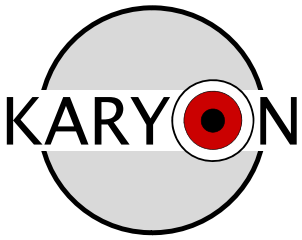


Kernel-based ARchitecture for safetY-critical cONtrol

KARYON

FP7-288195

D3.1 – First Report on Supporting Technologies (Annex)

Work Package	WP3		
Due Date	M12	Submission Date	2012-10-26
Main Author(s)	Jörg Kaiser (OVGU), José Rufino (FFCUL), Elad Michael Schiller (CUT)		
Contributors	Tino Brade (OVGU), Sasanka Potluri (OVGU), Jeferson Souza (FFCUL, Luís Marques (FFCUL)		
Version	1.0	Status	Final
Dissemination Level	Public	Nature	Report
Keywords	Wireless protocols, inaccessibility analysis, adaptive middleware, reliable collaborative sensing, self-stabilizing protocols, assessment of global state.		

Part of the Seventh

Framework Programme

Funded by the EC - DG INFSO

This page is intentionally left blank.

Table of Contents

This page is intentionally left blank.

Annex A Papers and Reports

A.1 Predictability and Resilience in Embedded Networks

A.1.1 An Approach to Enhance the Timeliness of Wireless Communications

"An Approach to Enhance the Timeliness of Wireless Communications". J. L. R. Souza and J. Rufino, Fifth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2011), November 2011, Lisbon, Portugal.

This page is intentionally left blank.

An Approach to Enhance the Timeliness of 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*—Wireless technologies are the present and the future of network communications. However, the support of real-time data transmission in wireless communications — providing support for execution of well-timed networked operations — is still an open issue, not fully addressed by current wireless network standards and technologies. Thus, this paper proposes a solution to enhance the timeliness of wireless communications without a need for fundamental modifications to the standard specifications. The IEEE 802.15.4 wireless network is used as a relevant case study. Our main contributions in this paper are: (a) a proposal to enhance the timeliness of wireless communications; (b) the extension of the data frame transmission service in order to control the effects of temporary partitions caused by disturbances in the medium and medium access control protocols; (c) a strategy to reduce the negative effects caused by the aforementioned disturbances.**

***Index Terms*—medium access control, inaccessibility, wireless communication, real-time systems.**

I. Introduction

The provision of temporal guarantees on wireless communications is still an open issue. Several approaches [1]–[4] to the problem of enhancing the timeliness of wireless communications assume that the network always operates normally, disregarding the occurrence of disturbances in the medium and medium access control (MAC) protocols.

However, wireless networks are extremely sensitive to external disturbances such as those resulting from electromagnetic interference, or application scenarios requiring intense mobility. These disturbances may lead to the occurrence of temporary partitions, also called periods of inaccessibility, where there may be sets of nodes which cannot communicate with each other [5]. Standard MAC protocols, including those used in wireless communications, can recover from these situations. However, this recovery process takes time and in the meanwhile the network is partitioned. The duration of a period of inaccessibility is dependent on each MAC layer, and must be analyzed for each network, such as the one defined in the IEEE 802.15.4 standard [6].

The occurrence of periods of inaccessibility leads to disruptions in the provision of MAC layer services. Furthermore, the analysis of the wireless protocol stack with a bottom-up approach shows that these disturbances may affect the entire stack, implying that service disruption may propagate upwards, and therefore interfere with the execution of higher layer protocols and applications. Thus, this paper proposes a new component layer executing on top of the MAC exposed interface to control the timeliness of wireless communications and reduce the impact of MAC layer service disruptions on the execution of the entire wireless protocol stack. This component layer improves the MAC layer functionality, mediating and isolating its interaction with higher layers, and allowing the configuration of the MAC layer parameters face to application requirements and environment restrictions.

The IEEE 802.15.4 wireless sensor and actuator network is used as a case study to present the main features of our proposal. A strategy is also presented to control the negative effects induced by the occurrence of periods of inaccessibility in network operation. Our approach does not require fundamental modifications of wireless network standards and therefore is in compliance with existing Commercial Off-the-Shelf (COTS) network components.

The paper is organized as follows: Section II presents a brief description of the system model used in our analysis. Section III presents an overview of the IEEE 802.15.4 standard. Section IV presents our proposal, describing its main components, the advantages of its use, and the improvements introduced at the data link layer service interface. Section V describes our results, extending the characterization of the data frame transmission service, and the strategy to control the periods of inaccessibility on wireless communications, using the IEEE 802.15.4 as a case study. Finally, Section VI draws some conclusions and future directions of this work.

II. System Model

Our system model is formed by a set of communicating entities (processes/nodes) described by $P = \{p_1, p_2, p_3, ..., p_N\}$. Each entity, p_n, represents a process/node within a wireless network segment with n varying from 1 to N.

In an arbitrary geographic region we assume that all wireless nodes either communicate with each other at only one hop of distance or are out of reach. This means, all communicating wireless nodes are within the region of influence of one another and therefore each node can sense all transmissions of any other node. Hence, we assume the given wireless network

This work was partially motivated by our work within the scope of the ESA (European Space Agency) Innovation Triangle Initiative program, through ESTEC Project AIR-II (ARINC 653 in Space — Industrial Initiative), URL: http://air.di.fc.ul.pt. This work was partially supported by EC, through project IST-STREP-288195 (KARYON) and by FCT through the Multiannual Funding and CMU-Portugal Programs and the Individual Doctoral Grant SFRH/BD/45270/2008.

segment being composed of N nodes interconnected by a channel. Each communicating node $p_n \in P$ connects to the channel by a transmitter and a receiver. Network components either behave correctly or crash upon exceeding a given number of consecutive omissions, the omission degree bound, k. An omission is an error that destroys a data or control frame. Wireless communication channels are especially susceptible to omission errors, which may be due to a number of causes: electromagnetic interference in the medium; disturbances in a node transmitter/receiver circuitry; collisions derived from transmissions performed by different nodes on the same time; glitches in the network protocol operation; or even effects resulting from node mobility.

Despite its importance, the presence of channel malicious attacks [7], [8] is not considered in this paper, in order to simplify the system model and our analysis. Malicious attacks will be thoroughly addressed in a future work.

The omission of control frames (e.g., a token or a beacon) may generate temporary network partitions, logical rather than physical, called periods of inaccessibility [5]. A period of inaccessibility is a time interval where the network does not provide service although it cannot be considered failed. The characterization of IEEE 802.15.4 inaccessibility with respect to non-malicious disturbances is addressed in [6]. In addition, we assume that the wireless network is, at most, inaccessible i times, during a time interval relevant for protocol execution.

III. IEEE 802.15.4 Overview

The IEEE 802.15.4 standard specifies that each network must contain a coordinator, which defines the characteristics of the network such as addressing, supported radio channels, and operation mode. Normally, the coordinator is the node with the highest power and energy capabilities to support the execution of management operations required to maintain the network active throughout two operation modes: NonBeacon-enabled and Beacon-enabled. The case study addressed in this work (Section V) assumes a Beacon-enabled operation.

In the Beacon-enabled mode, the access to the wireless medium is controlled by information carried in a special frame sent by the coordinator. This special frame is called beacon and bounds a special structure called superframe, illustrated in Fig. 1. The information inside the beacon helps the nodes to know the entire duration of the superframe, allowing the synchronization and the control of the medium access.

The superframe organization of Fig. 1 identifies two main parts: the active and inactive periods. The active period is mandatory and it is, in turn, constituted by the Contention Access Period (CAP) and the Contention Free Period (CFP). CAP is also mandatory and allows all nodes to compete for the utilization of the shared physical medium. CFP is optional, being designed for bandwidth reservation, and therefore a node may previously allocate a slot, called Guarantee Time Slot (GTS), for exclusive medium access. The slotted version of Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) protocol [9] is used in node competition for medium access during the CAP portion of the superframe.

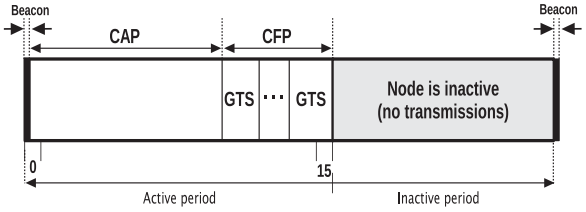


Fig. 1: Superframe structure

Since GTS slots are reserved to a single node, no contention occurs and, within its allocated slot, the node can freely access the medium.

Completing the superframe structure the inactive period is optional and designed to optimize energy consumption. Thus, during this period all nodes in the network may turn off their transceivers to accomplish this goal [10].

IV. An approach to enhance the timeliness of wireless communications

Our approach to enhance the timeliness of wireless communications consists of an extensible component layer build around a standard MAC layer, dubbed *Mediator Layer*. This extensible component layer intermediates the communication and provides error isolation between the MAC and higher layers, minimizing the negative effects caused by disturbances in the medium and medium access protocols. The *Mediator Layer* is a standard-compliant solution which extends the MAC layer services with additional features and guarantees, enhancing the timeliness of wireless communications.

As drawn in Fig. 2 the *Real-Time Protocol Suite*, the *Timeliness and Partition Control*, and the *Configuration and Management Control* are fundamental components handling and managing the actions required to secure reliability and timeliness in data communications, thus enhancing the properties of the native MAC service.

The *Real-Time Protocol Suite* is responsible for handling data transmissions. This component enhances the frame transmission service provided by the MAC layer, establishing a foundation to offer a set of different service guarantees, with respect to reliability and timeliness, such as message transmission time bounds. Different protocols, serving requests with different types of requisites, can be incorporated in this component, augmenting the applicability of standard MAC layers on different areas with different requirements, namely on those with strict real-time demands, such real-time control and monitoring.

The *Timeliness and Partition Control* component deals with the temporal aspects related to the data transmission service, controlling and monitoring the timing of the actions within the *Mediator Layer*, and helping to provide resilience against all the occurrences of temporary network partitions. This component monitors the MAC layer to detect the occurrence and to be aware of any partitioning incidents, providing services to the *Real-Time Protocol Suite*. For example, a

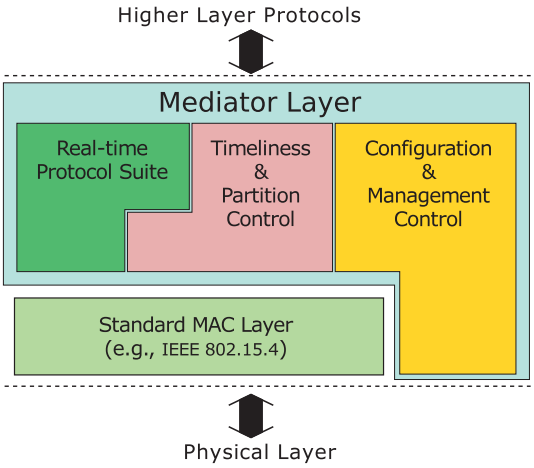


Fig. 2: An approach to enhance the timeliness of wireless communications

timer service controls the temporal execution of protocols, and integrated with the partition control functionality, allows the use of optimal timeout values even in the presence of periods of inaccessibility. Timeout values are automatically extended in this case, thus avoiding a premature and equivocal error propagation to other components and to higher layers.

The *Configuration and Management Control* component manages and controls the configuration of all parameters of the standard MAC layer and the internal parameters of the *Mediator Layer*, respecting realistic application requirements, resource limitations, and environment restrictions. This component makes the *Mediator Layer* (self-)adaptive, and (self-)managed, allowing the possibility to perform some changes in its internal state, and on its configuration profile, thus improving the timeliness of wireless communications.

A. Improving the control of data transmission services

The *Mediator Layer* implements the data layer programming interface. This implementation is represented by MI:

$$MI = \{request,\ confirm,\ indication\}\ . \qquad (1)$$

where the MI set defines the primitives in the *Mediator Layer* service interface. As usual in this kind of interfaces, the primitives are in compliance with the service specification interface described in the IEEE 802.2 standard [11]. Thus, a data transmission service provides three different primitives utilized to request and confirm a data transmission, and to indicate the reception of data.

The services provided by the *Mediator Layer* interface are build on top of MAC level primitives, which description is presented in Table I.

Without the *Mediator Layer*, higher layers shall implement mechanisms to control a frame transmission, ensuring that the frame arrives at its destination. In other words, higher layer protocols shall be: (a) aware of the occurrence of disturbances in the medium and MAC protocols, including periods of inaccessibility; (b) capable to configure parameters of the

Primitives	Description
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 I: Standard MAC layer primitives for data transmission

MAC layer to adapt to different conditions. However, the incorporation of these characteristics increases the complexity of higher layer protocols, forcing each of these protocols to have the capability to cope with low level problems outside the scope of their domains. The introduction of the *Mediator Layer* avoids these design complexities.

The *Mediator Layer* and its components handle all aspects related to a data frame transmission service and its configuration, implying the reduction of the complexity of higher layer protocols. Additionally, with the capability to extend the internal components, our approach also enables the introduction of different types of control mechanisms, transmission protocols, (self-)management and (self-)adaptive strategies, providing an extremely useful service layer. The extension of the MAC data frame transmission service and the control of partition incidents (addressed in Section V-C) are examples of mechanisms implemented in the *Mediator Layer* that improve the services provided to higher layers.

Thus, the *Mediator Layer* is an innovative solution to enhance dependability and timeliness of wireless communications, as low as possible at the protocol stack. Its benefits are flexibly offered at the service interface, being transparently propagated throughout the entire stack, up to highest layer communication protocols and to the applications.

V. Preliminary results: A Case Study on the IEEE 802.15.4 Standard

A. General characterization of the MAC frame transmission service

Based on a user perspective of a MAC frame transmission service we represent in general the time interval required to access the wireless medium as $\mathcal{T}_{W-access}$. The effective time consumed by the node to access the medium is directly related to the medium access protocol in use.

After medium access protocol grants permission to access the medium, a frame is transmitted in the time interval represented by $\mathcal{T}_{MAC-type}$. Hence, equations 2 and 3 represent the best (^{bc}) and worst (^{wc}) cases of MAC frame transmission times.

$$\mathcal{T}^{bc}_{\tau-MAC}(type) = \mathcal{T}^{bc}_{W-access} + \mathcal{T}^{bc}_{MAC-type} \qquad (2)$$

$$\mathcal{T}^{wc}_{\tau-MAC}(type) = \mathcal{T}^{wc}_{W-access} + \mathcal{T}^{wc}_{MAC-type} \qquad (3)$$

These equations contribute to specify a general timeliness representation of a MAC level, presenting simple and easy-to-use formulas to calculate the time bounds of a MAC frame transmission service.

B. The IEEE 802.15.4 Characterization

As we use the IEEE 802.15.4 as a case study to present our results, we calculate the specific bounds of the IEEE 802.15.4 MAC frame transmission service considering a beacon enabled network. All data frame transmissions, with the exception of those performed in the GTS portion of the superframe, need to use of the slotted version of the CSMA/CA protocol [12], [13], analyzed as part of the MAC frame transmission service.

The CSMA/CA is a non-deterministic protocol, and the effective wait value is characterized by a random function, which execution may spam throughout several iterations. In each iteration, the wait time a node uses up is defined by a backoff exponent, as represented by the following equation:

$$\mathcal{T}_{access}(m) \;=\; \mathcal{T}_{backoff} \cdot (2^{BE(m)} - 1) \tag{4}$$

where, $\mathcal{T}_{backoff}$ is the base value defining the minimum duration of a backoff period. Observing that the variability of the backoff exponent is dependent on the iteration number, m, the value of $BE(m)$ for each iteration is given by the following equation:

$$BE(m) = \begin{cases} minBE & \text{if } m = 0 \\ min(minBE + m,\, maxBE) & \text{if } m > 0 \end{cases} \tag{5}$$

The lower and upper bounds of $BE(m)$ are given by $minBE$ and $maxBE$, respectively. The value assigned to $BE(m)$ in the first iteration is equal to $BE(0) = minBE$. For each additional iteration of the CSMA/CA protocol a new value is calculated for $BE(m)$.

