WRITE\_PERFORMANCE\_LEVEL$ and $READ\_PERFORMANCE\_LEVEL$ for allowing safety kernel to send, and respectively, receive its effective performance level to the cooperative service level evaluator.

The information flow of data validity also includes notifications on component timing failures. This specifically includes all network-centric mechanisms, such as localization, neighborhood discovery and cooperative service level evaluator. The safety kernel architecture requires each of these functionalities to periodically, and at a predefined minimal rate, send a heartbeat by calling the primitive $WRITE\_TFD\_DATA$ and informing the safety kernel that progress is being made and that its planned schedule is being fulfilled. For each of these components, the timing failure detector (TFD) component at the safety kernel checks the heartbeats timely reception.

We assume that the safety kernel uses data component multiplexers, e.g., a multiplexer that switches the $PilotChannel$ and has the inputs $CoopDriveId$ and $AutoDriveId$. This multiplexer takes care of switching from the cooperative driving to the autonomous driving.

4 Local Dynamic Map

This component provides location information about the system objects, such a vehicles and roads. The aim here is not to provide an implementation of `ETSI TR 102 863`, rather we would like to have the most essential elements that are critical to the demonstration of the safety kernel. For example, we do not consider the complete environment, such as the weather, roadside units and obstacles, but we do consider the position of intersections, lanes, and the vehicles.

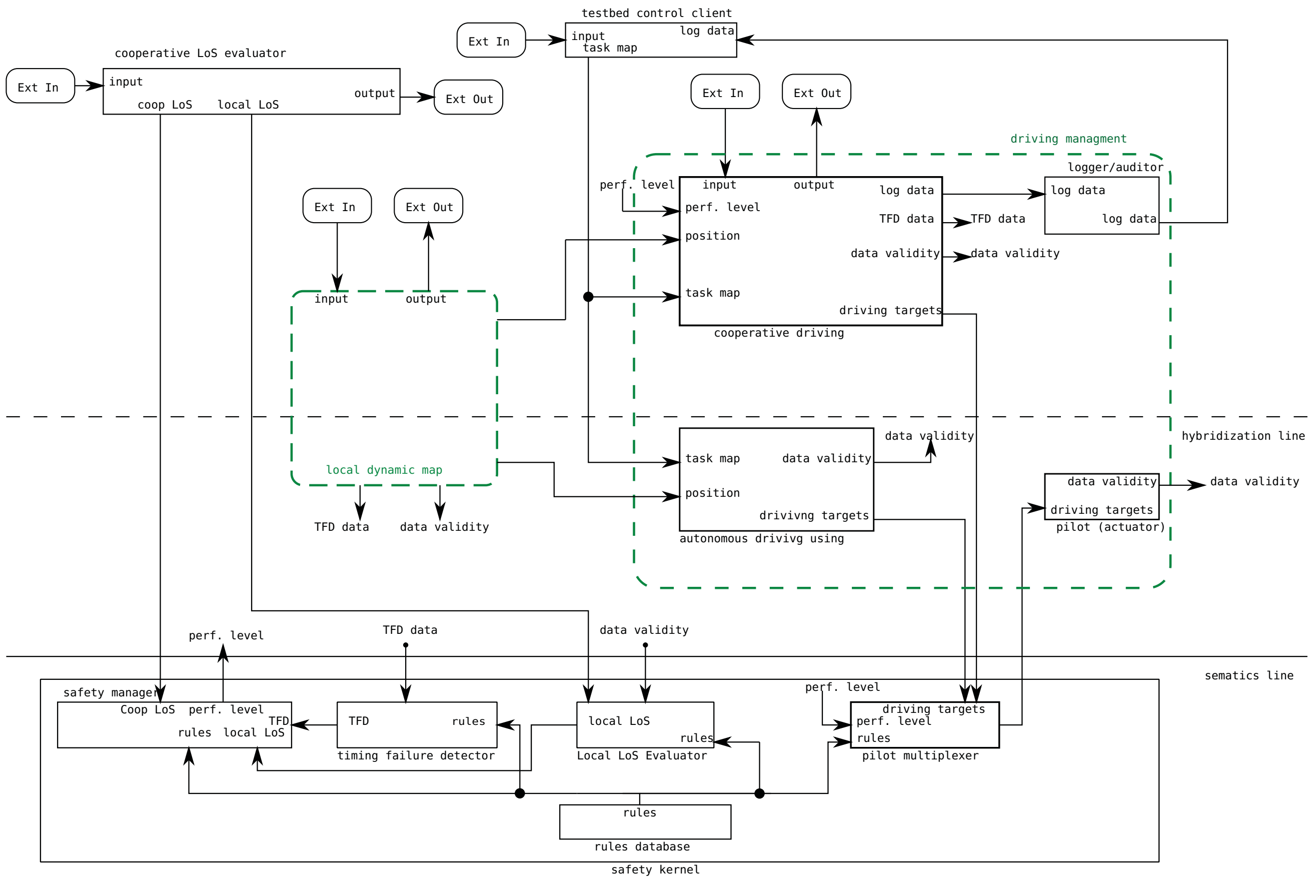


Figure 1: System Architecture for Vehicular Coordination with a Safety Kernel in the Gulliver Test-bed. The internals of the local dynamic map are presented in Figure 3

constants:
$Validity \in \{LowValidity, HighValidity\}$
$Channel \in \{PilotChannel, PositionChannel\}$: multiplexer channels
$PilotChannelId \in \{CoopDriveId, AutoDriveId\}$: valid input ids for the $PilotChannel$
$PositionChannelId \in \{NetBasedId, NetFreeId\}$: valid input ids for the $PositionChannel$

interface:
$WRITE\_TFD\_DATA(id)$ id is the identification of the caller module
$WRITE\_VALIDITY\_DATA(id, validity)$ id is the identification of the caller, $validity$ the validity to report.
$WRITE\_APP\_OUTPUT\_DATA(channel, input, data)$:writes $data$ to channel $channel$ with input $input$
$READ\_APP\_INPUT\_DATA(channel)$:returns the effective data on a channel $channel$
$READ\_PERFORMANCE\_LEVEL()$: returns the current effective Level of Service
$WRITE\_COOPERATIVE\_LOS(LoS)$: sends $LoS \in \Sigma$ as cooperative Level of Service to the Safety Kernel
$READ\_LOCAL\_MAXIMUM\_LOS() \in \Sigma$: returns the maximal local Level of Service

Figure 2: Interface to the Safety Kernel of vehicle $v_i \in V$

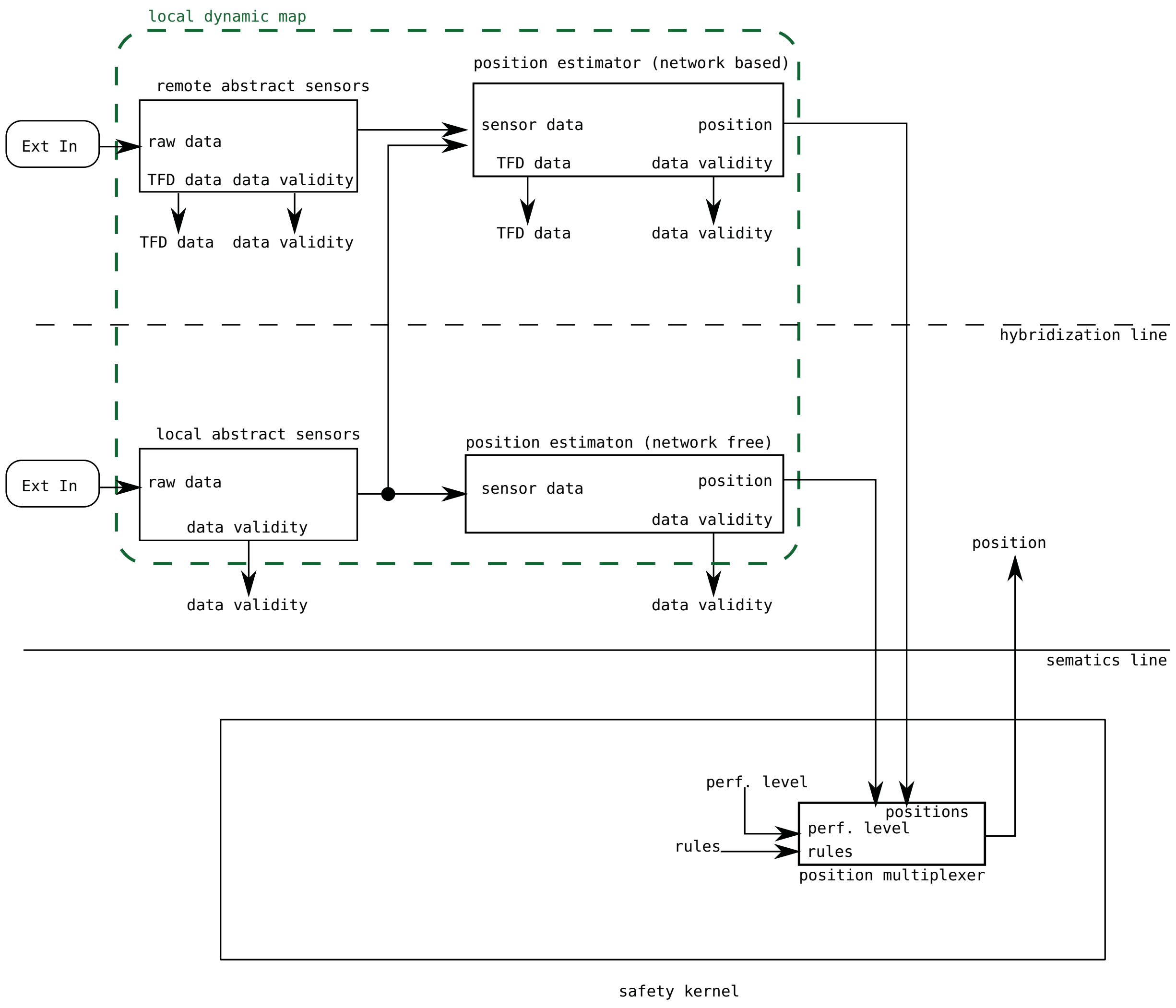


Figure 3: The position estimator uses the remote abstract sensors to create a position database.

We note that the latter objects requires periodic updates regarding their position. The static elements of the map are provided at the start of the experiment. This information includes the entire *task map*, which is a scenario description that is based on a discrete graph, and an instruction of which test case algorithm we are to demonstrate, see Section 7. Each vertex in the task map has predefined coordinates so that the graph's edges encodes roads, lanes, intersection, etc. Using that task map, the position estimator can point out the vehicle whereabouts within a bounded error. The algorithms in Section 7 use this position estimation to discover the closets vertices on the task map and any additional information that is required for the path planning.

The position estimator implementation is divided into two parts, the local and the global position estimator. The local position estimator uses only local sensor with predictable timing for the position estimation of the vehicle and thus itself can be implemented in a real-time manner. The data output includes for any given set of system vehicles, *member*, the vector $vehicle = [\langle id, velocity = \langle speed, position, heading\rangle \rangle]_{id \in members}$.

The abstract sensor combines the data of an (abstract) onboard range sensor and an (abstract) onboard odometer sensor. Ultra wide-band radio communication is used for indoor and outdoor localization. Such a radio can determine the distance to any other radio in range with a ranging request. The test-bed uses three radios, called *anchors*, with a known fixed position. The (abstract) onboard range sensor sends ranging requests to the anchors in a periodic manner

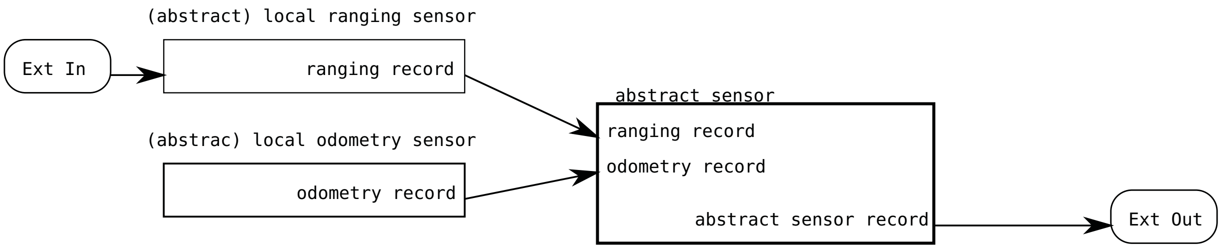


Figure 4: Local ranging and odometer information is joint in the abstract sensor. This information is exchanged with the corresponding abstract sensors on the nearby vehicles.

and returns the result, i.e., anchor id and distance, expected error and measurement time. Vehicles exchange their localization information and use that for discovering the location of all nearby nodes.