The time needed for medium access under normal IEEE 802.15.4 network operation can thus be characterized, in the best and worst cases, by the following equations:

$$\mathcal{T}^{bc}_{W-access} = \mathcal{T}_{access}(0) \tag{6}$$

$$\mathcal{T}^{wc}_{W-access} = \sum_{m=0}^{maxBackoff-1} \mathcal{T}_{access}(m) \tag{7}$$

where, $maxBackoff$ is the maximum number of iterations.

For the evaluation of absolute access time durations, we assume the use of the $2.4\ GHz$ IEEE 802.15.4 frequency operation, with a $62.5\,k\,symbols/s$ symbol rate and with four bits being coded into a single symbol. The default values of Table II are used. Under these conditions, the access to the shared medium may require in the worst case a delay as long as 2563 symbols, i.e., $\mathcal{T}^{wc}_{W-access} \cong 41ms$.

For the maximum frame length of 1016 bits, including headers, the corresponding worst case data frame transmission delay is $\mathcal{T}^{wc}_{\tau-MAC}(data) = 57ms$, assuming no errors during the entire process of a data frame transmission. However, disturbances on the medium and medium access protocols may cause the occurrence of periods of inaccessibility which may induce the occurrence of errors during a data frame transmission.

Parameter	Range	Default	Unit
$maxBackoff$	0-5	4	Integer
$minBE$	0-$maxBE$	3	Integer
$maxBE$	3-8	5	Integer
$\mathcal{T}_{backoff}$	—	20	Symbols

TABLE II: Relevant network parameters defined in the IEEE 802.15.4 standard

C. Dependability and Timeliness of Wireless Communications

Our proposal to control the dependability and timeliness of a frame transmission is divided on two issues: (a) the classical omission error handling present on reliable transmission protocols; (b) and the effective control of periods of inaccessibility.

1) Handling omission errors: Let us consider that the *Real-Time Protocol Suite* component uses a reliable unicast transmission service as an extension of the unreliable transmission service traditionally provided by MAC level standards. This reliable service is a rather classic transmit with acknowledgement (ACK) protocol required to enforce the reliability of a data communication service. To start a reliable transmission some higher level entity shall request a unicast data transmission with delivery guarantee through the *Mediator Layer* programming interface. During protocol execution, the transmitted frame or its associated ACK may be corrupted by disturbances which lead to omission errors. In this case, the destination node does not receive a correct frame, or the sender node does not receive the ACK associated with this frame. As frame corruptions are **transformed into omission errors**, detected when the time interval needed to transmit and receive the corresponding ACK frame ends, the sender node protocol activates a retransmission mechanism and tries to send the frame again, until a maximum number of attempts limited by the bounded omission degree, k, is reached.

However, the occurrence of temporary partitions during a frame transmission may cause a violation of the omission degree limited by k, and therefore the failure of the protocol in delivering the frame to its destination. This happens because the value of k is specified without contemplate the occurrence of periods of inaccessibility, and the standard MAC layer does not provide the additional control provided by our approach.

2) Controlling periods of inaccessibility: Our strategy to handle the occurrence of periods of inaccessibility during a frame transmission **also transforms inaccessibility incidents into omission errors**. A bounded inaccessibility degree, i, is introduced to (self)-adapt and configure the reliable unicast transmission service, and therefore the *Mediator Layer* as well. The combination of i and k (line 8 in Algorithm 1) makes the retransmission mechanism more dynamic, maintaining the timeout used to control reception of the ACK

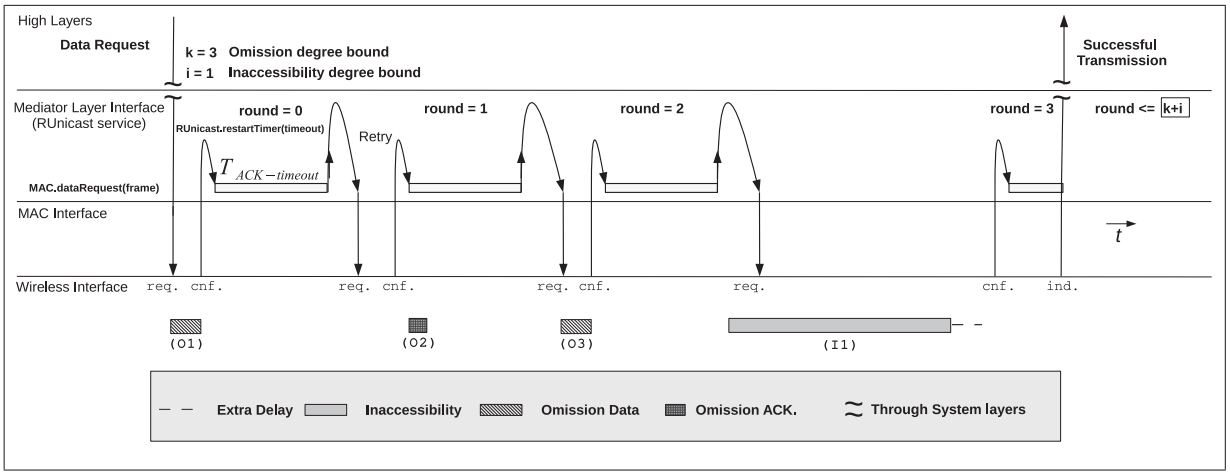


Fig. 3: The Effective Inaccessibility Control Mechanism

($\mathcal{T}_{ACK-timeout}$) with its original and optimized value, and allowing the adaptation of this mechanism to the different durations of each type of inaccessibility scenario (see Table III). The utilization of the same control mechanism for temporary partitions is only possible by the causal relation that exists among the frame transmission request and confirm primitives. Fig. 3 presents a frame transmission mediated by our proposed solution, evidencing that the local confirmation is only provided to the *Mediator Layer* after the actual transmission of the frame on the wireless medium.

Algorithm 1 presents the reliable unicast algorithm with simple, yet fundamental, mechanisms to handle the occurrence of periods of inaccessibility. In Algorithm 1, lines 8 specifies the incorporation of the bounded inaccessibility degree control mechanism in protocol operation, and line 11 the usage of the MAC level confirmation to start the timer which controls the retransmission process (in line 12). The value assigned to the inaccessibility degree bound depends on each network type and its parameters. However, it is reasonable to assume that only one period of inaccessibility would occur during a data transmission, i.e., it is reasonable to assume $i = 1$. The main advantage of such control mechanism is the temporal adaptation of timeout values to the duration of each period of inaccessibility, which may occur at most i times. Although a pure reliability enforcement algorithm only uses k to control the number of retransmissions, the transformation of inaccessibility events into omissions adds i to k and increases the maximum number of retransmissions to $k + i$. That means, the protocol is given a consolidated omission degree bound, K, being $K = k + i$.

In practical terms, this is equivalent to redefining the value assigned to the omission degree bound. This is very important because our control mechanism and the *Mediator Layer*, can be incorporated in any off-the-shelf equipment. In other words, is possible to improve the functionality traditionally offer by the MAC level without change the hardware devices operating in an existent wireless network, being totally transparent to the higher levels.

Algorithm 1 Controlling Inaccessibility (Trapping)

```
1: Initialization phase.
2: k ← omission degree bound;
3: i ← inaccessibility degree bound;
4: round ← 0; accounts for the number of omissions
5: ack_rcv ← 0;
6: Begin.
7: RUcast.data.request(pckt)
8: while round ≤ k+i AND ack_rcv = 0 do
9:    frame ← pckt;
10:   MAC.data.request(frame);
11:   when MAC.data.confirm() do
12:      RUcast.restartTimer(T_ACK−timeout);
13:      when MAC.indication(ACK) received do
14:         ack_rcv ← 1;
15:      end when
16:      when RUcast.timer(timeout_expired) do
17:         round ← round + 1;
18:      end when
19:   end when
20: end while
21: if ack_rcv = 1 then
22:   RUcast.data.confirm(Success);
23: else
24:   RUcast.data.confirm(Failure);
25: end if
26: End.
```

The value of the consolidated omission degree bound shall be dimensioned to consider the specific behavior of each MAC level standard. The related transmission technologies shall also be considered to accomplish the maximum efficiency against environment conditions during the provision of a reliable and timely service. Temporary partitions which may occur and disturb a frame transmission during the operation of the network are handled by the activation of the *Timeliness and Partition Control* component, improving the capabilities of the reliable transmission service provided by the *Mediator Layer*.

Scenario	Designation	Periods of Inaccessibility	
		best case (ms)	worst case (ms)
Single Beacon Frame Loss - No Tracking	$t_{ina \leftarrow sbfl}$	——	3947.71
Multiple Beacon Frame Loss - Tracking	$t_{ina \leftarrow mbfl}$	3947.71	15790.08
Synchronization Loss	$t_{ina \leftarrow nosync}$	15790.08	15790.08
Orphan Node	$t_{ina \leftarrow orphan}$	15794.15	18421.70
Coordinator Realignment	$t_{ina \leftarrow realign}$	2.24	43.30
Coordinator Conflict Detection	$t_{ina \leftarrow C\_Conflict}$	1.14	42.40
Coordinator Conflict Resolution	$t_{ina \leftarrow C\_Resolution}$	63171.54	63822.54
GTS request	$t_{ina \leftarrow GTS}$	0.66	41.47

TABLE III: IEEE 802.15.4 best and worst periods of inaccessibility for the $2.4GHz$ frequency band [6]

D. Extending the general characterization of a MAC frame transmission service

Traditionally, a MAC frame transmission service is not aware of the occurrence of periods of inaccessibility during the network operation. Thus, we shall extend the general characterization of a MAC frame transmission service to incorporate the duration of these periods. This extension is presented in the following equations:

$$\mathcal{T}^{bc}_{\tau-MAC}(type) = \mathcal{T}^{bc}_{W-access} + \mathcal{T}^{bc}_{MAC-type} + \mathcal{T}_{ina} \quad (8)$$

$$\mathcal{T}^{wc}_{\tau-MAC}(type) = \mathcal{T}^{wc}_{W-access} + \mathcal{T}^{wc}_{MAC-type} + \mathcal{T}_{ina} \quad (9)$$

where $\mathcal{T}_{ina}$ represents the duration of a given period of inaccessibility. $\mathcal{T}_{ina}$ is a general term which supports the adaptation of this transmission service to the different durations of each inaccessibility scenario (see Table III). In case of non occurrence of a period of inaccessibility, $\mathcal{T}_{ina} = 0$.

Additionally, to evidence the importance of our proposal and of this control strategy we present in Table III a summary of relevant set of periods of inaccessibility, which if were compared to a data transmission with 1016 bits and transmission time around 57ms, are extremely higher. These values were obtained with an exhaustive analysis of the IEEE 802.15.4 made in [6]. Using the results presented in this paper, the occurrence of a timing fault is detected by the *Mediator Layer*, and its propagation to higher layers is avoided.

VI. Conclusion and Future Work

The potential of wireless networks to support communications on different kinds of environments and applications, with strict timing restrictions, is still an open issue. In this paper we presented our approach to enhance the timeliness of wireless communications, introducing a new component layer with an effective control strategy, avoiding time faults even in the presence of errors in the medium and medium access protocols. Our approach presented a (self-)adaptive and (self-)managed solution, which being in compliance with standards can be used with the existent COTS components.

Future directions involve: reducing the duration of the inaccessibility scenarios based on mechanisms present in the IEEE 802.15.4 standard; improving the support to periodic traffic and applications with hard temporal restrictions; and defining relevant real-time metrics to evaluate the wireless communications with regard to application requirements and environment restrictions.

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 Internatonal Conference on Mobile Ad hoc and Sensor Systems (MASS)*, October 2007, pp. 1 –10.

[2] 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 International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2008, pp. 1016 –1022.

[3] E. Egea-López, J. Vales-Alonso, A. S. Martínez-Sala, J. García-Haro, P. Pavón-Mariño, and M. V. Bueno Delgado, "A wireless sensor networks MAC protocol for real-time applications," *Personal Ubiquitous Computing*, vol. 12, pp. 111–122, January 2008.

[4] X.-Y. Shuai and Z.-C. Zhang, "Research of real-time wireless networks control system MAC protocol," *Journal of Networks*, vol. 5, no. 4, pp. 419–426, April 2010.

[5] P. Veríssimo, J. Rufino, and L. Rodrigues, "Enforcing Real-Time Behaviour on LAN-Based Protocols," in *10th IFAC Workshop on Distributed Computer Control Systems*, Sept. 1991.

[6] 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 International Embedded Systems Symposium(IESS)*, ser. IFIP Advances in Information and Communication Technology, vol. 310, Langenargen, Germany, September 2009.

[7] R. Sokullu, I. Korkmaz, O. Dagdeviren, A. Mitseva, and N. R. Prasad, "An investigation on IEEE 802.15.4 MAC layer attacks," in *10th Int. Symposium on Wireless Personal Multimedia Communications*, 2007.

[8] P. Radmand, A. Talevski, S. Petersen, and S. Carlsen, "Taxonomy of wireless sensor network cyber security attacks in the oil and gas industries," in *24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*. Washington, DC, USA: IEEE Computer Society, 2010, pp. 949–957.

[9] L. Kleinrock and F. Tobagi, "Packet switching in radio channels: Part i–carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Transactions on Communications*, vol. 23, no. 12, pp. 1400 – 1416, December 1975.

[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, 2006, Revision of IEEE Standard 802.15.4-2003.

[11] *ISO IEC 8802-2:1998, Logical Link Control*, IEEE, 1998.

[12] C. Jung, H. Hwang, D. Sung, and G. Hwang, "Enhanced markov chain model and throughput analysis of the slotted CSMA/CA for IEEE 802.15.4 under unsaturated traffic conditions," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 1, January 2009.

[13] J. He, Z. Tang, H.-H. Chen, and Q. Zhang, "An accurate and scalable analytical model for IEEE 802.15.4 slotted CSMA/CA networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 1, pp. 440–448, January 2009.

A.1.2 Characterization of Network Inaccessibility in IEEE 802.15.4 Wireless Networks

"Characterization of Network Inaccessibility in IEEE 802.15.4 Wireless Networks". J. L. R. Souza and J. Rufino, Technical Report DI/FCUL, September 2012, Lisbon, Portugal.

This page is intentionally left blank.

Characterization of network inaccessibility in IEEE 802.15.4 wireless networks

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

Wireless communications are vulnerable to the presence of errors during the network operation. These errors may be originated from different sources such as external electromagnetic interferences, obstacles in communication path, or even glitches in the communication circuitry. Such origins may lead the medium access control (MAC) layer to deviate from its normal operation (without presence of errors), forcing the execution of additional actions to maintain the network operational. The execution of such actions may imply the occurrence of periods of "communication silence", where the network although not being failed, is not performing communications. These periods of "communication silence" are dubbed network inaccessibility, which may induce inaccurate fault detections and deadline misses. Additionally, the occurrence of network inaccessibility may compromise network properties such as predictability, dependability, and timeliness. Thus, this report presents an exhaustive study about network inaccessibility, using the 802.15.4 standard as a case study. All network inaccessibility scenarios are presented, discussing important steps to achieve predictability, dependability, and timeliness in wireless communications.

I. Introduction

Wireless sensor networks (WSNs) are flexible communication networks with a great interest in many areas with temporal restrictions such as industrial, vehicular, military, and aerospace. The main advantages provided by WSNs are the elimination of cables, mobility, and reduced Size, Weight, and Power (SWaP) consumption.

There are some works trying to provide temporal guarantees on WSNs, namely those proposed in [1]–[6]. However, these works do not pay attention to deviation on the medium access control (MAC) layer service that may be caused by electromagnetic interferences, obstacles in the communication path, mobility or even glitches in communication circuitry. Such deviation forces the occurrence of periods where the network does not provide service although it cannot be considered in a fail state, which are dubbed network inaccessibility [7]–[9]. The occurrence of network inaccessibility may induce inaccurate fault detections, and deadlines violations, which may put the overall system at risk.