The (abstract) onboard odometry sensor facilitates the motor controller to get information about speed and steering angle. Additionally, it can use an inertial measurement unit (IMU).

The Gulliver test-bed based its localization mechanisms on external reference and the exchange of this information among the vehicles for the sake of neighborhood discovery. The position database contains a position estimation for all vehicles from which sensor data has been received. The position database uses a Bayes filter for each vehicle $v_i \in V$ to estimate the probability density function of v_i's position, speed and heading.

5 Pilot Actuator

Algorithm 1: Pilot

```
1 upon  $Timeout\,(k)$ ;
2 begin
3     Let $target := READ\_APP\_INPUT\_DATA(PilotChannel)$;
4     Let $state = (x, y, speed, heading) := CurrentPositionEstimation()$;
5     Let $path := ComputePath(state, target)$;
6     $WRITE\_TFD\_DATA()$;
```

The Pilot computes the actual driving actions that have to be done. It gets its input from the pilot multiplexer. The input is, depending on the service level, either from the cooperative driving or from the autonomous driving. The input contains the actual position the plane where the vehicle is the positions the vehicle is aiming to and the estimated arrival time at these positions.

6 Management of Cooperative and Autonomous Driving

The key objective of the safety kernel architecture is to provide system solutions for predictable and safe coordination of smart vehicles that autonomously cooperate and interact. This model for cooperative driving provides instructions that the pilot actuator can use whenever the safety kernel decides on a cooperative service level. We show how the proposed system and safety kernel architecture address each test case. The system considers the models for providing local dynamic map and vehicle-to-vehicle communication, as well as the future horizon of events.

In cooperative driving, vehicle must be aware of the localization, speed, acceleration, intention, etc, of each other within a given horizon. For the vehicular coordination, we propose a variant of the model proposed by Ando *et al.* [4] that includes consensus. In each round, the vehicles execute four phases: (1) *Observe* the environment for a fixed period,(2) *Compute* a local plan according to a deterministic algorithm, (3) *Agree:* on the cooperative service level and (4) *move.* For the sake of presentation simplicity, we structure the pseudo-code of the computation phase using two parts: one for the *Autonomous Driving* and one for the *Cooperative Driving*. We let the safety kernel determine the system perform level and select accordingly which output should be direct to the pilot actuator.

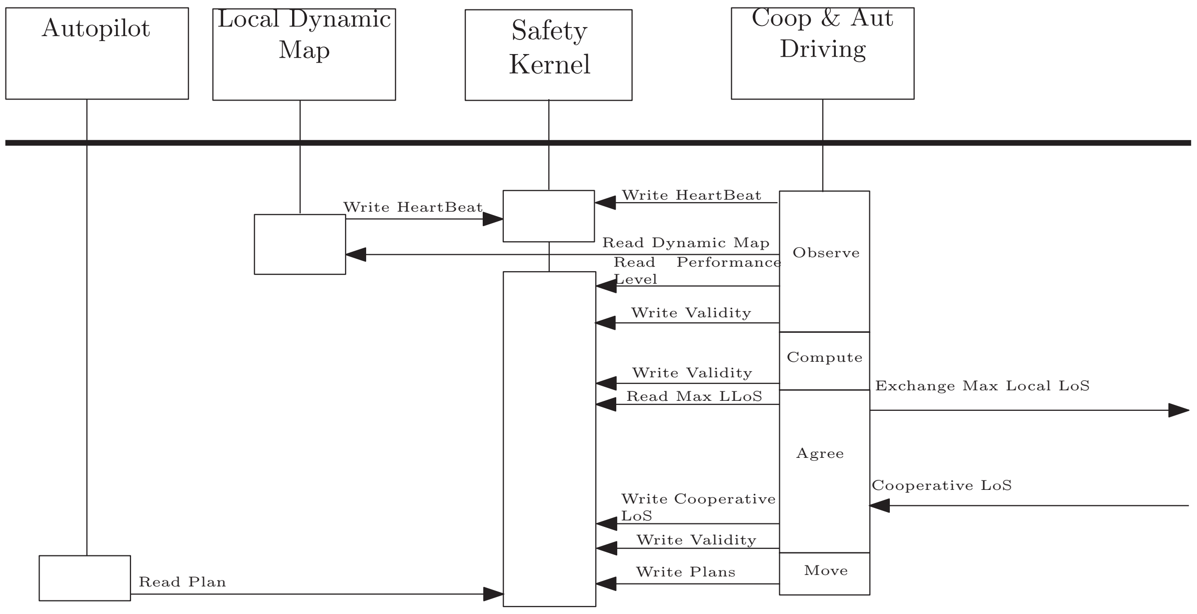


Figure 5: Interaction of the cooperative and autonomous driving module with the other modules. (Sequence Diagram.)

We assume a fully synchronous system that ensures that all the vehicles start the rounds at the same time. Let $OBSERVE\_TIME$, $COMPUTE\_TIME$, $AGREE\_TIME$ and $MOVE\_TIME$ be the time that each round requires to complete the task. Therefore, each round takes the sum of these four values, i.e., $ROUND = OBSERVE\_TIME + COMPUTE\_TIME + AGREE\_TIME + MOVE\_TIME$. We are interested in short rounds to bound the uncertainty. Indeed, as we consider at most three vehicles, the *computation phase* can be completed in few milliseconds. Similarly, the *observation phase* and the *movement phase* only require few milliseconds since they are the results of parallel processes. Unfortunately, there is no algorithm that guarantees agreement in constant time. Thus, for the *agreement* phase we consider empirical values, such as the average case scenario. For simplicity of presentation, we assume that these values are known constants.

In the *observation* phase, the vehicles timely collect information from the *local dynamic map* (LDM). The information includes the Id, localization, speed, etc, of every member (when available). At the beginning of the *observation* phase, the vehicles write in the *timely failure detector* a heartbeat.