This document presents an exhaustive study of network inaccessibility in IEEE 802.15.4 wireless networks. This study of the IEEE 802.15.4 standard specification is important to the knowledge of how network inaccessibility affects the operation of WSNs, and to show the impact of the network inaccessibility times in the transmission time bounds. Additionally, the characterization of network inaccessibility is a first step towards the provisioning of timeliness, dependability and predictability guarantees in WSNs, using off-the-shelf IEEE 802.15.4 technology.

This work was partially supported by EC, through project IST-STREP-288195 (KARYON) and by FCT through the Multiannual Funding and CMU-Portugal Programs and the Individual Doctoral Grant SFRH/BD/45270/2008.

Document Organization

The remainder of the document is organized as follows. Section II presents the concept of network inaccessibility. Section III discusses the system model used in our analysis, describing the corresponding assumptions and fundamental properties. Section IV presents an overview of the IEEE 802.15.4 standard while section V characterizes fundamental aspects of the IEEE 802.15.4 service interface. Section VI presents an exhaustive study of network inaccessibility in the IEEE 802.15.4 standard and Section VII discusses the corresponding analytical results, showing that real-time operation over wireless sensor networks is still an open problem. Finally, section VIII concludes the document and presents some future research directions.

II. What is network inaccessibility?

The operation of a MAC layer may be disturbed by errors caused by different sources such as external electromagnetic interferences, obstacles in the communication path, glitches in the node circuitry, or even malicious attacks in the more opened environments. These errors may induce the MAC layer to deviate from its normal operation (without the presence of errors), forcing the execution of actions, such as the transmission of control frames, to maintain the network operational. The period comprised from the detection of such deviation until to the end of execution of all the actions needed to reestablish the normal operation of the MAC layer is dubbed network inaccessibility ([7], [8]). During a period of network inaccessibility a node[1] locally experiences a period of "communication silence", being not able to communicate with any other node. The definition of network inaccessibility present in [8] is summarized here:

> *Certain kinds of components may temporarily refrain from providing service, without that having to be necessarily considered a failure. That state is called* ***network inaccessibility****. It can be made known to the users of network components; limits are specified (duration, rate); violation of those limits implies permanent failure of the component.*

III. System Model

The rigorous definition of a system model is a crucial step to understand and describe the fundamental aspects of a wireless (sensor) network operation. 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. The set of nodes X itself establishes a node relationship entity dubbed wireless network segment, using a given communication channel and a given wireless network segment identifier ($WNID$).

A. Assumptions

In our system model the behavior of a wireless network segment is sustained by assumptions utilized to characterize the network communication capabilities and restrictions of wireless nodes. During a wireless network segment operation cycle we use the following assumptions:

[1]Node is the designation for a wireless network device capable to carry applications on top, and to send and receive frames following the specification of the physical and MAC layers currently in use.

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).

Assumptions 1, 2, and 3 define the physical relationship between nodes within the wireless network segment. Our system model characterizes the relationship between nodes at MAC level, where nodes must be in the communication range of each other to communicate and are able to sense one another (assumption 3). Mobility may drive nodes away of wireless network segment (assumption 2).

In the context of network components, an omission is an error that destroys a data frame. Omissions may be caused by different sources such as node mobility, external electromagnetic interference, fading caused by multipath or transient obstacles on the communication medium, glitches on the MAC layer operation, and malicious attacks. Despite of their importance we are not considering malicious attacks in our analysis, being such topic addressed in future work.

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. We can also see in Fig. 1 the indication of which node is the coordinator. The management activities of the coordinator comprises the assignment of the current communication channel in use by the wireless network segment, the wireless network segment identifier definition, address space delimitation, and so on.

B. Wireless MAC-level properties

A relevant set of properties, presented in Figure 2, are defined for the MAC layer and hold for the wireless network segment. In wired networks, it has been proven that those properties are extremely useful for enforcing dependability and timeliness at higher layers [9], [10]. Thus, we are applying those techniques to the realm of wireless networks [11].

Properties WMAC1 and WMAC2 impose correctness in the value domain. Property WMAC1 (*Broadcast*) formalizes that it is physically impossible for a node in the wireless network segment to send conflicting information to different nodes, in the same broadcast [12]. Property WMAC2

[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.

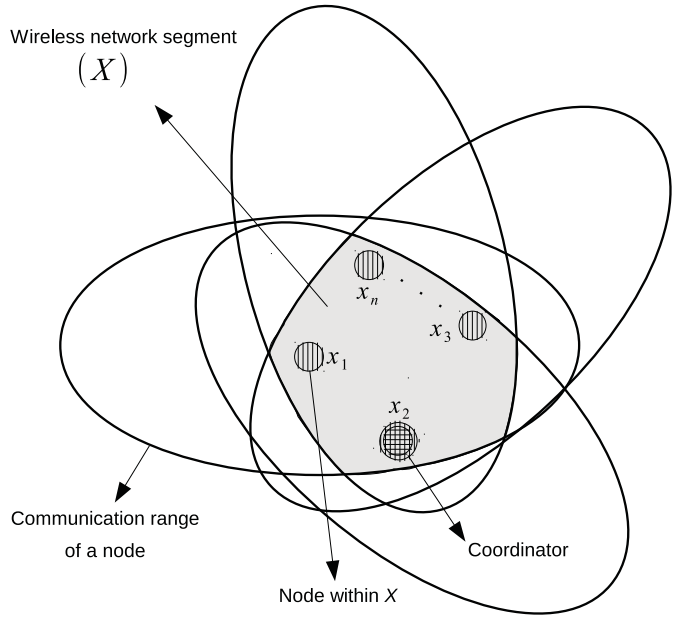


Fig. 1: The graphical representation of a wireless network segment

WMAC1 - ***Broadcast***: correct nodes, receiving an uncorrupted frame transmission, receive the same frame.

WMAC2 - ***Error Detection***: correct nodes detect any corruption done by the network in a locally received frame.

WMAC3 - ***Bounded Omission Degree***: in a known time interval T_{rd}, omission failures may occur in at most k transmissions.

WMAC4 - ***Bounded Inaccessibility***: in a known time interval T_{rd}, a network may be inaccessible at most i times, with a total duration of at most T_{ina}.

WMAC5 - ***Bounded Transmission Delay***: any frame transmission request is transmitted on the network within a bounded delay $T_{td} + T_{ina}$.

Fig. 2: General Wireless MAC-level properties

(*Error Detection*) derives directly from frame protection through a CRC[3] polynomial, as provided by the MAC layer. Frames affected by errors are discarded, usually by the MAC controller itself. This means, frame errors are transformed into omissions. The residual probability of undetected frame errors is negligible [13], [14].

The extension of property WMAC2 to include the signaling 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 layer solutions. No modifications are needed to the IEEE 802.15.4 standard.

Property WMAC3 (*Bounded Omission Degree*) formalizes the failure semantics introduced ear-

[3]CRC - Cyclic Redundancy Check.

lier, being $k \geq f_o$. This property is crucial to implement protocols yielding bounded termination times. For example, the IEEE 802.15.4 specification makes use of the bounded omission degree technique in the definition of a (data/control) frame reliable unicast protocol, at the MAC layer [15].

Considering only the presence of accidental transient faults, the omission degree (i.e. the number of consecutive omission errors during a given protocol execution) of a single channel wireless network infrastructure can be bounded, given its error characteristics [14], [16], [17]. The IEEE 802.15.4 standard defines a MAC protocol configuration parameter equivalent to the channel omission degree bound, k, setting a default value $k \equiv macMaxFrameRetries = 3$ [15].

The *Bounded Omission Degree* property is one of the most complex properties to secure in wireless networks. Securing this property with optimal values and with a high degree of dependability coverage will require the use of multiple communication channels [11]. Although an innovative solution to this problem needs to be further investigated, as soon as achieved it may also provide an effective defence against a class of malicious physical layer attacks, such as radio jamming [11], [18], [19].

The network behaviour in the time domain is described by the remaining properties. Property WMAC5 (*Bounded Transmission Delay*) specifies a maximum network transmission delay, which is T_{td} in the absence of faults. The value of T_{td} may include the queuing, network access and transmission delays and it depends on message latency classes and offered load bounds [3], [20]. 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 [15]. However, T_{td} may include the extra delays resulting from the queuing effects caused by the occurrence of network inaccessibility.

The bounded network transmission delay includes T_{ina}, a corrective term, which accounts for the worst case duration of network inaccessibility glitches, given the bounds specified by property WMAC4 (*Bounded Inaccessibility*). The network inaccessibility characteristics depend on the network alone and can be predicted by the analysis of the MAC protocol. Some preliminary results on this analysis have been advanced in [17]. This work consolidates and extends those earlier results, doing an exhaustive study of network inaccessibility in IEEE 802.15.4 networks.

IV. IEEE 802.15.4 - Overview

The IEEE 802.15.4 [15] is a standard specified for wireless sensor networks (WSNs) with potential utilization on vehicular, industrial, and aerospatial communications. Each IEEE 802.15.4 network must contain a coordinator which defines the network parameters and characteristics such as addressing, supported channels, and operation mode.

There are two operation modes defined in the standard specification called 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 medium access. This control is decentralized, lacking a native support for communications with temporal restrictions.

Conversely, beacon enabled mode has a native specification for supporting communications with temporal restrictions, being the operation mode we are concentrating our further analyses. A coordinator controls the medium access using the superframe structure represented in Fig. 3. 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 synchronize nodes for medium access actions on the whole network. In CAP all nodes compete

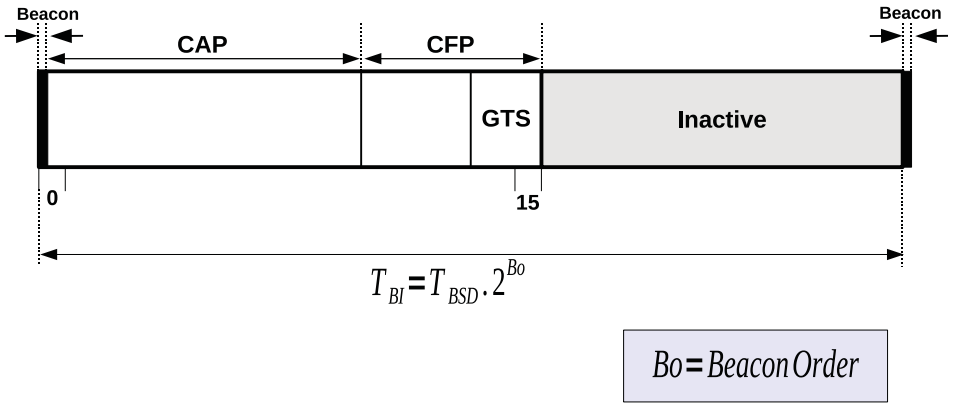


Fig. 3: Superframe Structure of IEEE 802.15.4 Beacon enabled mode

for accessing the medium. For this reason, a slotted version of CSMA/CA protocol must be used to access the medium within CAP [15]. Details of this protocol can be found in [15], [20]–[22].

On the other hand, when CFP exists, i.e., when a coordinator supports the allocation of reserved slots, these slots are called guaranteed time slots (GTS). The CFP always appears at the end of the CAP and each slot must be allocated previously to only one node. This allocation "guarantees" that the medium is free and the aforementioned node can transmit frames without using the CSMA/CA protocol. Furthermore, this feature is used to support the execution of the real time applications [3], [4]. However, the exclusive use of bandwidth reservation is not a complete solution to support the execution of protocols and applications with real-time restrictions.

The inactive period is used to allow all nodes to enter in sleep mode, or shutdown their transceiver, to reduce their energy consumption in a known time interval 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, i. e., there is no period that devices may enter in low-power state. The standard values were used in our analyses and are summarized in Tables I and II.

V. IEEE 802.15.4 Service Interface

The standard IEEE 802.15.4 MAC service interface defines two types of service for the transmission of MAC data and control frame. The set of general equations describing frame transmission times is defined next. Equations 1 and 2 are used for unreliable (non acknowledged) frame transmission, and equations 3 and 4 for reliable (acknowledged) frame transmission.

$$\mathcal{T}^{bc}_{MAC}(type) = \mathcal{T}_{backoff} + \mathcal{T}^{bc}_{MAC-type} \quad (1)$$

$$\mathcal{T}^{wc}_{MAC}(type) = \sum_{j=1}^{maxBackoff} \left\{\mathcal{T}_{backoff}.(2^{BE}+1)\right\} + \mathcal{T}^{wc}_{MAC-type} \quad (2)$$

$$\mathcal{T}^{bc}_{MAC\_ack}(type) = \mathcal{T}^{bc}_{MAC}(type) + \mathcal{T}^{bc}_{ackDelay} + \mathcal{T}_{ack} \quad (3)$$

$$\mathcal{T}^{wc}_{MAC\_ack}(type) = \sum_{j=0}^{maxRetries} \mathcal{T}^{wc}_{MAC}(type) + \mathcal{T}^{wc}_{ackDelay} + \mathcal{T}_{ack} \quad (4)$$

IEEE 802.15.4 Name	Abbr.	Range	Default Value
macBeaconOrder	*BO*	0 - 15	8
macSuperframeOrder	*SO*	0 - 15	5
macMinBE	*minBE*	0 - maxBE	3
macMaxBE	*maxBE*	3 - 8	5
macMaxCSMABackoffs	*maxBackoff*	0 - 5	4
macMaxFrameRetries	*maxRetries*	0 - 7	3
macResponseWaitTime	*nrWait*	2 - 64	32
aMaxLostBeacons	$nrLost$	-	4
aNumSuperframeSlots	$nrSlots$	-	16

TABLE I: Relevant integer parameters of the IEEE 802.15.4 standard

IEEE 802.15.4 Name	Identifier	Value (symbol times)
aBaseSlotDuration	$\mathcal{T}_{base}$	60
aBaseSuperframeDuration	$\mathcal{T}_{BSD}$	960
aMinCAPLength	$\mathcal{T}_{minCAP}$	440
aUnitBackoffPeriod	$\mathcal{T}_{backoff}$	20
aTurnaroundTime	$\mathcal{T}_{xvrcmd}$	12

TABLE II: Relevant time-related constants of the IEEE 802.15.4 standard

where, BE is the backoff exponent that defines the length of the CSMA/CA contention window, being $minBE \leq BE < maxBE$ (Table I); $\mathcal{T}^{bc}_{ackDelay} = \mathcal{T}_{xvrcmd}$ and $\mathcal{T}^{wc}_{ackDelay} = \mathcal{T}_{xvrcmd} + \mathcal{T}_{backoff} + \mathcal{T}_{freq}$ are the times to wait the acknowledgment in reliable transmissions. $\mathcal{T}_{freq}$ depends of technology and to simplify we will consider an upper bound $\mathcal{T}_{freq} = 100$ symbols. The reference $type$ in equations (1) to (4) identifies one specific type of MAC frames.

The superscripts ^{bc} and ^{wc} used in equations 1 to 4 specify the best and worst case MAC frame transmission times, respectively.

VI. Network inaccessibility in IEEE 802.15.4

This section presents an exhaustive study of network inaccessibility in IEEE 802.15.4 wireless networks. A comprehensive set of scenarios leading to network inaccessibility is thoroughly discussed. For many of them we start with very simple situations that then evolve to less restrictive – and thus more general – operating conditions/fault assumptions. For most of the cases, we explicitly derive best and worst case figures, that we will signal with superscripts ^{bc} and ^{wc}, respectively.

A. Single Beacon Frame Loss

Let us start our analysis considering that a subset of nodes (may have a single element) in a wireless network segment does not track beacon frames. If a node in this set needs to transmit a frame it should enable the radio transceiver (receive mode) and start a wait and network synchronization

[1]The worst case duration, for the wait of an acknowledgement frame, follows the IEEE 802.15.4 standard specification [15].