In the *computation* phase, the vehicles deterministically determine the plans for $round_{now} + 1$ with the information collected by the current round. The plans include the target (or trajectory) for all service levels that the vehicle supports. We require a special trajectory, called *transition trajectory*, that guarantees a safe transition from a higher service level to the lowest service level in at most $transitionTime$ rounds. This trajectory is used in case that either the system fails due to communication, lack of agreements and/or low data validity, in the next round and, the safety kernel changes the performance level to the lowest one. The transition trajectory is intended for the $round_{now} + 2$. Thus, to obtain the transition trajectory for the worst case scenario, we assume that the vehicles will perform the highest service level in $round_{now} + 1$. For simplicity, we assume that the vehicles are able to follow the road and plans with high certainty.

In the *agreement* phase, the vehicles exchange the local maximum level of service so far in order to reach agreement on the cooperative LoS. We define the cooperative level of service as the minimum among the maximum local service levels. The driver manager indicates a low validity whenever on round $round_{now}$ vehicles have not reached an agreement (within $AGREE\_TIME$) on the cooperative level of service for $round_{now} + 1$. We note this low validity can cause the safety kernel to lower the performance level for round $round_{now} + 1$.

In the *movement* phase, all the trajectories are written in the *multiplexer pilot.* The safety kernel decides the trajectory that corresponds to the performance level according to the data validity. We require to the *autonomous driving* that when the performance level is lowered to the lowest, it sends to the *pilot multiplexer* the transition trajectory for $transitionTime$ rounds.

Figure 5 depicts the interaction over a round of the cooperative and autonomous driving with the safety kernel and LDM.

7 Test Cases

We demonstrate the ability of the safety kernel to facilitate the implementation of vehicular coordination algorithms. The algorithms consider a set of parameters given by the test-bed and the vehicle constraints; see Table 1. For simplicity of presentation, we consider these parameters without making a formal declaration of them.

Parameter	Description
$maxAcceleration$	Maximum vehicles' acceleration capability (bound)
$maxDeceleration$	Maximum vehicles' deceleration capability (bound)
$maxSpeed$	The road speed limit
$cruisingSpeed$	The cruising speed that all vehicles are aiming at. We assume that the vehicle's velocity may temporarily exceed their *cruisingSpeed* but never $maxSpeed$
$length$	A bound on the vehicle length considering the maximum error
$autonomousLoSHeadway$	Minimum headway required in the autonomous level of service
$fullyCooperativeLoSHeadway$	Minimum headway required in the highest level of service
$transitionTime$	The transition time (number of rounds) between the highest service level and the lowest service level

Table 1: Test-bed Parameters

Adaptive Cruise Control and Vehicle Platooning. This application adjusts the headway between vehicles according to the data validity and the performance level. Namely, the safety kernel adjusts the inter-vehicle distance according to the performance level.

The adaptive cruise control has been studied in [11, 16, 17]. The main pieces of information in this vehicular application are the relative distance and speed to the vehicle ahead. We assume that the information that is coming from onboard sensors, such as the inter-vehicle distance, has sufficiently high quality. Vehicle platooning [6, 8, 14, 15] uses vehicle to vehicle communication for agreeing on the joint (cooperative) cruising speed.

In the scope of this test case, we consider two service levels, the highest that corresponds to the cooperative application of vehicular platooning that is based in [15] and the lowest that corresponds to the autonomous application of adaptive cruise control based on [11]. We refer to [11] and [15] for the proofs of these algorithms. Our extendable approach does not consider external factors, such as friction, mass, and road condition. In this context, the vehicular platooning depends only on the speed and relative position of each other vehicle. Meanwhile, the adaptive cruise control depends only on the speed and relative position of the vehicle in front. In this test, we consider a single lane loop with three vehicles, we will assume that the vehicle with minimum id is the leading vehicle of the adaptive cruise control and platooning. On a broader scope than KARYON, cooperative vehicle systems have the task of determining the participating vehicle set that is forming a platoon since different views may result in conflicting trajectories.

Data Validity	Service Level (Headway)
High Validity	Platooning ($fullyCooperativeLoSHeadway$)
Low Validity	Adaptive Cruise Control ($autonomousLoSHeadway$)

Table 2: Safety Kernel Rules for the ACC/Platooning

In the pseudo-code depicted in Algorithm 2, we use the structure using two parts: one for autonomous driving (adaptive cruise control) and one for cooperative driving (vehicular platooning). We compute the trajectory plan for

Algorithm 2: Adaptive Cruise Control and Vehicle Platooning. (Each vehicle c executes it.)

```
Input: $member = \{u, v, w\}$: the set of the three vehicles in the domain.
Input: $vehicle = [\langle id, frontDistance, speed\rangle]_{id \in members}$ a vector of vehicles in which each record refers to the
       distance to the vehicle in front and speed at the beginning of $round_{now}$.
Output: $preliminaryPathPlan[autonomousLoS, fullyCooperativeLoS, transition] : speed$ a vector with the
        preliminary path plans of $c$.
Data: $speed[fullyCooperativeLoS, transition] : [\langle id, speed\rangle]$ speed of each vehicle.
1  function $non\_coop\_speed()$: determines the speed of $c$ given $autonomousLoSHeadway$, the distance to the
   vehicle in front and its speed, e.g., using the algorithm given in [11];
2  function $coop\_speed(v, frontVehicle, leadingVehicle)$: determines the speed of $v$ given
   $fullyCooperativeLoSHeadway$, the vehicle in front and leading vehicle in the platoon, e.g., using the algorithm
   given in [15];
3  function $tran\_speed(v, vehicleInFront, positionInPlatoon, speed)$: determines the speed of $v$ for the transition
   from $fullyCooperativeLoSHeadway$ to the $autonomousLoSHeadway$ in $transitionTime$ rounds given the
   vehicle in front and its position in the platoon. To obtain the transition trajectory for $round_{now} + 2$ of the worst
   case scenario we assume that all the vehicles will perform their trajectory $speed[fullyCooperativeLoS]$ for one
   round;
4  let $Platoon$ be the sorted set of vehicles on the lane where the vehicle with smallest id is the leading vehicle;
5  foreach $vehicle \in Platoon$ do
6      if vehicle is the first in platoon then
7          let $speed[fullyCooperativeLoS][vehicle.id]$ be the $cruisingSpeed$ of $vehicle$;
           /* Transition trajectories maintain constant speed of the leading vehicles      */
8          $speed[transition][vehicle.id] = speed[fullyCooperativeLoS][vehicle.id]$;
9      else
10         let $leading$ and $next$ be the leading and front of $vehicle$ in $Platoon$;
11         $speed[fullyCooperativeLoS] = [\langle vehicle.id, coop\_speed(vehicle, next, leading)\rangle]$;
           /* The transition trajectory guarantees a safe transition                        */
12         let $position$ be the position of $vehicle$ in $platoon$;
13         $speed[transition] = [\langle vehicle.id, tran\_speed(vehicle, next, leading, position, speed)]$;
14 $preliminaryPathPlan[fullyCooperativeLoS] = speed[fullyCooperativeLoS][c.id]$;
15 $preliminaryPathPlan[transition] = speed[transition][c.id]$;
16 $preliminaryPathPlan[autonomousLoS] = non\_coop\_speed()$;
17 return $preliminaryPathPlan$
```

both service levels and let the safety kernel determine the system performance level before selecting the appropriate plan. Algorithm 2 considers the inter-vehicle distance and speed of all the vehicles. The system computes this by using the vehicle positions that the local dynamic map (LDM) provides. The autonomous trajectory (adaptive cruise control) in Algorithm 2 is computed in Line 16. Meanwhile, the cooperative trajectory (vehicular platooning) and transition trajectory are computed in Lines 5-15. For the vehicular platooning, the speeds are determined in the order in which they appear in the cluster starting with the leading vehicle.

One can show that Algorithm 2 is collision-free provided that all vehicles can follow the trajectories with a high level of precision. Indeed, the system is stable and safe if it is performing adaptive cruise control and continue in adaptive cruise control by [11]. Similarly, the system is safe and stable if it is performing platoon and continue performing platoon by [15]. Furthermore, the transition between lower service level to higher service level is safe as the vehicles have complete knowledge of the members. The risky part is the transition between the highest service level to the lowest service level. However, as they compute the transition trajectory at $round_{now} - 1$ and they follow the transition trajectory for $transitionTime$ rounds with a high level of precision, the transition is safe.

The demonstration plan includes at most three vehicles and a single vehicle cluster or platoon. When considering a trajectory plan that includes several clusters, one must take into account their dependencies among them. However, these considerations are orthogonal to this demonstration of the safety kernel. As the communication can be lost, one has to consider the worst case scenario for the transition trajectories. Thus, the transition time becomes the fundamental issue for the correct design of vehicular systems. The problem is more evident when the number of platoons is greater than one. Indeed, one can maintain safety in the transition time by keeping a minimum distance between the leading vehicles of two platoons p_1, p_2 with p_1 in front of p_2 of at least $(|p_1|length)$ times the distance that corresponds to the lowest level of service where length is the largest vehicle length. This distance allows the platoons to return to cruise control independently as no communication may be available. Figure 6 depicts the transition between highest service level and lowest service level. However, external events may create cascade effects as depicted in Figure 7.

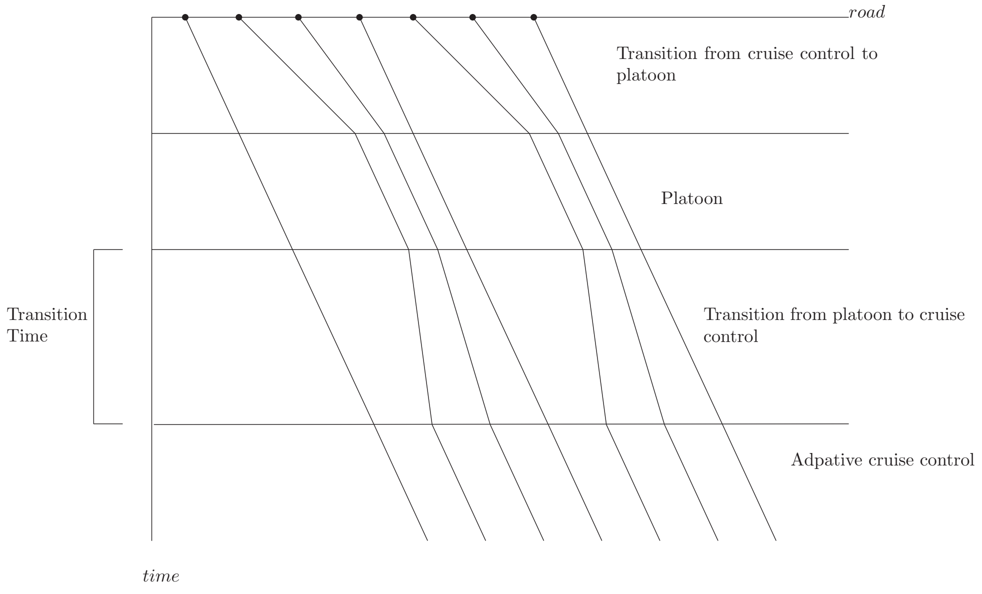

road
Transition from cruise control to platoon
Platoon
Transition Time
Transition from platoon to cruise control
Adpative cruise control
time

Figure 6: Transition periods between highest service level and the lowest service level.

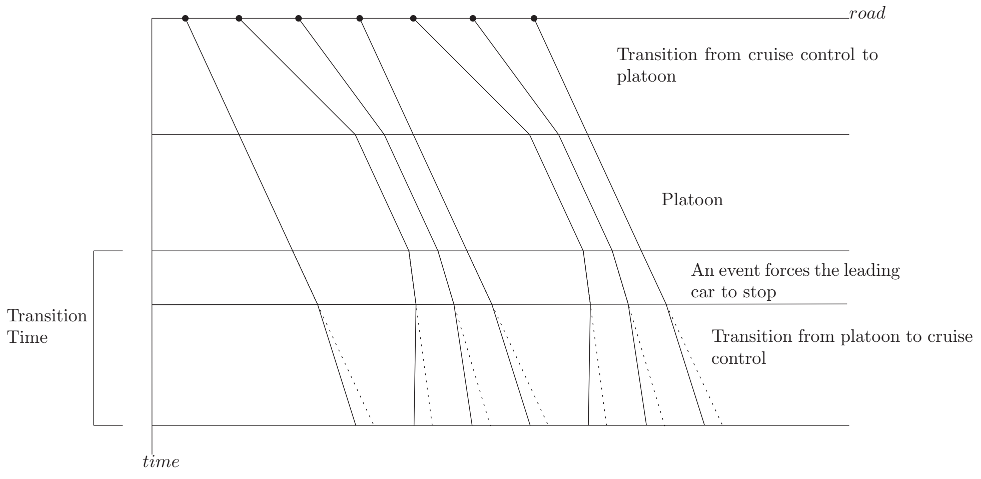

road
Transition from cruise control to platoon
Platoon
An event forces the leading car to stop
Transition Time
Transition from platoon to cruise control
time

Figure 7: Cascade effect in the transition time. (The dotted points depicts the computed trajectories.)