Frame type	Symbol	Length (bit)	Duration (ms)
Data frames			
Data (Minimum payload)	$\mathcal{T}_{data}^{bc}$	8	0.03
Data (Maximum payload)	$\mathcal{T}_{data}^{wc}$	1016	4.07
Data request	$\mathcal{T}_{Ext\_R}$	320	1.28
Data acknowledgment[1]	$\mathcal{T}_{ack}$	40	1.00
MAC control frames			
Beacon	$\mathcal{T}_{Beacon}$	1016	4.07
Beacon request	$\mathcal{T}_{Beacon\_R}$	64	0.26
Network ID conflict notification	$\mathcal{T}_{Conflict}$	304	1.22
Orphan notification	$\mathcal{T}_{Orphan}$	128	0.52
Realign	$\mathcal{T}_{Realign}$	280	1.12
Association request	$\mathcal{T}_{Assoc\_R}$	312	1.25
GTS request	$\mathcal{T}_{GTS\_R}$	72	0.29
Control request	$\mathcal{T}_{Ext\_R}$	320	1.28
MAC frame acknowledgment[1]	$\mathcal{T}_{ack}$	40	1.00

TABLE III: IEEE 802.15.4 frame durations, using the $2.4\,GHz$ frequency band

period of at most $\mathcal{T}_{BSD}\ .\ (2^{BO}+1)$ symbols. If the beacon frame is received before the end of this search period, the frame shall be transmitted in the appropriate portion of the superframe. No network inaccessibility event exists. Otherwise, the operation of the MAC protocol is disturbed by the lack of beacon frame synchronization and the network is inaccessible, as described by equation:

$$\mathcal{T}_{ina\leftarrow sbfl}^{wc} \;=\; \mathcal{T}_{xvrcmd} \;+\; \mathcal{T}_{BSD}\,.\,(2^{BO} \;+\; 1) \tag{5}$$

Since $\mathcal{T}_{xvrcmd} \ll \mathcal{T}_{BSD}$, equation 5 can be simplified to equation 6, which represents the period of network inaccessibility upon the loss of a single beacon in a beacon enabled wireless network segment:

$$\mathcal{T}_{ina\leftarrow sbfl}^{wc} \;=\; \mathcal{T}_{BSD}\,.\,(2^{BO} \;+\; 1) \tag{6}$$

After the period of network inaccessibility, it is assumed a new instance of a beacon frame will be received and the node may proceed with the transmission of the frame using the unslotted version of the CSMA/CA algorithm. The entire period of network inaccessibility is *local* to the node.

B. Multiple Beacon Frame Loss

A beacon-enabled wireless network segment uses the superframe structure for controlling medium access. Under normal operation, a node must receive the beacon frame before it is allowed to transmit data. If some nodes in the wireless network segment do not receive the beacon frame, the network will be inaccessible for such nodes.

Based on the superframe structure of the last received beacon, the node can control the radio interface and track consecutive beacon transmissions. The tracking mechanism is also called beacon synchronization and allows all nodes to know the characteristics of the superframe structure (duration of active and inactive periods, number of allocated GTS slots, etc.).

For tracking a beacon frame, a node searches for beacons during at most $\mathcal{T}_{BSD} \,.\, (2^{BO}+1)$ symbol times. If a beacon frame with the current wireless network segment identifier is not received, this search is repeated from one to at most $nrLost \equiv aMaxLostBeacons$ times. The best and worst case network inaccessibility durations are obtained under the assumption that a beacon frame is successfully received right after the first and the last of the $nrLost$ wait periods. The corresponding periods of network inaccessibility are therefore given by equations 7 and 8, respectively.

$$\mathcal{T}^{bc}_{ina \leftarrow mbfl} = \mathcal{T}_{BSD} \,.\, \left(2^{BO} + 1\right) \tag{7}$$

$$\mathcal{T}^{wc}_{ina \leftarrow mbfl} = \mathcal{T}_{BSD} \,.\, \left(2^{BO} + 1\right) \,.\, nrLost \tag{8}$$

These periods of network inaccessibility may *locally* affect only a given set of nodes (this set may have a single element) or its effects may extend to all the nodes of the wireless network segment, but the wireless network segment coordinator.

C. Synchronization Loss

If the search for the beacon frame does not succeed in any of the $nrLost$ tries, a node loses synchronization with its coordinator, being obliged to signal a BEACON LOST event to high layer protocol management entities, such as the *Mediator Layer* management entities [23]. The corresponding period of network inaccessibility up to this point is simply given by:

$$\mathcal{T}_{ina \leftarrow nosync} = \mathcal{T}_{BSD} \,.\, \left(2^{BO} + 1\right) \,.\, nrLost \tag{9}$$

The BEACON LOST event is signaled upon exceeding the allowed maximum number of beacon frame losses, $aMaxLostBeacons \equiv nrLost$. This is in strict conformity with the standard specification and with our system model.

There are a number of causes for network inaccessibility due to loss of node synchronization: a burst of electromagnetic interference in the medium; disturbances in the node receiver circuitry; collisions derived from the presence of obstacles or influenced by the activity of hidden or mobile nodes; glitches in the coordinator or even its failure. Based on the information it owns, the *Mediator Layer* management entities may take a decision on the appropriate recovery action.

This period of network inaccessibility may affect only a set of nodes or it may include all the nodes of the wireless network segment, but the wireless network segment coordinator.

D. Orphan Node

If the high layer protocol management entities (e.g. the *Mediator Layer*) decide that the device 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. While the node does not receive the MAC realignment command, the network is inaccessible. Once such MAC command is received the node terminates the scan and the network becomes accessible. The MAC realignment frame is transferred using the frame reliable unicast service. Thus, the worst case period of network inaccessibility is obtained assuming that the MAC realignment command is received only while scanning the last of the $nrchannels$ logical channels, being its upper bound given by equation 10.

$$\mathcal{T}^{wc}_{ina\leftarrow orphan} = \mathcal{T}_{ina\leftarrow nosync} + \mathcal{T}_{MLA}(Orphan) + \sum_{j=1}^{nrchannels} (\mathcal{T}^{wc}_{MAC}(Orphan) + nrWait \,.\, \mathcal{T}_{BSD}) + \mathcal{T}^{wc}_{MAC\_ack}(Realign) \tag{10}$$

where, $\mathcal{T}_{MLA}$ is the normalized (symbol) time taken in the *Mediator Layer* management actions. Should the orphan realignment succeed at the first attempt, the period of network inaccessibility will be simply given by equation 11.

$$\mathcal{T}^{bc}_{ina\leftarrow orphan} = \mathcal{T}_{ina\leftarrow nosync} + \mathcal{T}_{MLA}(Orphan) + \mathcal{T}^{bc}_{MAC}(Orphan) + \mathcal{T}_{MLA}(Realign) + \mathcal{T}^{bc}_{MAC\_ack}(Realign) \tag{11}$$

which assumes that $\mathcal{T}_{MLA}(Realign) < nrWait \,.\, \mathcal{T}_{BSD}$, represents the duration of the *Mediator Layer* management actions at the network coordinator, in response to the MAC orphan notification command. The whole period of network inaccessibility may affect only a single node, a given set of nodes or all the nodes of the wireless network segment, but the network coordinator. In the worst-case, all the N nodes of the wireless network segment, but the network coordinator may be inaccessible, as specified by equation 10, where the superscript ^{mn} signals that multiple nodes may be inaccessible:

$$\mathcal{T}^{wc-mn}_{ina\leftarrow orphan} = \mathcal{T}_{ina\leftarrow nosync} + \mathcal{T}_{MLA}(Orphan) + \sum_{j=1}^{nrchannels} (\mathcal{T}^{wc}_{MAC}(Orphan) + nrWait \,.\, \mathcal{T}_{BSD}) + (N-1) \,.\, \mathcal{T}^{wc}_{MAC\_ack}(Realign) \tag{12}$$

However, since MAC control frames are being exchanged between nodes, the time taken in those actions should be seen as a period of network inaccessibility by all the nodes in the wireless network segment. These *global* periods of network inaccessibility are lower and upper bounded by the duration of the events specified in equations 13 and 12, respectively.

$$\mathcal{T}^{bc}_{ina\leftarrow orphan(mac)} = \mathcal{T}^{bc}_{MAC}(Orphan) + \mathcal{T}^{bc}_{MAC\_ack}(Realign) \tag{13}$$

$$\mathcal{T}^{wc-mn}_{ina\leftarrow orphan(mac)} = (N-1) \,.\, [\sum_{j=1}^{nrchannels} (\mathcal{T}^{wc}_{MAC}(Orphan)) + \mathcal{T}^{wc}_{MAC\_ack}(Realign)] \tag{14}$$

where, N is the number of nodes in the wireless network segment. Equation 13 assumes that a single node has been declared as an orphan while equation 14 is derived assuming that all the nodes but the network coordinator have entered into the orphan state. Equations 13 and 14 do not account for local actions, such as the event detection latencies, frame waiting periods and processing overheads, included in equations 10 to 12.

E. Coordinating Orphan Realignment

At the coordinator the need to assist MAC layer management actions starts when a MAC orphan notification command is received. Upon processing by *Mediator Layer* management entities, the reliable unicast, i.e. the acknowledged transmission of a MAC realignment command is requested. The time taken in these actions is seen as network inaccessibility by the wireless network segment coordinator. The best and worst periods of network inaccessibility concerning the interaction with a single orphan node are given by equations 15 and 16, respectively.

$$\mathcal{T}_{ina\leftarrow realign}^{bc} = \mathcal{T}_{MLA}(Realign) + \mathcal{T}_{MAC\_ack}^{bc}(Realign) \quad (15)$$

$$\mathcal{T}_{ina\leftarrow realign}^{wc-sn} = \mathcal{T}_{MLA}(Realign) + \mathcal{T}_{MAC\_ack}^{wc}(Realign) \quad (16)$$

On the other hand, if the operation of the network is disturbed in such a way that all the nodes of the wireless network segment, but the wireless network segment coordinator, enter into the orphan state, the corresponding worst case period of network inaccessibility is given by equation 17.

$$\mathcal{T}_{ina\leftarrow realign}^{wc-mn} = \mathcal{T}_{MLA}(Realign) + (N-1)\,.\,\mathcal{T}_{MAC\_ack}^{wc}(Realign) \quad (17)$$

where, it is assumed that the processing of the different MAC orphan notifications by the *Mediator Layer* management entities mostly proceeds in parallel with the transmission of MAC coordinator realignment frames. All these operations may heavily disturb the superframe structure and the corresponding network operation cycle and may even introduce a significant jitter in the forthcoming beacon frame transmissions. Therefore, this period of network inaccessibility should be seen as *global*, i.e. affecting all network nodes.

F. Coordinator Conflict Detection

In general, there is the possibility that two different potential coordinators may render the same $WNID$, within the same wireless network segment broadcast domain. A similar scenario may also result from node mobility, when a moving node and potential coordinator enters into the broadcast domain of a functioning coordinator. In any of these scenarios, one have a situation called *coordinator conflict*, which can either be detected by the wireless network segment coordinator or by its directly associated nodes.

There are two forms to be aware of a coordinator conflict: a beacon frame with the same $WNID$ is received from different coordinators within the same wireless network segment broadcast domain; a coordinator receives a $WNID$ conflict notification from a node. The former is a local event and does not directly generate a network inaccessibility incident. The latter involves the reliable unicast

of a MAC Coordinator $WNID$ Conflict notification frame, which may individually lead to a period of network inaccessibility, bounded in the best and worst case by equations 18 and 19, respectively.

$$\mathcal{T}^{bc}_{ina\leftarrow C\_Detection} = \mathcal{T}^{bc}_{MAC\_ack}(C\_Conflict) \quad (18)$$

$$\mathcal{T}^{wc-sn}_{ina\leftarrow C\_Detection} = \mathcal{T}^{wc}_{MAC\_ack}(C\_Conflict) \quad (19)$$

These periods of network inaccessibility should be seen as *global* by all the nodes of the wireless network segment broadcast domain, since it implies the transaction of MAC control frames. In a best case scenario the coordinator conflict will be detected by a single node and only one MAC notification is sent in the wireless network segment, as specified by equations 18 and 19. In the worst case, the conflict will be detected and signaled by all the wireless network segment nodes, but the wireless network segment coordinator, and the corresponding period of network inaccessibility is upper bounded by equation 20.

$$\mathcal{T}^{wc}_{ina\leftarrow C\_Detection} = (N-1)\,.\,\mathcal{T}^{wc}_{MAC\_ack}(C\_Conflict) \quad (20)$$

G. Coordinator Conflict Resolution

A wireless network segment coordinator must signal a COORDINATOR ID CONFLICT to *Mediator Layer* management entities, which in turn will request the MAC layer to perform an active scan. This scan is realized in all currently used logical channels. Scanning each channel involves the transmission of a MAC beacon request command and wait for replies (beacon frames), during a given period.

The identifiers recorded from the received beacons can be issued to the *Mediator Layer* management entities all at once, as specified in equation 21, or each time a beacon frame is received, as drawn in equation 22. During all this process, the network is inaccessible. The best and worst case periods of network inaccessibility are given by equations 21 and 22, respectively.

$$\mathcal{T}^{bc}_{ina\leftarrow C\_Resolution} = \mathcal{T}_{MLA}(C\_Conflict) + \mathcal{T}^{bc}_{MAC}(Beacon\_R) + nrWait.\mathcal{T}_{BSD} + \mathcal{T}_{MLA}(Realign) + \mathcal{T}^{bc}_{MAC}(Realign) \quad (21)$$

$$\mathcal{T}^{wc}_{ina\leftarrow C\_Resolution} = \mathcal{T}_{MLA}(Conflict) + \sum_{j=1}^{nrchannels}[\,\mathcal{T}^{wc}_{MAC}(Beacon\_R) + nrWait.\mathcal{T}_{BSD}\,] + \mathcal{T}_{MLA}(Realign) + \mathcal{T}^{wc}_{MAC}(Realign) \quad (22)$$

If, at the end of the search, the network coordinator does not found a beacon frame with its own identifier no further action is taken and the network becomes accessible again. Otherwise, a new identifier is selected and, if necessary, a MAC coordinator realignment command is broadcast. Since all these events are originated at the network coordinator, they should be regarded as *global*, i.e. observed by all the nodes in the wireless network segment. If the network coordinator selects a new identifier, some nodes may not be synchronized with the ”new” superframe structure, which may induce a loss of synchronization, as explained in Section VI-C.

H. Extract Request

There are two ways to transmit data between a node and a coordinator: the direct and indirect transmission. In the direct transmission, the coordinator sends a data to a node directly, i.e., the coordinator access the medium an send a data frame using a slotted version of CSMA/CA algorithm. Otherwise, in the indirect transmission the coordinator storage the data in a queue and waits the reception of a command that request the extraction of this data. In this case, the node sends a command to extract data of the coordinator and waits for the reception of an acknowledgement. The node repeat this operation until $maxRetries$ times.

Thus, while the node does not receive the acknowledgement frame the network is inaccessible to it. Additionally, if the node receives an acknowledgement from the coordinator, this node enables its transceiver in receive mode during $\mathcal{T}_{wait}$ and the network may continue inaccessible within this period. The best and worst case network inaccessibility durations are therefore given by equations 23 and 24, respectively.

$$\mathcal{T}^{bc}_{ina\leftarrow extReq} = \mathcal{T}^{bc}_{MAC_{ack}}(ExtReq) \tag{23}$$

$$\mathcal{T}^{wc}_{ina\leftarrow extReq} = \mathcal{T}^{wc}_{MAC_{ack}}(ExtReq) + \mathcal{T}_{wait} \tag{24}$$

where, $\mathcal{T}_{wait}$ is the period, addressed by the attribute $macMaxFrameTotalWaitTime$, that is dependent upon a combination of physical and MAC attributes and constants, being defined in the IEEE 802.15.4 standard [15].