Intersection Crossing This application schedules a safe intersection crossing by adjusting the vehicle speed according to the data validity and the performance level that is determined by the safety kernel. The cooperative intersection crossing algorithm is based on [13]; see that paper for the formal proof of the algorithm. We use the first-come first served approach for deciding who crosses the intersection first. We break symmetry using the right hand rule, i.e., we assume that the roads have a given priority. Such symmetry breaking techniques are essential especially at low service levels for which there are no communication-based coordination guarantees. We note that our approach is extendable to systems that further consider the details and optimality of trajectory planning [1, 5, 9, 13].

This test case considers an intersection of the roads that have a single conflicting directions. We define the *critical zone* as the area of high risk of collision. This area corresponds to the intersection, as well as, the road sections next to the intersection; see Figure 8. We assume that the road sections of the critical zone are at least the distance that every vehicle requires to stop completely without entering the intersection. We define the *negotiating zone* as the road sectors next to the critical zone that are in the communication range. We refer to these two zones as the vital zone. The safety kernel rules are presented in Table 3, and the pseudo-code is depicted in Algorithm 3.

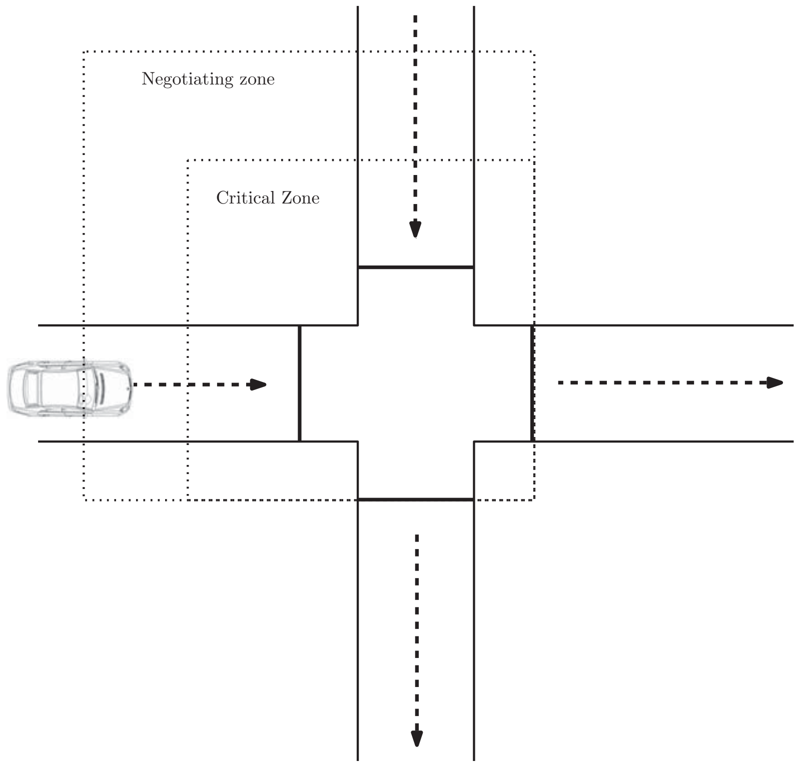


Figure 8: Transition periods between highest service level and the lowest service level.

Data Validity	Service Level (Headway)
High Validity	Cooperative Intersection Crossing
Low Validity	Autonomous Intersection Crossing

Table 3: Safety Kernel Rules for the Intersection crossing.

In the pseudo-code depicted in Algorithm 3, we use the structure using two parts: one for the autonomous driving (autonomous intersection crossing) and one for the cooperative driving (cooperative intersection crossing). We compute the trajectory plan for both service levels and let the safety kernel determine the system performance level before selecting the appropriate plan. The algorithm considers the distance to the critical zone, speed and lane of the vehicles. The system computes this by using the vehicle positions and road map that the local dynamic map (LDM) provides. We note that the priority in the input is determined by the driving management and refers to the priority obtained in the previous round.

```
Algorithm 3: Intersection Crossing. (Each vehicle $c$ executes it.)
    Input: $member = \{u, v\}$: the set of the two vehicles in the domain.
    Input: $vehicle = [\langle id, distanceToCriticalZone, speed, lane, priority\rangle]_{id \in members}$ a vector of vehicles in which
        each record refers to the distance to the critical zone, speed, lane and priority.
    Output: $preliminaryPathPlan[autonomousLoS, fullyCooperativeLoS, transition] = [\langle acc, priority\rangle]_{id \in members}$
        a vector of the vehicle acceleration and priorities so that the vehicle with highest priority safely cross
        before the vehicle with lowest priority.
    Data: $speed[fullyCooperativeLoS, transition]$ : The speed of $c$;
1   function $conflictFreePath(priority)$ computes the path for $c$ as follows: If $priority = HighPriority$, $c$ will
    cross before, otherwise $c$ will cross after. The path guarantees that the vehicle with lower priority enters the
    critical zone after the vehicle with higher priority has left. The output of this function is the speed for the next
    round;
2   function $stopAtCriticalZone(r)$ computes the path for $c$ so that it stops just before the critical zone in at most $r$
    rounds. The output of this function is the speed for the next round;
3   let $b = \{u, v\} \setminus \{c\}$;
4   let $priority = lowestPriority$;
5   if $c$ is not in the vital zone then  $priority = \perp$;
6   else if $c$ is in the vital zone and $b$ is not in the vital zone then  $priority = highestPriority$;
7   else if $c$ and $b$ are in the vital zone then
8       if $c.priority = b.priority$ then
9           if $c$ arrives at the intersection before $b$ at the current speeds then  $priority = highestPriority$;
10          else $c$ arrives at the intersection at the same time as $b$ at the current speeds and $c$ is on the right lane
            $priority = highestPriority$;
11      else if $c.priority \neq \perp$ then  $priority = c.priority$;
12  $speed[fullyCooperativeLoS] = conflictFreePath(priority)$;
13  $preliminaryPathPlan[fullyCooperativeLoS] = [\langle speed[fullyCooperativeLoS], priority\rangle]$;
14  $speed[transition] = speed[fullyCooperativeLoS]$;
15  if $c$ is in the negotiating zone then  $speed[transition] = stopAtCriticalZone(transitionTime)$;
16  $preliminaryPathPlan[transition] = [\langle speed[transition], priority\rangle]$;
17  if $c$ is in the vital zone and  $b$ is not in the vital zone then
    $preliminaryPathPlan[autonomousLoS] = [\langle conflictFreePath(HighPriority), HighPriority\rangle]$;
18  else  if $c$ is in the vital zone on the right lane and  $b$ is stop then
    $preliminaryPathPlan[autonomousLoS] = [\langle conflictFreePath(HighPriority), HighPriority\rangle]$;
19  else  $preliminaryPathPlan[autonomousLoS] = [\langle stopsAtCriticalZone(\infty), priority\rangle]$;
20  return $preliminaryPathPlan$
```

The main idea of Algorithm 3 is to maintain the intersection with at most one vehicle at every time. This is guaranteed by calculating consistent priorities. Lines 5 to 11 calculate the priority. We give the highest priority to the vehicle in the right lane. One can show that the algorithm is safe by proving that exactly one vehicle has the highest priority and priorities are consistent between rounds and vehicles. Line 13 computes the speed of c according to its priority. If c is in the critical zone and there is the need to lowered the level of service, c will cross the intersection at the cruise speed; line 14. However, if this happens when c is in the negotiating zone, it will stop before crossing the intersection in at most $transitionTime$ rounds; line 15. The autonomous trajectory is determined in lines 17-19. This trajectory lets the vehicle cross if there is not other vehicle in the vital zone, or when the vehicle is on the right lane inside the vital zone and the other vehicle is stopped completely in the vital zone.

Our demonstration plan includes two vehicles. When considering a trajectory plan that includes several vehicles, one must take into account their priorities. However, these considerations are orthogonal to this demonstration of the safety kernel.

Coordinated Lane-Change This application schedules a safe lane-change maneuver by adjusting the inter-vehicle distances in the subject lane according to the data validity and the performance level that the safety kernel determines. The lane change maneuver has been studied in [10, 12] in the non-cooperative context and [2, 3] in the cooperative context.

We assume that the vehicles are able to follow the paths to change lanes and when, it needs to abort the maneuver, return to the original lane [12]. Algorithm 4 considers the accessibility of (primitive) applications for maintaining appropriate inter-vehicle distance, e.g., adaptive cruise control and vehicular platooning. The algorithm considers a vehicle that is changing lane to a *subject (target) lane*, as well as the vehicle projection on the subject lane; see the safety kernel rules given in Table 4. We define the *projection* as the intersection between the bisector ray of the vehicle position and the subject lane.

Data Validity	Service Level (Headway)
High Validity	$2 \cdot fullyCooperativeLoSHeadway \cdot speed + length$
Low Validity	$2 \cdot autonomousLoSHeadway \cdot speed + length$

Table 4: Safety Kernel Rules for Lane-Change

In the pseudo-code depicted in Algorithm 4, we use the structure using two parts: one for the autonomous maneuver and one for the cooperative maneuver. We compute the trajectory plan for both service levels and let the safety kernel determine the system performance level before selecting the appropriate plan. The autonomous lane change algorithm is based on [12] and the ACC-platooning system; see [12] for the formal proof. The cooperative lane change algorithm is based on [3]; see [3] for the formal proof; The algorithms consider the inter-vehicle distance, lateral distance, speed and lane for each vehicle. The system computes this by using the vehicle positions and road map that the local dynamic map (LDM) provides. We note that the state in the inputs is determined by the driving management and refers to the state in the maneuver in the previous round.

We implement algorithm 4 as a distributed state machine. The sate machine can be implemented using a virtual stationary automaton [7]. The main idea of the algorithm is the following: Consider a vehicle c that aims to change lanes. Assume that the two closest vehicles on the subject lane are c_1 and c_2 with c_2 following c_1. Without loss of generality, we may assume that the projection of c is in-between c_1 and c_2. At first, they start preparing to open a space by performing an adaptive cruise control or platooning with c_1, c_2 and the projection of c according to the information quality; line 22. They will start performing the maneuver if the space $|c_1c_2|$ is at least the distance that corresponds to $2fullyCooperativeLosHeadway + length$ and the performance level of c_1 and c_2 is the highest; line 18. However, if the performance level of c_1 and c_2 is the lowest and $|c_1c_2|$ is at least the distance that corresponds to the $2autonomousLosHeadway + length$, they will start performing the maneuver; line 19. While they are performing the maneuver, they check for hazardous situation. If there is one hazardous situation that can avoid completing the maneuver, they will abort it; line 14. Otherwise, they will finish it.

Our demonstration plans include three vehicles. This allows us to determine the vehicles involved in the maneuver. When considering an implementation that is more extensive than this pilot demonstration of the safety kernel one has to consider more conflict scenarios. For example, a conflicting situation may arise when two vehicles are aiming to change lanes simultaneously in the same spot. As a future direction, we propose to monitor the participating vehicle sets in the maneuver and maintain the set consistencies, e.g., through state replication and consensus protocols.

As in the adaptive cruise control, the transitions from the highest service level to the lowest service level while they are performing is the most risky maneuver. This is specially true when the communication fails. For example, assume that the vehicles are performing the maneuver and there is the need to abort due to a hazardous situation. However, if communication broke just before aborting, the vehicle must detect the situation with its local sensory information.

8 Fault Injection

We consider fault injection on the cooperative evaluator as well as in vehicles communication. However, we do not consider fault injections on the local dynamic map. Details will be provided in a further report.