I. Association

The association procedure starts with a active scan in each logical channel available. The active scan involves the send of a MAC beacon request command, for each available logical channel, and the wait for replies (beacon frames), during a given period. After processing the beacon frames, the *Mediator Layer* management entities select a wireless network segment, send an Association Request command, and wait for a confirmation (acknowledgement). However, the association procedure is done only after to extract the information about this association using the indirect transmission method (see subsection VI-H). The best and worst periods of network inaccessibility are given by equations 25 and 26, respectively.

$$\begin{aligned}\mathcal{T}^{bc}_{ina\leftarrow assoc} = {} & \mathcal{T}^{bc}_{MAC}(Beacon\_R) + nrWait.\mathcal{T}_{BSD} + \mathcal{T}_{MLA}(Beacon) + \\ & \mathcal{T}^{bc}_{ina\leftarrow extReq} + \mathcal{T}_{MLA}(AssocReq) + \mathcal{T}^{bc}_{MAC\_ack}(AssocReq)\end{aligned} \tag{25}$$

$$\begin{aligned}\mathcal{T}^{wc}_{ina\leftarrow assoc} = \sum_{j=1}^{nrchannels} & [\mathcal{T}^{wc}_{MAC}(Beacon\_R) + nrWait.\mathcal{T}_{BSD}] + \mathcal{T}_{MLA}(Beacon) + \\ & \mathcal{T}^{wc}_{ina\leftarrow extReq} + \mathcal{T}_{MLA}(AssocReq) + \mathcal{T}^{wc}_{MAC\_ack}(AssocReq)\end{aligned} \tag{26}$$

J. Re-Association

After a synchronization loss, the *Mediator Layer* management entities should decide which will be done: to consider that the device is orphan; or that an association procedure will be realized

again. In case of re-association, a MAC layer should perform a reset operation before beginning the association procedure. The best and worst network inaccessibility times are given by equations 27 and 28, respectively.

$$\mathcal{T}^{bc}_{ina\leftarrow reAssoc} = \mathcal{T}_{ina\leftarrow nosync} + \mathcal{T}^{bc}_{ina\leftarrow assoc} \tag{27}$$

$$\mathcal{T}^{wc}_{ina\leftarrow reAssoc} = \mathcal{T}_{ina\leftarrow nosync} + \mathcal{T}^{wc}_{ina\leftarrow assoc} \tag{28}$$

K. GTS request

The allocation of a GTS slot is performed using the reliable unicast service to send a MAC GTS request command to the associated coordinator. During this period, the network is seen as inaccessible. The best and worst periods of network inaccessibility are given by equations 29 and 30, respectively.

$$\mathcal{T}^{bc}_{ina\leftarrow GTS} = \mathcal{T}^{bc}_{MAC\_ack}(GTS) \tag{29}$$

$$\mathcal{T}^{wc}_{ina\leftarrow GTS} = \mathcal{T}^{wc}_{MAC\_ack}(GTS) \tag{30}$$

These periods of network inaccessibility are seen as *global* by all the nodes of the wireless network segment. This scenario is extremely important because GTS slots can be used for bandwidth reservation. Several solutions advanced in the literature try to solve the problem of real-time communications, over the IEEE 802.15.4 standard, using GTS allocation mechanisms [3]–[5], [24]. The effectiveness of such solutions should be re-analysed under the scope of a comprehensive network inaccessibility model.

VII. Results: network inaccessibility duration in beacon enabled networks

The characterization of network inaccessibility presented in the section VI allows us to extract some useful information regarding to the temporal behavior of an IEEE 8022.15.4 wireless network. The default values of the IEEE 802.15.4 standard summarized in Tables I and II were utilized for the parameters and constants present in our network inaccessibility characterization. We establish an uniform duration for the management actions, represented by the $\mathcal{T}_{MLA}$ term, which is $\frac{1}{10}$ of the beacon interval, i.e., $\frac{2^{BO}.\mathcal{T}_{BSD}}{10}$. To be able to reproduce all the values obtained by our analysis Table IV also presents the number of channels ($nrChannels$ parameter) for each frequency band supported by the IEEE 802.15.4 standard. The value of each parameter that is represented in symbols can be converted in bits utilizing Table V, which presents the numbers of symbols per octet in all modulation technique and frequency band.

The impact of the network inaccessibility scenarios in the network temporal behavior is presented within Tables VI to VIII, which groups all frequency bands supported by the IEEE 802.15.4 standard. The results inscribed in these tables show that the periods of network inaccessibility

Frequency Band	Number of channels
868-868.6 MHz	1
902-928 MHz	10
2400-2483.5 MHz	16

TABLE IV: Number of channels per frequency band supported by the IEEE 802.15.4 standard

Modulation Technique	Frequency Band		
	868 MHz $(symbols/octet)$	**915 MHz** $(symbols/octet)$	**2400 MHz** $(symbols/octet)$
BPSK	8	8	—
ASK	0.4	1.6	—
O-QPSK	2	2	2

TABLE V: The number of symbols per octet in each modulation technique and frequency band

are extremely high, precluding any claim of obtaining from the network a real-time behavior, even if some specifically designed mechanisms are in place, since network inaccessibility incidents may always occur.

With the default network configuration of Table I, the worst case period of network inaccessibility is up to seven times higher than the beacon interval. Figure 4 presents this comparison. However, it should be noted that the beacon interval is in the order of the seconds, 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.

Defining methods and to reduce the duration of the periods of network inaccessibility in IEEE 802.15.4 wireless network is of crucial importance for achieving real-time operation. This study is a first but fundamental step towards that direction.

PHY (868-868.6 MHz)						
Scenario	**Modulation Technique**					
	BPSK - $20\,kb/s$		*ASK* - $250\,kb/s$		*O-QPSK* - $100\,kb/s$	
	best case (ms)	**worst case** (ms)	**bc** (ms)	**wc** (ms)	**bc** (ms)	**wc** (ms)
$\mathcal{T}_{ina \leftarrow sbfl}$	——	12337	——	19739	——	9870
$\mathcal{T}_{ina \leftarrow mbfl}$	12337	49345	19739	78952	9870	39476
$\mathcal{T}_{ina \leftarrow nosync}$	49345	49345	78952	78952	39476	39476
$\mathcal{T}_{ina \leftarrow orphan}$	51834	52851	82896	84441	41452	42233
$\mathcal{T}_{ina \leftarrow realign}$	1250	1833	1974	2824	990	1423
$\mathcal{T}_{ina \leftarrow C\_Detection}$	20	609	8	858	7	441
$\mathcal{T}_{ina \leftarrow C\_Resolution}$	2772	2900	4428	4632	2215	2317
$\mathcal{T}_{ina \leftarrow extReq}$	13	612	6	858	5	442
$\mathcal{T}_{ina \leftarrow assoc}$	4032	5351	6407	8313	3208	4182
$\mathcal{T}_{ina \leftarrow reAssoc}$	53377	54695	85358	87265	42684	43658
$\mathcal{T}_{ina \leftarrow GTS}$	13	557	6	845	4	428

TABLE VI: The best and worst cases for $868MHz$ frequency band

PHY (902-928 MHz)						
Scenario	**Modulation Technique**					
	BPSK - $40\,kb/s$		*ASK* - $250\,kb/s$		*O-QPSK* - $250\,kb/s$	
	best case (ms)	**worst case** (ms)	**bc** (ms)	**wc** (ms)	**bc** (ms)	**wc** (ms)
$\mathcal{T}_{ina \leftarrow sbfl}$	——	6169	——	19739	——	3948
$\mathcal{T}_{ina \leftarrow mbfl}$	6139	24673	19739	78952	3948	15791
$\mathcal{T}_{ina \leftarrow nosync}$	——	24673	——	78952	——	15791
$\mathcal{T}_{ina \leftarrow orphan}$	25917	33958	82896	108427	16581	21696
$\mathcal{T}_{ina \leftarrow realign}$	625	917	1974	2823	396	570
$\mathcal{T}_{ina \leftarrow C\_Detection}$	10	305	8	857	3	177
$\mathcal{T}_{ina \leftarrow C\_Resolution}$	1386	8968	4428	28616	886	5727
$\mathcal{T}_{ina \leftarrow extReq}$	7	306	5	858	2	177
$\mathcal{T}_{ina \leftarrow assoc}$	2016	10193	6406	32296	1284	6473
$\mathcal{T}_{ina \leftarrow reAssoc}$	26689	34865	85358	111248	17074	22264
$\mathcal{T}_{ina \leftarrow GTS}$	7	279	5	845	2	171

TABLE VII: The best and worst cases for $915MHz$ frequency band

PHY (2400-2483.5 MHz)		
Scenario	**Modulation Technique**	
	O-QPSK - $250\,kb/s$	
	best case (ms)	**worst case** (ms)
$\mathcal{T}_{ina \leftarrow sbfl}$	——	3948
$\mathcal{T}_{ina \leftarrow mbfl}$	3948	15791
$\mathcal{T}_{ina \leftarrow nosync}$	——	15791
$\mathcal{T}_{ina \leftarrow orphan}$	16581	24897
$\mathcal{T}_{ina \leftarrow realign}$	396	570
$\mathcal{T}_{ina \leftarrow C\_Detection}$	3	177
$\mathcal{T}_{ina \leftarrow C\_Resolution}$	886	8927
$\mathcal{T}_{ina \leftarrow extReq}$	2	177
$\mathcal{T}_{ina \leftarrow assoc}$	890	9280
$\mathcal{T}_{ina \leftarrow reAssoc}$	16681	25070
$\mathcal{T}_{ina \leftarrow GTS}$	2	171

TABLE VIII: The best and worst cases for $2.4GHz$ frequency band

VIII. Conclusion

This report presented the characterization of network inaccessibility in the IEEE 802.15.4 networks. The existence and duration of network inaccessibility are still neglected by existent temporal characterization of wireless communications. Network inaccessibility has a strong negative impact in the temporal behavior of IEEE 802.15.4 networks, being extremely important its characterization. In that way, future work directions will focus on providing means to reduce the periods of network inaccessibility; to provide support to signal the periods of network inaccessibility for higher layers, improving the means of analyzing network delays and message schedulability over wireless networked communications.

References

[1] A. Sahoo and P. Baronia, "An energy efficient MAC in wireless sensor networks to provide delay guarantee," in *15th IEEE Workshop on Local Metropolitan Area Networks (LANMAN)*, June 2007, pp. 25 –30.

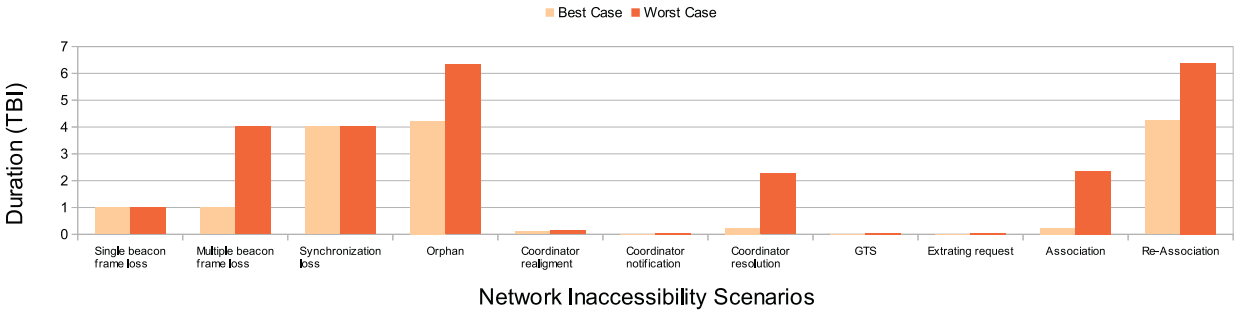


Fig. 4: Network inaccessibility scenarios for the $2.4GHz$ frequency band, normalized by the beacon interval duration, $\mathcal{T}_{BI} = 3932ms$ and $BO = 8$

[2] E. Egea-López, J. Vales-Alonso, A. S. Martínez-Sala, J. García-Haro, P. Pavón-Mariño, and M. V. Bueno Delgado, "A wireless sensor networks MAC protocol for real-time applications," *Personal Ubiquitous Computing*, vol. 12, pp. 111–122, January 2008.

[3] 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 International Conference on Emerging Technologies and Factory Automation (ETFA)*, September 2008, pp. 1016 –1022.

[4] Y.-K. Huang, A.-C. Pang, and H.-N. Hung, "An adaptive GTS allocation scheme for IEEE 802.15.4," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 5, May 2008.

[5] J. Chen, L. Ferreira, and E. Tovar, "An explicit GTS allocation algorithm for IEEE 802.15.4," in *16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, September 2011, pp. 1 –8.

[6] M.-G. Park, K.-W. Kim, and C.-G. Lee, "Holistic optimization of real-time IEEE 802.15.4/zigbee networks," in *IEEE International Conference on Advanced Information Networking and Applications (AINA)*, March 2011, pp. 443 –450.

[7] P. Veríssimo, L. Rodrigues, and M. Baptista, "AMp: A Highly Parallel Atomic Multicast Protocol," *SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 4, pp. 83–93, 1989.

[8] P. Veríssimo and J. A. Marques, "Reliable broadcast for fault-tolerance on local computer networks," in *In Proceedings of the Ninth Symposium on Reliable Distributed Systems*. Alabama, USA: IEEE, 1990, pp. 24–90.

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

[10] J. Rufino, C. Almeida, P. Veríssimo, , and G. Arroz, "Enforcing dependability and timeliness in controller area networks." in *Proc. of the 32nd Annual Conf. of the IEEE Ind. Electronics Society (IECON)*, Paris, France, Nov. 2006.

[11] J. L. R. Souza and J. Rufino, "Building Fundamental Properties For Real-Time Wireless Sensor Networks," AIR-II Technical Report RT-10-01, Tech. Rep., 2010.

[12] O. Babaoğlu and R. Drummond, "Streets of Byzantium: Network Architectures for Fast Reliable Broadcasts," *IEEE Trans. on Soft. Engineering*, vol. SE-11, no. 6, Jun. 1985.

[13] T. Fujiwara, T. Kasami, A. Kitai, and S. Lin, "On the undetected error probability for shortened hamming codes," *IEEE Trans. on Comm.*, vol. 33, no. 6, Jun. 1985.

[14] D. Eckhardt and P. Steenkiste, "Measurement and analysis of the error characteristics of an in-building wireless network," in *SIGCOMM '96: Conf. Proc. on Applications, Tech., Arch. and Protocols for Computer Comm.*, New York, NY, USA, 1996.

[15] 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.

[16] M. Petrova, J. Riihijarvi, P. Mahonen, and S. Labella, "Performance study of IEEE 802.15.4 using measurements and simulations," in *Proceedings of the Wireless Communications and Networking Conference (WCNC 2006)*. Las Vegas, NV, USA: IEEE, Apr. 2006, pp. 487 – 492.

[17] 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 International Embedded Systems Symposium(IESS)*, ser. IFIP Advances in Information and Communication Technology, vol. 310, Langenargen, Germany, September 2009.

[18] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: attack and defense strategies," *IEEE Network*, vol. 20, no. 3, pp. 41–47, May 2006.

[19] S. Khattab, D. Mosse, and R. Melhem, "Modeling of the channel-hopping anti-jamming defense in multi-radio wireless networks," in *5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services (MobiQuitous)*. Dublin, Ireland: ACM, July 2008.

[20] I. Ramachandran, A. K. Das, and S. Roy, "Analysis of the contention access period of IEEE 802.15.4 MAC," *ACM Transactions on Sensor Networks*, vol. 3, March 2007. [Online]. Available: http://doi.acm.org/10.1145/1210669.1210673

[21] J. He, Z. Tang, H.-H. Chen, and Q. Zhang, "An accurate and scalable analytical model for IEEE 802.15.4 slotted CSMA/CA networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 1, pp. 440–448, January 2009.

[22] C. Jung, H. Hwang, D. Sung, and G. Hwang, "Enhanced markov chain model and throughput analysis of the slotted CSMA/CA for IEEE 802.15.4 under unsaturated traffic conditions," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 1, January 2009.

[23] J. L. R. Souza and J. Rufino, "An approach to enhance the timeliness of wireless communications," in *The Fifth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM)*, Lisbon, November 2011.

[24] 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*, vol. 39, no. 1-3, pp. 169–204, August 2008.

A.1.3 Characterizing Inaccessibility in IEEE 802.15.4 Through Theoretical Models and Simulation Tools

"Characterizing Inaccessibility in IEEE 802.15.4 Through Theoretical Models and Simulation Tools". J. L. R. Souza, A. Guerreiro and J. Rufino. INForum 1012 Simpósio de Informática – Embeddeed and Real-Time Systems Track. September 2012, Caparica, Portugal.

This page is intentionally left blank.

Characterizing Inaccessibility in IEEE 802.15.4 Through Theoretical Models and Simulation Tools *

Jeferson L. R. Souza, André Guerreiro, and José Rufino

University of Lisbon - Faculty of Sciences
Large-Scale Informatics System Lab. (LaSIGE)
jsouza@lasige.di.fc.ul.pt, fc33698@alunos.fc.ul.pt, ruf@di.fc.ul.pt

Abstract Wireless networks are everywhere. Mobility and flexibility, together with the elimination of wires derived from the nature of wireless communications, represent extremely useful advantages in recent past, present, and future of networked communications. However, the open and shared communication medium used by wireless networks is highly susceptible to interferences, which may induce the occurrence of temporary network partitioning (dubbed network inaccessibility), caused by disturbances in the medium access control (MAC) layer management operations. Existing analyzes of network operation in the time domain do not usually pay much attention to temporal issues caused by the occurrence of network inaccessibility, and therefore violation in timeliness and dependability properties may occur. In this paper we present our advances in the enhancement of the IEEE 802.15.4 NS-2 simulation module to measure and validate the occurrence of network inaccessibility in a IEEE 802.15.4 simulated network. Additionally, the correctness of a previous theoretical model designed to characterize network inaccessibility within IEEE 802.15.4 networks is also validated, improving the temporal modeling of IEEE 802.15.4 wireless communications.

Keywords: inaccessibility; timeliness; dependability; wireless communications; real-time.

1 Introduction

Wireless networks have been considered the communication technology of the future. Environments such as autonomous vehicles, natural resources and health monitoring, planetary exploration and aerospace [17,3,14] are examples where such kind of technology has a potential applicability, with a special interest in wireless sensor and actuator networks (WSANs) [10].

* This work was partially motivated by our work within the scope of the ESA (European Space Agency) Innovation Triangle Initiative program, through ESTEC Project AIR-II (ARINC 653 in Space — Industrial Initiative), URL: http://air.di.fc.ul.pt. This work was partially supported by EC, through project IST-STREP-288195 (KARYON), URL: http://www.karyon-project.eu/, and by FCT through the Multiannual Funding and CMU-Portugal Programs and the Individual Doctoral Grant SFRH/BD/45270/2008.

All of the aforementioned environments have temporal restrictions, needing communication systems capable to provide guarantees in accordance with such temporal requirements. In case of wireless sensor networks (WSANs included), some approaches try to improve the reliability and temporal aspects of such sensor networks, including new protocol proposals [1,2,15], while others proposes modifications/extensions of existent protocols and standards [4,11,6,5,9]. However, none of these works pay attention to network inaccessibility, which is caused by disturbances in the management operations of the medium access control (MAC) layer, and may induce violations in the timeliness and dependability properties of wireless sensor networks.

In this paper, we present our advances in the improvement of the NS-2 IEEE 802.1.4 module [16] to measure and validate the occurrence of network inaccessibility [13] fault scenarios in a IEEE 802.15.4 simulated network. Simulation results are then compared to the network inaccessibility theoretical model presented in [12], confirming the correctness of this previous theoretical work.

To present our contributions, this paper is organized as follows: Section 2 presents a brief description of the system model used in our analysis and simulations. Section 3 presents an overview of the IEEE 802.15.4 standard. Section 4 describes briefly what is network inaccessibility. Section 5 describes our advances in the IEEE 802.15.4 NS-2 module to measure and validate the duration of network inaccessibility in a simulated network. Section 6 presents a summary of the theoretical model [12] for self-containment purposes. Section 7 presents, compares, and discusses the results obtained from the theoretical model and through the simulation work. Finally, section 8 draws some conclusions and future directions.

2 System model

In this section we provide a formal description of our system model, which establishes a base foundation for our analysis 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 nodes. In the rest of the paper the terms wireless node and node will be used interchangeably. The set of nodes X itself establishes a node relationship entity dubbed wireless network segment, using a given communication channel and a given wireless network segment identifier.

2.1 Assumptions

In our system model the behavior of a wireless network segment is sustained by assumptions utilized to characterize the network communication capabilities and restrictions of wireless nodes. During a wireless network segment operation cycle we use the following assumptions:

1. The communication range of X, i.e. its broadcast domain, is given by: $B_X = \bigcap_{j=1}^{n} B_D(x)$, $\forall x \in X$, 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 behave correctly or crash upon exceeding a given number 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, 2, and 3 define the physical relationship between nodes within the wireless network segment. Our system model characterizes the relationship between nodes at MAC level, where nodes must be in the communication range of each other to communicate and are able to sense one another (assumption 3). Mobility may drive nodes away of wireless network segment (assumption 2).

In the context of network components, an omission is an error that destroys a data frame. Omissions may be caused by different sources such as node mobility, external electromagnetic interference, fading caused by multipath or transient obstacles on the communication medium, glitches on the MAC layer operation, and malicious attacks. Despite of their importance we are not considering malicious attacks in our analysis, being such topic addressed in future work.

Omission errors are detected either using timeout-based techniques or cyclic redundancy check (CRC) mechanisms. Establishing a bound for the *omission degree* of individual components 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. We can also see in Fig. 1 the indication of which node is the coordinator. The management activities of the coordinator comprises

[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.

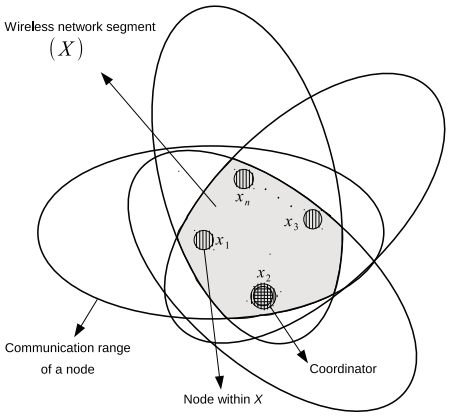


Figure 1: The graphical representation of a wireless network segment

the assignment of the current communication channel in use by the segment, the wireless network segment identifier definition, address space delimitation, and so on.

3 IEEE 802.15.4 Overview

The IEEE 802.15.4 standard specifies that each network must contain a coordinator, which defines a set of characteristics of the network such as addressing, supported channels, and operation mode. The coordinator should be the node with the highest power and energy capabilities to support the execution of management operations required to maintain the network active. There are two operation modes supported by the standard [7]: nonbeacon-enabled and beacon-enabled. Here we are focused on the beacon-enabled mode, designed to support data transmissions with temporal restrictions, and which is the target mode of our analysis and simulations.

In the beacon-enabled mode the medium access is controlled by a special structure dubbed superframe, which is bounded by beacon frames transmitted by the coordinator. The information inside the beacon helps the nodes to know the entire duration of the superframe, allowing the synchronization and the control of the medium access. The organization of the superframe structure is illustrated in Fig. 2. The duration of the superframe is specified using the following equation:

$$\mathcal{T}_{BI} = \mathcal{T}_{BSD} \cdot 2^{BO} \tag{1}$$

where $\mathcal{T}_{BI}$ is the beacon interval, $\mathcal{T}_{BSD}$ is the base value defining the minimum duration of a beacon interval, and BO is the beacon order, which is the main parameter to specify the duration of beacon interval and in this sense specifies how often the beacon frame should be transmitted.

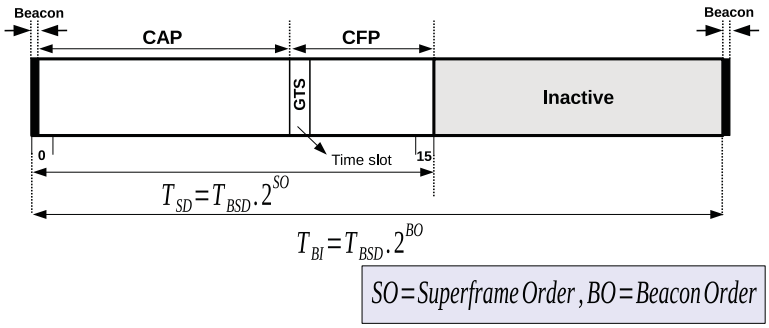


Figure 2: Superframe structure

The superframe organization presented in Fig. 2 identifies two main parts: the active and inactive periods. The active period is mandatory and it is, in turn, constituted by the Contention Access Period (CAP) and the Contention Free Period (CFP). CAP allows all nodes to compete for the utilization of the shared medium. CFP was designed for bandwidth reservation, and therefore time slots dubbed guarantee time slots (GTSs) are utilized to guarantee exclusive medium access. Completing the superframe structure the inactive period (IP) is designed to optimize energy consumption.

4 Network inaccessibility

Disturbances on the network operation may affect two different types of frames: data and control frames. Control frames are utilized by the MAC layer to manage and maintain the network operational. When disturbances cause errors within control frame transmissions, the MAC layer must perform actions to reestablish network operation upon the occurrence of those errors. The period comprised from the instant that the aforementioned errors occurs, to the instant that the network operation is reestablished, is dubbed a period of network inaccessibility. During a period of network inaccessibility a node cannot access the network, and is not able to communicate with other nodes, causing a temporary blackout on the network communication services.

To establish known the bounds for such blackout periods, a network inaccessibility characterization must be performed. The MAC layer analysis should be concentrated in the management protocol of the MAC layer, being the characterization of network inaccessibility events specific for each kind of network technology.

5 Improving the IEEE 802.15.4 NS-2 simulation module

The NS-2 [8] provides a powerful event discrete simulation platform to simulate the dynamics of a computer network. Its modular architecture allows the

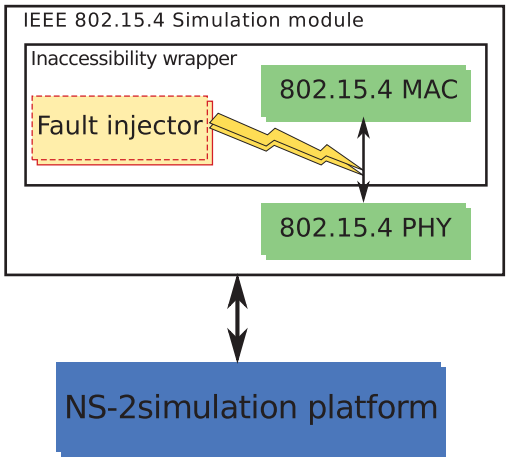


Figure 3: The simulation protocol stack highlighting our modifications in the IEEE 802.15.4 NS-2 Module

modeling of wired and wireless networks and protocols, being an efficient tool to specify and study the behavior of a communication network under different environmental conditions.

The IEEE 802.15.4 module for the NS-2 simulator was designed and developed at the Joint Laboratory of Samsung and the City University of New York [16]. The MAC entity present within this NS-2 module provides all data and management primitives defined in the IEEE 802.15.4 standard. Thus, we use and modify such module to investigate the influence of network inaccessibility in a IEEE 802.15.4 simulated network.

Figure 3 shows the IEEE 802.15.4 NS-2 module. We also present in this figure our improvements designed to simulate, validate, and measure the occurrence of network inaccessibility events. We surround the MAC entity with an inaccessibility wrapper, which internally has two different components: a fault injector and a time measurement component.

The fault injector is capable to disturb the operation of the MAC layer, being its internal structure presented in Fig. 4. The fault injector component performs the fault injection based on the definition of a fault pattern. The criteria to define the fault pattern is totally configurable, allowing the definition of deterministic or probabilistic fault patterns. Probabilistic patterns allow the simulation of arbitrary fault injections, which can be utilized to simulate different environmental aspects inducing the occurrence of network inaccessibility. We are intended to address the use of such probabilistic patterns in future work. In this paper we are interested to analyze the network inaccessibility in a pragmatic way, demanding the use of deterministic fault patterns. The fault injector is then configured to corrupt only beacon frames, injecting faults between the MAC and PHY communication interface (Figure 3).

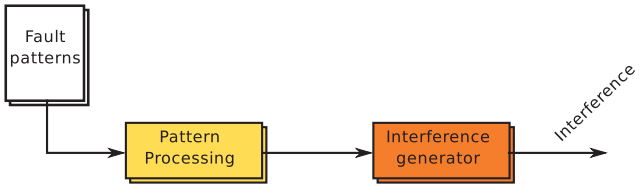


Figure 4: The fault injector component

For example, in the context of this paper there are two different ways to perform the fault injection on the simulated scenario. The first one is performed in the coordinator (whole wireless network segment), where none of the nodes receive the beacon and therefore the whole wireless network segment becomes inaccessible. The second way is performed within nodes tracking the reception of beacon frames (specific nodes). When the corruption is performed in such nodes, only nodes with the fault injector component activated, i.e. beacon corruptions occurring, cannot access the medium and become inaccessible.

Additionally, the time measurement service component allows the measure of the network inaccessibility durations derived from faults generated by the fault injector component. When the fault injector generates a fault that causes a network inaccessibility event, the time measurement service component starts a timer to account for the duration of such network inaccessibility period. A successful reestablishment of the MAC layer communication services indicates the end of a network inaccessibility, and therefore stopping the timer that reveals the duration of its correspondent network inaccessibility event.

6 Theoretical modeling of network inaccessibility in IEEE 802.15.4

A previous study [12] presents a comprehensive characterization of the network inaccessibility in IEEE 802.15.4 beacon-enabled networks. For self-completeness of the issues at hand, this section summarizes such analysis, addressing the most relevant network inaccessibility scenarios in IEEE 802.15.4 beacon-enabled mode, i.e. those scenarios related with the loss of beacon frames.

The first scenario we address assumes a **single beacon frame loss (*SBFL*)**. The beacon frame issued by the coordinator is not received by some node, disturbing the node synchronization imposed by the beacon frame transmissions. The duration of the corresponding inaccessibility scenario is given by the following equation:

$$\mathcal{T}_{ina \leftarrow sbfl}^{wc} = \mathcal{T}_{BSD} \cdot (2^{BO} + 1) \tag{2}$$

In the SBFL scenario, the network inaccessibility ends upon reception of the next beacon transmission by the coordinator. When multiple beacon frames are

Table 1: Normalized theoretical best and worst case results for each network inaccessibility scenario

Scenarios	*Periods ($\mathcal{T}_{BI}$-time)*	
	Best Case	**Worst Case**
Single beacon frame loss	1.1	1.1
Multiple beacon frame loss	1.1	4.1
Synchronization loss	4.1	4.1

not correctly received, a node enters a **multiple beacon frame loss (*MBFL*)** scenario. The occurrence of consecutive beacon frame omissions may therefore vary between 1 and a threshold not exceeding the value defined by $nrLost$. The best case of the MBFL considers only the omission of one beacon frame while the worst case assumes the need to wait for $nrLost$ beacon frame transmissions; only the last one is successfully received. Thus, the MBFL scenario is characterized by the following equations:

$$\mathcal{T}^{bc}_{ina \leftarrow mbfl} = \mathcal{T}_{BSD} . \left(2^{BO} + 1\right) \tag{3}$$

$$\mathcal{T}^{wc}_{ina \leftarrow mbfl} = \left(\mathcal{T}_{BSD} . \left(2^{BO} + 1\right)\right) . nrLost \tag{4}$$

Finally, the **synchronization loss (*SyncLoss*)** is a variation of MBFL that occurs when the $nrLost$ threshold is exceeded. We assume the node immediately enters a state where it completely loses the synchronization with the coordinator. The duration of this scenario is specified by the following equation:

$$\mathcal{T}_{ina \leftarrow nosync} = \left(\mathcal{T}_{BSD} . \left(2^{BO} + 1\right)\right) . nrLost \tag{5}$$

Table 1 summarizes the best and worst case durations of the presented network inaccessibility scenarios. The values presented in Table 1 were normalized using the superframe duration, $\mathcal{T}_{BI}$, which represents the network cycle for the medium access. We perform network simulations to validate these theoretical results, enabling the use of our easy-to-use inaccessibility formulas to enhance timeliness models of IEEE 802.15.4 networks.