Algorithm 4: Lane Change. (Each vehicle c executes it.)

```
Input: $member = \{u, v, w\}$: the set of the three vehicles in the domain.
Input: $vehicle =$
    $[\langle id, distanceToFrontVehicle, lateralDistance_1, lateralDistance_2, speed, lane, state\rangle]_{id \in members}$ a vector
    of vehicles in which each record refers to the distance to the vehicle in front, the distance to the lateral
    vehicles, speed, lane and state in the maneuver at the end of $round_{now} - 1$.
Output: $preliminaryPathPlan[autonomousLoS, fullyCooperativeLoS, transition] = [\langle speed, state\rangle]_{id \in members}$
    a vector of $c$'s speed and state of the maneuver.
1  let $maneuver$ be the sorted set of vehicles such that $maneuver[1]$ is in front of $maneuver[2]$ in the same lane and
   $maneuver[3]$ is the vehicle that is aiming to change lanes;
2  let $virtualVehicle$ be the projection of $maneuver[3]$ on $maneuver[1].lane$ using
   $distanceToFrontVehicle, lateralDistance_1$ and $lateralDistance_2$;
3  let $speedPlan$ be the preliminary path plan obtained from executing Algorithm 2 (Adaptive cruise control) with
   input $\{maneuver[1], maneuver[2], virtualVehicle\}$;
4  $preliminaryPathPlan = [\langle speedPlan, state\rangle]$;
5  switch $state$ do
6      case $ABORTING$
7          if $maneuver[3]$ has completed returning to its original lane then
           $preliminaryPathPlan = [\langle speedPlan, PREPARING\rangle]$;
8      case $PERFORMING$
9          if maneuver is completed then $preliminaryPathPlan = [\langle speedPlan, COMPLETED\rangle]$;
10         else
11             if it is safe to complete the maneuver then
12                 $preliminaryPathPlan = [\langle speedPlan, PERFORMING\rangle]$;
13                 $preliminaryPathPlan[transition] = preliminaryPathPlan[fullyCooperativeLoS]$;
14             else
15                 $preliminaryPathPlan = [\langle speedPlan, ABORTING\rangle]$;
16                 $preliminaryPathPlan[transition] = preliminaryPathPlan[fullyCooperativeLoS]$;
17     case $PREPARING$
18         if the inter-vehicle distances between members in $\{maneuver[1], maneuver[2], virtualVehicle\}$ are at
           least $fullyCooperativeLoSHeadway$ and $maneuver[1].LoS = fullyCooperativeLoS$ then
           $preliminaryPathPlan = [\langle speedPlan, PERFORMING\rangle]$;
19         if the inter-vehicle distances between members in $\{maneuver[1], maneuver[2], virtualVehicle\}$ are at
           least $autonomousLoSHeadway$ then $preliminaryPathPlan = [\langle speedPlan, PERFORMING\rangle]$;
20     otherwise
21         if $maneuver[3].lane \neq maneuver[1].lane$ then
22         $preliminaryPathPlan = [\langle speedPlan, PREPARING\rangle]$;
23         else $preliminaryPathPlan = [\langle speedPlan, COMPLETED\rangle]$;
24 return $preliminaryPathPlan$;
```

References

[1] Samer Ammoun and Fawzi Nashashibi. Design and efficiency measurement of cooperative driver assistance system based on wireless communication devices. *Transportation Research Part C: Emerging Technologies*, 18(3):408 – 428, 2010.

[2] Samer Ammoun and Fawzi Nashashibi. Design and efficiency measurement of cooperative driver assistance system based on wireless communication devices. *Transportation research part C: emerging technologies*, 18(3):408–428, 2010.

[3] Samer Ammoun, Fawzi Nashashibi, and Claude Laurgeau. An analysis of the lane changing manoeuvre on roads: the contribution of inter-vehicle cooperation via communication. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 1095–1100. IEEE, 2007.

[4] Hideki Ando, Yoshinobu Oasa, Ichiro Suzuki, and Masafumi Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *Robotics and Automation, IEEE Transactions on*, 15(5):818–828, 1999.

[5] Reza Azimi, Gaurav Bhatia, R Rajkumar, and Priyantha Mudalige. Intersection management using vehicular networks. In *Society for Automotive Engineers (SAE) World Congress*, 2012.

[6] Eric Chan, Peter Gilhead, Pavel Jelínek, and Petr Krejei. Sartre cooperative control of fully automated platoon vehicles. In *18th ITS World Congress*, 2011.

[7] Shlomi Dolev, Seth Gilbert, Limor Lahiani, Nancy A. Lynch, and Tina Nolte. Timed virtual stationary automata for mobile networks. In James H. Anderson, Giuseppe Prencipe, and Roger Wattenhofer, editors, *OPODIS*, volume 3974 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2005.

[8] J Alexander Fax and Richard M Murray. Information flow and cooperative control of vehicle formations. *Automatic Control, IEEE Transactions on*, 49(9):1465–1476, 2004.

[9] Javier Ibanez-Guzman, Stephanie Lefevre, Abdelkader Mokkadem, and Sylvain Rodhaim. Vehicle to vehicle communications applied to road intersection safety, field results. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 192–197. IEEE, 2010.

[10] Hossein Jula, Elias B Kosmatopoulos, and Petros A Ioannou. Collision avoidance analysis for lane changing and merging. *Vehicular Technology, IEEE Transactions on*, 49(6):2295–2308, 2000.

[11] Maziar E Khatir and Edward J Davison. Decentralized control of a large platoon of vehicles using non-identical controllers. In *American Control Conference, 2004. Proceedings of the 2004*, volume 3, pages 2769–2776. IEEE, 2004.

[12] Iakovos Papadimitriou and Masayoshi Tomizuka. Fast lane changing computations using polynomials. In *American Control Conference, 2003. Proceedings of the 2003*, volume 1, pages 48–53. IEEE, 2003.

[13] Gabriel Rodrigues de Campos, Paolo Falcone, and Jonas Sjöberg. Autonomous cooperative driving: a velocity-based negotiation approach for intersections crossing. In *16th International IEEE Conference on Intelligent Transportation Systems*, 2013.

[14] Steven E Shladover. Longitudinal control of automotive vehicles in close-formation platoons. *Advanced automotive technologies, 1989*, 1989.

[15] Steven E Shladover, Charles A Desoer, J Karl Hedrick, Masayoshi Tomizuka, Jean Walrand, W-B Zhang, Donn H McMahon, Huei Peng, Shahab Sheikholeslam, and Nick McKeown. Automated vehicle control developments in the path program. *Vehicular Technology, IEEE Transactions on*, 40(1):114–130, 1991.

[16] Srdjan S Stankovic, Milorad J Stanojevic, and Dragoslav D Siljak. Decentralized overlapping control of a platoon of vehicles. *Control Systems Technology, IEEE Transactions on*, 8(5):816–832, 2000.

[17] Youping Zhang, B Kosmatopoulos, Petros A Ioannou, and CC Chien. Using front and back information for tight vehicle following maneuvers. *Vehicular Technology, IEEE Transactions on*, 48(1):319–328, 1999.