7 Results

To measure the duration of each network inaccessibility scenario, the NS-2 module was instrumented using the internal timer class of the NS-2 simulator. In case of SBFL scenario, the first timestamp is obtained when the first beacon is dropped, as a consequence of a fault generated by the fault injector. Thus, when the next beacon arrives at the node, and is not corrupted, other timestamp is acquired. The difference among these two timestamps is the duration of the

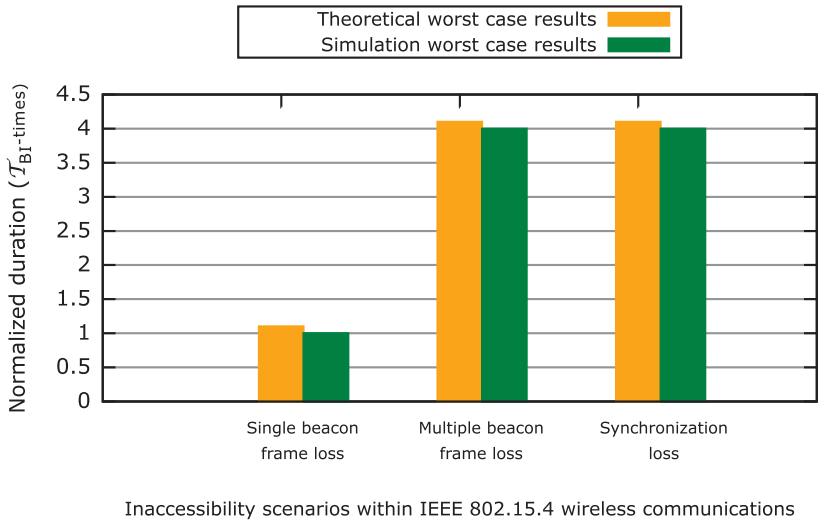


Figure 5: Comparison among simulation an theoretical worst case results

SBFL network inaccessibility scenario. The process is similar to the MBFL and SyncLoss scenarios.

The sensor network was simulated with ten nodes, where one of these nodes was the coordinator. The simulation is in compliance with our system model, where all the nodes are within the range of each other. 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.

In order to simulate the corruption and consequently the loss of beacons, we configure our fault injector component to inject deterministic faults to corrupt beacons. Thus, a know number of beacons are corrupted, causing the occurrence of the network inaccessibility scenarios described in the Section 6. The fault injector achieves the beacon corruption by changing some bits in the beacon frames, implying the drop of these packets in the MAC level of the receiving nodes.

Figure 5 presents the comparison among simulation and theoretical results obtained by the computation of the formulas presented in Section 6. As the NS-2 module of the IEEE 802.15.4 follows the standards, it allows a fair comparison among the results obtained in both simulation and theoretical model. A beacon order $BO = 8$ was utilized to normalize the network inaccessibility durations. It is possible to verify that the theoretical values present the upper bound of the network inaccessibility scenarios compared with the simulation results. The simulation performed with the NS-2 simulator then confirms the correctness of the theoretical model specified to characterize the inaccessibility scenarios in the IEEE 802.15.4 standard.

A beacon loss may cause a minor temporal deviation between node clocks, which is accounted by the IEEE 802.15.4 standard within the beacon search routine. The theoretical search of the next correct beacon considers the possibility of such clock deviation, which is covered by the addition of a corrective value that increases the beacon search by one $\mathcal{T}_{BSD}$ unit. However, all nodes have their clocks synchronized within the simulation, implying in no temporal deviations among them, even in case of beacon losses. Thus, the minor deviation among the theoretical and simulation results is derived from the use of such corrective value, which is incorporated within the equations utilized to account network inaccessibility in IEEE 802.15.4 wireless communications.

With the potential of the wireless networks to support the communication in scenarios with temporal restrictions, this validation is important, allowing an effective use of a set of easy-to-use formulas that characterize relevant temporal aspects, which must be considered by a temporal modeling of wireless communications.

Additionally, at the lowest levels of the system, the results of the theoretical model validated in this paper may be used in a message schedulability analysis that takes inaccessibility effects into account, thus securing the predictability and real-time support provided by IEEE 802.15.4 wireless communications. At a highest level, the occurrence of network inaccessibility events may influence the definition of Quality-of-Service (QoS) metrics, which evaluate the compliance of network communications with respect to the requirements of given applications. One major result of this study is that the influence of network errors, leading to periods of network inaccessibility cannot be ignored if predictability and real-time operation are system requirements. Otherwise, the timeliness and safety of the entire system may be at risk.

8 Conclusion and future directions

The issues addressed in this paper represent a first step to obtain predictability and real-time operation support out of wireless communication systems. In particular, the paper addressed the behavior of IEEE 802.15.4 networks in the presence of network errors, leading to periods of network inaccessibility. The results obtained by the work herein described has allowed to validate a previous theoretical model [12], consolidating its importance and confirming its correctness.

Relevant additions and modifications in the NS-2 simulator IEEE 802.15.4 module were presented allowing the simulation and evaluation of network inaccessibility scenarios. Based on this simulation the theoretical model was validated providing a fundamental source of information about relevant temporal aspects of the IEEE 802.15.4 beacon-enabled networks. Thus, the knowledge of a worst-case network inaccessibility time bound allows a better analysis and definition of a robust timeliness model, representing a first though crucial step for achieving an effective support to real-time operation in IEEE 802.15.4 networks.

Future research directions involve the reduction of network inaccessibility duration based on mechanisms present in the standard, the measurement and accountability of network inaccessibility considering different fault patterns (including fault patterns caused by malicious attacks), and the incorporation of the effects of network inaccessibility in the timeliness model of wireless communications. It may result in: improvements in the support of traffic and applications with hard temporal restrictions; definition of relevant real-time QoS metrics to evaluate the communication network, e.g verifying if the communication network is compliant with the level of requirements needed by given applications.

Acknowledgements

The authors give special thanks to Sweta Singh (in memorial), which contributed effectively in all technical aspects presented in the paper. Her dedication and knowledge will be always missed and remembered.

References

1. Ahmed, A.A., Fisal, N.: A real-time routing protocol with load distribution in wireless sensor networks. Computer Communications 31, 3190–3203 (September 2008)
2. Bartolomeu, P., Ferreira, J., Fonseca, J.: Enforcing flexibility in real-time wireless communications: A bandjacking enabled protocol. In: IEEE Conference on Emerging Technologies Factory Automation (ETFA). pp. 1 –4 (September 2009)
3. Cena, G., Valenzano, A., Vitturi, S.: Hybrid wired/wireless networks for real-time communications. IEEE Industrial Electronics Magazine 2(1), 8 –20 (March 2008)
4. Egea-López, E., Vales-Alonso, J., Martínez-Sala, A.S., García-Haro, J., Pavón-Mariño, P., Bueno Delgado, M.V.: A wireless sensor networks MAC protocol for real-time applications. Personal Ubiquitous Computing 12, 111–122 (January 2008)
5. Hameed, M., Trsek, H., Graeser, O., 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). pp. 1016 –1022 (September 2008)
6. Huang, Y.K., Pang, A.C., Hung, H.N.: An adaptive GTS allocation scheme for IEEE 802.15.4. IEEE Transactions on Parallel and Distributed Systems 19(5) (May 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. IEEE P802.15 Working Group (2011), Revision of IEEE Standard 802.15.4-2006
8. Issariyakul, T., Hossain, E.: Introduction to Network Simulator NS2. Springer (2009)
9. Koubâa, A., Cunha, A., Alves, M., Tovar, E.: i-GAME: An implicit GTS allocation mechanism in IEEE 802.15.4, theory and practice. Springer Real-Time Systems Journal 39(1-3), 169–204 (August 2008)
10. 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)

11. Sahoo, A., Baronia, P.: An energy efficient MAC in wireless sensor networks to provide delay guarantee. In: 15th IEEE Workshop on Local Metropolitan Area Networks (LANMAN). pp. 25 –30 (June 2007)
12. Souza, J.L.R., Rufino, J.: Characterization of inaccessibility in wireless networks-a case study on IEEE 802.15.4 standard. In: 3th IFIP International Embedded Systems Symposium(IESS). IFIP Advances in Information and Communication Technology, vol. 310. Langenargen, Germany (September 2009)
13. Veríssimo, P., Rufino, J., Rodrigues, L.: Enforcing Real-Time Behaviour on LAN-Based Protocols. In: 10th IFAC Workshop on Distributed Computer Control Systems (September 1991)
14. Wilson, W., Atkinson, G.: Wireless sensing opportunities for aerospace applications. Sensors & Transducers Journal 94, 83–90 (July 2008)
15. Xue, Y., Ramamurthy, B., Vuran, M.C.: SDRCS: A service-differentiated real-time communication scheme for event sensing in wireless sensor networks. Computer Networks 55(15), 3287 – 3302 (June 2011)
16. 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)
17. Zhuang, L., Goh, K., Zhang, J.: The wireless sensor networks for factory automation: Issues and challenges. pp. 141 –148 (September 2007)

A.1.4 Reducing Inaccessibility in IEEE 802.15.4 Wireless Communications

"Reducing Inaccessibility in IEEE 802.15.4 Wireless Communications". J. L. R. Souza and J. Rufino, (submitted for publication).

This page is intentionally left blank.

Reducing network inaccessibility in IEEE 802.15.4 wireless communications*

Jeferson L. R. Souza, and José Rufino

University of Lisbon - Faculty of Sciences
Large-Scale Informatics System Lab. (LaSIGE)
jsouza@lasige.di.fc.ul.pt, ruf@di.fc.ul.pt

Abstract Network inaccessibility is a threat that may compromise the timeliness and dependability of wireless communications, which is derived from a temporary "communication blackout" in the services supplied by the medium access control layer. During such temporary blackout the network is inaccessible, causing temporal hazards which difficult the provision and use of communication services with real-time guarantees. We define in this paper a set of policies to reduce network inaccessibility, using the IEEE 802.15.4 standard as a case study. Our reduction policies enhance the correctness of temporal analysis of IEEE 802.15.4 wireless communications face to the temporal restrictions imposed by sensor-based applications. Additionally, reducing network inaccessibility is a crucial step to enable the real use of such kind of communication technology in the provision of communication services with real-time guarantees.

Keywords: network inaccessibility, timeliness, dependability, wireless sensor networks, real-time.

1 Introduction

There is a demand for the use of wireless sensor networks (WSNs) on environments with temporal restrictions, where real-time communications are fundamental. This trend is guided by the need to reduce system size, weight, and power (SWaP) without lessen timeliness and dependability guarantees.

A lot of works have been presented, proposing new medium access control (MAC) protocols [13,1,5,3,15], modifications on the existent standards [8,7,10], and abstract models [11] trying to enhance the real-time guarantees and reliability of wireless communications. However, none of these works handled the central

* This work was partially motivated by our work within the scope of the ESA (European Space Agency) Innovation Triangle Initiative program, through ESTEC Project AIR-II (ARINC 653 in Space — Industrial Initiative), URL: http://air.di.fc.ul.pt. This work was partially supported by EC, through project IST-STREP-288195 (KARYON) and by FCT through the Multiannual Funding and CMU-Portugal Programs and the Individual Doctoral Grant SFRH/BD/45270/2008.

problem addressed by this paper: temporary "communication blackouts" caused by disturbances on the physical and MAC layers, which lead to the execution of additional procedures to reestablish normal MAC protocol operation. During the execution of such recovery procedures, the MAC layer although cannot be considered failed does not provide service, creating periods of network inaccessibility. When a network inaccessibility incident occurs communications cannot be performed. 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 the communication service.

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 a set of policies to reduce the duration of network inaccessibility within IEEE 802.15.4 [9] wireless communications.

To present our advances the paper is organized as follows: Section 2 presents an overview of the IEEE 802.15.4 standard. Section 3 presents briefly what is network inaccessibility and how network inaccessibility is characterized on IEEE 802.15.4 wireless networks. Section 5 explains the reduction policies defined to attenuate the duration of network inaccessibility, presenting analytical results which use the IEEE 802.15.4 standard as a case study. Finally, section 6 draws the conclusions, presenting insights to the future work.

2 IEEE 802.15.4 - Overview

The IEEE 802.15.4 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 coordinator node that manages and control the network access. The coordinator uses the superframe structure represented in Fig. 1 to control the access to the network. The duration of a superframe is calculated utilizing a constant that defines the minimum (also known as base) superframe duration, $\mathcal{T}_{BSD}$, and a beacon order exponent, BO, which is utilized 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, a superframe 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 optionally an inactive period (IP), where nodes may enter in a power-save

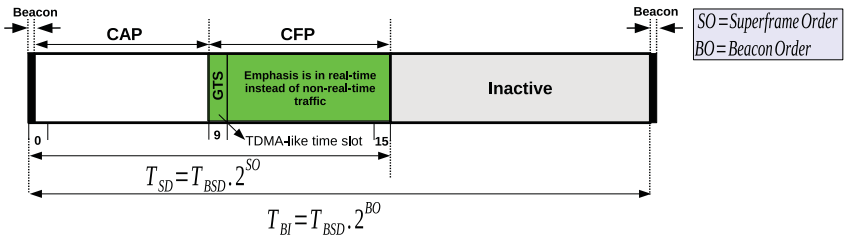


Figure 1: Superframe structure of the IEEE 802.15.4 in beacon-enabled mode

mode. A node may also allocate more than one contiguous time slot for exclusive and contention-free access.

The CAP and CFP together represents the active portion of the superframe structure, which has a duration given by:

$$\mathcal{T}_{SD} = \mathcal{T}_{BSD} \cdot 2^{SO} \tag{2}$$

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.

3 Network inaccessibility in IEEE 802.15.4 wireless communications

The concept of network inaccessibility was firstly introduced in 1989 by [17], in the context of local area networks (LANs). Network inaccessibility is characterized by the temporary absence of access to the network, being described in [17] as events with limited durations and rates, which a violation in one of those limits leads to a permanent failure of the network.

This section provides an overview of network inaccessibility in IEEE 802.15.4 wireless communications, presenting a comprehensive set of relevant network inaccessibility scenarios, being their worst case durations represented by the superscript (^{wc}). A more detailed characterization of network inaccessibility in IEEE 802.15.4 wireless communications can be found in [16].

The beacon frame controls the access to the network, and its reception is essential to maintain all the nodes synchronized within the different periods of the superframe structure. If a beacon frame is not correctly received an inaccessibility incident occurs. Thus, a **single beacon frame loss** occurs when only one beacon is lost, being the duration of such scenario given by:

$$\mathcal{T}_{ina \leftarrow sbfl} = \mathcal{T}_{BSD} . (2^{BO} + 1) \tag{3}$$

The value of $\mathcal{T}^{wc}_{ina \leftarrow sbfl}$ is equivalent to $\mathcal{T}_{BI}$ plus one $\mathcal{T}_{BSD}$ period, which is utilized as a margin to overcome some clock deviations that may occur between nodes. The **multiple beacon frame loss** occurs when multiple and consecutive beacons are lost. The duration of this scenario is given by:

$$\mathcal{T}^{wc}_{ina\leftarrow mbfl}=\mathcal{T}_{BSD}.\left(2^{BO}+1\right).nrLost \tag{4}$$

where a correct beacon frame is successfully received after the loss of $nrLost$ beacons. The **synchronization 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 this scenario results from the loss of synchronization among a node and its coordinator is represented by equation:

$$\mathcal{T}_{ina\leftarrow nosync}=\mathcal{T}_{BSD}.\left(2^{BO}+1\right).nrLost \tag{5}$$

To recover from such loss of synchronization two different strategies were identified in the standard specification [9]. Each individual node chooses the recovery strategy to be used. We assume that if some 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 the 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:

$$\begin{aligned}\mathcal{T}^{wc}_{ina\leftarrow orphan} = \mathcal{T}_{ina\leftarrow nosync} + \sum_{j=1}^{nrchannels} [\,\mathcal{T}^{wc}_{MAC}(Orphan) + nrWait\;.\;\mathcal{T}_{BSD}\,] + \\ \mathcal{T}_{MLA}(Realign) + \mathcal{T}^{wc}_{MAC\_ack}(Realign)\end{aligned} \tag{6}$$

where: $nrchannels$, represents the number of channels to be scanned; $nrWait$, defines the waiting period for a beacon frame in each channel scan, assuming the default value of $nrWait=32$ in the IEEE 802.15.4 standard; $\mathcal{T}_{MAC\_ack}(frame)$ and $\mathcal{T}_{MAC}(frame)$ represent the delay from request to confirmation of a MAC frame transmission time with and without acknowledgement, respectively; 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 execution of the **re-association** procedure, the channel scan is followed by a beacon processing action, an association procedure and the extract of control information. The period of network inaccessibility is given by:

$$\begin{aligned}\mathcal{T}^{wc}_{ina\leftarrow reAssoc} = \mathcal{T}_{ina\leftarrow nosync} + \sum_{j=1}^{nrchannels} [\mathcal{T}^{wc}_{MAC}(Beacon\_R) + nrWait.\mathcal{T}_{BSD}] + \\ \mathcal{T}_{MLA}(Beacon) + \mathcal{T}^{wc}_{MAC\_ack}(Assoc\_R) + \\ \mathcal{T}_{MLA}(Assoc) + \mathcal{T}^{wc}_{MAC\_ack}(Ext\_R)\end{aligned} \tag{7}$$

[1] A logical channel is a numerical representation of a radio frequency (RF) channel utilized by the MAC layer to perform its network communications.

Finally, a **coordinator conflict** occurs when more than one coordinator are active within the same network. By default, each network has an unique identifier, $networkID$, which identifies the network uniquely and is used by the coordinator in beacon transmissions. If some other (possibly old) coordinator enters the network operation space, e.g., after moving away during a long period of time, the network may have two different coordinators transmitting beacons with the same $networkID$. To solve such conflict, the actual coordinator performs a search within a set of specified logical channels. If the coordinator does not found other coordinator sending beacons with its own identifier after the scan in all logical channels, no further action is taken and the network becomes accessible again. Otherwise, a new identifier is selected and, if necessary, a MAC coordinator realignment command is broadcast. The corresponding period of network inaccessibility has a worst case duration given by:

$$\mathcal{T}^{wc}_{ina \leftarrow Conflict} = \mathcal{T}_{MLA}(Conflict) + \sum_{j=1}^{nrchannels} [\,\mathcal{T}^{wc}_{MAC}(Beacon\_R) + nrWait.\mathcal{T}_{BSD}\,] + \mathcal{T}_{MLA}(Realign) + \mathcal{T}^{wc}_{MAC}(Realign) \tag{8}$$

4 Network parametrization for real-time operation

Network parametrization is the first and crucial step to prepare the network platform for real-time operation, consisting in the fine-adjustment of a relevant set of network configuration parameters, which improves the network real-time characteristics and support. This is formalized by the following proposition:

Proposition 1. *Each node accesses the network in a bounded and known time interval of, at most,* $\mathcal{T}_{ac}$.

This proposition represents the real-time capabilities of a network and the characterization of $\mathcal{T}_{ac}$ is dependent from the network characteristics, which implies in different temporal guarantees being offered. The value of $\mathcal{T}_{ac}$ simply accounts for the raw network access delay observed at MAC layer before starting a frame transmission; it does not include any buffering/queueing effects, frame transmission times and delays associated with possible frame retransmissions in the presence of omission errors. For the particular case of IEEE 802.15.4 operating in beacon-enabled mode, a bounded and known $\mathcal{T}_{ac}$ is secured given that contention-free access within GTS is provided based on a periodic time interval equal to $\mathcal{T}_{BI}$. Therefore:

$$\mathcal{T}_{ac} = \mathcal{T}_{BI} \tag{9}$$

A decision concerning the real value of $\mathcal{T}_{BI}$ is dependent of the temporal requirements of applications, and is set in function of BO, as shown in Table 1.

Table 1: Different configurations utilized by an IEEE 802.15.4 network, using the $2.4\,GHz$ frequency band timing.

Beacon order (BO)	**Beacon interval** $\mathcal{T}_{BI}$ (ms)	**Superframe order** (SO)	**Duty Cycle** (%)	**Time slot duration** (ms)
3	123	3	100	7.68
		2	50	3.84
		1	25	1.92
		0	12.5	0.96
2	61	2	100	3.84
		1	50	1.92
		0	25	0.96
1	31	1	100	1.92
		0	50	0.96
0	15	0	100	0.96

Table 2: IEEE 802.15.4 frame durations, using the $2.4\,GHz$ frequency band

Frame type	**Symbol**	**Length** (bit)	**Duration** (ms)
Data frames			
Data (Minimum payload)	$\mathcal{T}_{data}^{bc}$	8	0.03
Data (Maximum payload)	$\mathcal{T}_{data}^{wc}$	1016	4.07
Data request	$\mathcal{T}_{Ext\_R}$	320	1.28
Data acknowledgment[1]	$\mathcal{T}_{ack}$	40	1.00
MAC control frames			
Beacon	$\mathcal{T}_{Beacon}$	1016	4.07
Beacon request	$\mathcal{T}_{Beacon\_R}$	64	0.26
Network ID conflict notification	$\mathcal{T}_{Conflict}$	304	1.22
Orphan notification	$\mathcal{T}_{Orphan}$	128	0.52
Realign	$\mathcal{T}_{Realign}$	280	1.12
Association request	$\mathcal{T}_{Assoc\_R}$	312	1.25
GTS request	$\mathcal{T}_{GTS\_R}$	72	0.29
Control request	$\mathcal{T}_{Ext\_R}$	320	1.28
MAC frame acknowledgment[1]	$\mathcal{T}_{ack}$	40	1.00

The value of SO defines the duty cycle and the duration of each time slot. Table 2 shows the durations of the different IEEE 802.15.4 data/control frames.

For the remainder of our analyzes we use as an example $\mathcal{T}_{BI} = 123ms$, which can provide a reasonable beacon interval for periodic real-time transmissions, still allowing the use of reliable unicast data transmissions for the longer duty cycles. The (real) value of $\mathcal{T}_{BI}$ is also utilized to normalize the duration of the network the duration of network inaccessibility events, as shown in Fig. 2.

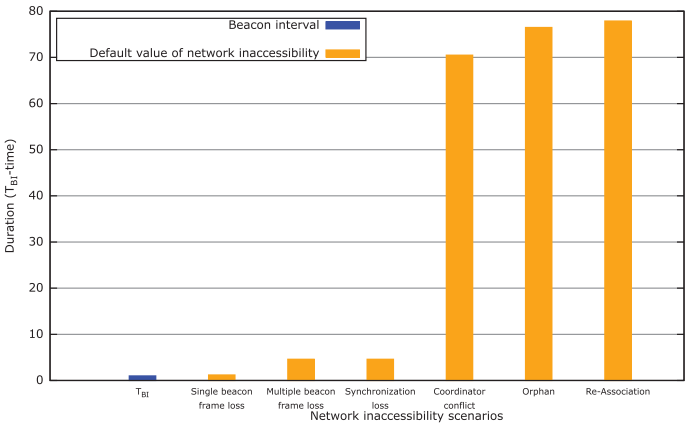


Figure 2: The IEEE 802.15.4 inaccessibility durations normalized by, and compared with, $\mathcal{T}_{BI}$ ($\mathcal{T}_{BI} = 123ms$).

5 Reducing network inaccessibility in IEEE 802.15.4 wireless communications

To figure out the impact of network inaccessibility Fig. 2 presents a comparison among the duration of inaccessibility incidents and the beacon interval, $\mathcal{T}_{BI}$. All values present in Fig. 2 were normalized by $\mathcal{T}_{BI}$. A value of $\mathcal{T}_{BI} = 123ms$ is used for normalization, being represented in Fig. 2 with the duration of one unit of time. Network inaccessibility incidents have very different durations, with some of them much longer than $\mathcal{T}_{BI}$. Long and highly variable inaccessibility periods are a source of unpredictability in network operation, evidencing the necessity and importance to reduce the duration of inaccessibility incidents.

Thus, we define a set of policies designed to reduce network inaccessibility in IEEE 802.15.4 wireless communications, essential to enhance communication properties such as dependability, timeliness and predictability.

5.1 Coordinator conflict avoidance policy

This policy was designed to avoid the occurrence of the *coordinator conflict* network inaccessibility scenario. The *coordinator conflict* scenario occurs when two or more coordinators transmit beacons with the same (unique) network identifier, $networkID$. When this situation is detected, the IEEE 802.15.4 standard specifies that a conflict resolution strategy should be triggered, causing then the occurrence of network inaccessibility. Our coordinator conflict avoidance policy defines a simple and effective strategy to avoid the coordinator conflict using a

[1] The worst case duration, for the wait of an acknowledgement frame, follows the IEEE 802.15.4 standard specification [9].

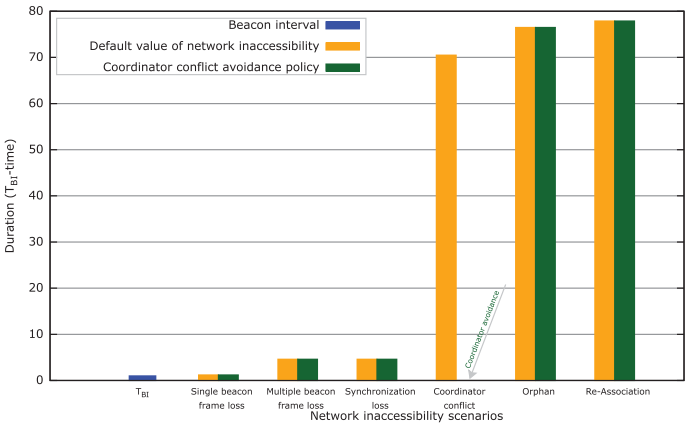


Figure 3: Impact of the coordinator conflict avoidance policy in the IEEE 802.15.4 network inaccessibility

compound network identifier, which consists in the use of a 2-Tuple with the actual identifier, $networkID$, and the address of the actual coordinator represented here by $coordinatorID$. The use of such compound identifier establishes the following proposition:

Policy proposition. Each node must use a 2-Tuple $\langle networkID, coordinatorID \rangle$ as a unique network identifier, avoiding then a coordinator conflict.

The result of the aforementioned proposition is applied as an extension of the IEEE 802.15.4, which adds the following check procedure applied to each locally 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.*

The beacons transmitted by coordinators different from the actual one are considered invalided at the exposed MAC interface, and are not delivered at the MAC management entities, therefore do not originate the necessity to solve a coordinator conflict problem. An indication of a coordinator conflict in coordinators different from the actual one can still be delivered to MAC management entities to allow such nodes to orderly join the network without producing inaccessibility incidents. This extension to IEEE 802.15.4 operation can be made compatible with the standard specification and is not hard to implement in modern wireless communication platforms [2]. The effectiveness of our policy, which eliminates the existence of the coordinator conflict scenario, is shown in Fig. 3.

In this paper we do not address the presence of malicious entities within the network, which may cause an intentional coordinator conflict problem. Malicious entities need to be handled with additional techniques to overcome the hazards that they may cause, being addressed in our future work.

5.2 Channel utilization awareness policy

To contribute to the reduction of other IEEE 802.15.4 network inaccessibility scenarios, namely the *orphan* and *re-association* scenarios, we design the channel utilization awareness policy, which consists in the use of the channel utilization knowledge by the network coordinator to reduce the time spent in channel scan operations. In the channel utilization awareness policy each node is "aware" of the number of logical channels which are available to use by the coordinator, being represented by the following proposition:

Policy proposition. Each node is aware of the channel utilization by the network coordinator, restricting the search for this coordinator in some $C_a := \{c \mid c \in C \wedge C \subset A\}$, where C_a is the proper search set and A is the set of the available logical channels, being $0 < \#C_a < \#A$.

where the set C is a general representation of a proper subset of A, and is utilized to formalize the class of such proper subsets. The coordinator uses a subset, C_a, of the available logical channels set, A, to delimit its operational channel scope. Each node is able to search and find the coordinator within this channel scope, reducing then the amount of time needed to search and find the current logical channel where the coordinator is in operation, which also reduces the duration of some network inaccessibility scenarios.

Our channel availability awareness policy assumes only $\#C_a$ logical channels are available to use, instead of the initial $\#A = 16$ available channels. The subset C_a of logical channels should be mapped into a subset of non-overlapping radio channels in order to minimize radio signal interference. For the $2.4Ghz$ frequency band, one can define a subset of four non-overlapping radio channels and therefore $\#C_a = 4$.

Figure 4 presents the impact of our channel utilization policy in the IEEE 802.15.4 network inaccessibility and evidences its effectiveness in reducing the duration of the longer network inaccessibility scenarios.

5.3 Network dependability awareness policy

An omission is an error that destroys a data or control frame. Wireless communication channels are especially susceptible to frame omissions, which may be due to a number of causes: electromagnetic interference in the medium; disturbances in a node transmitter/receiver circuitry; collisions derived from hidden nodes or originated by node mobility.

Figure 5 summarizes in three main classes how frame omissions can be detected. Different collision detection techniques have been proposed for wireless communications in the past years [12,14]. Such techniques require additional hardware, which is also susceptible to disturbances on the communication medium, being difficult to secure their effectiveness. Cyclic redundancy check (CRC) have an error detection coverage appropriate to accidental errors [6]. Frame-driven techniques use a timeout associated to a specific frame to monitor the success of frame transfer. In addition, timeouts can also be associated to

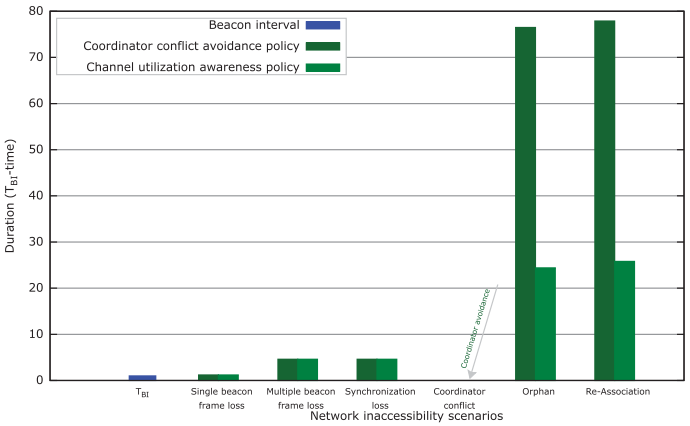


Figure 4: Impact of the channel utilization awareness policy in the IEEE 802.15.4 network inaccessibility ($\#C_a = 4$)

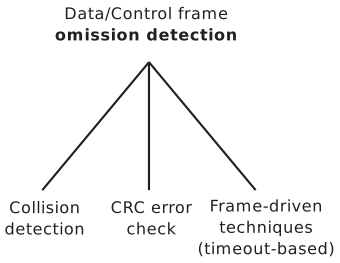


Figure 5: Mechanisms to detect omissions in communication networks

specific frame reception events to monitor network activity. One example is the monitoring of beacon frames performed by the IEEE 802.15.4 MAC protocol. Thus, CRC and frame-driven (timeout-based) techniques are the safe ways to detect omissions on wireless communications.

The first mechanism we introduce towards the improvement of network dependability awareness extends the semantic of the classical CRC error check procedure. In a classical CRC error check procedure when a frame is received with a CRC error, it is discarded and nothing is signalized. The extension we introduce in Algorithm 1 is able to notify MAC management entities when a data or control frame is received with CRC errors.

In the pseudo-code of Algorithm 1 the variable *crc_error_status* is utilized to represent the status of a CRC error check performed on a received frame. The difference from the classical CRC error check (lines 6 and 14) consists in extracting the $frame\_header$ and notify the corresponding *crc_error_status* to MAC management entities (line 14). Such indication allows a better knowledge

Algorithm 1 Extending frame CRC error check

```
1: Initialization phase.
2: crc_error_status ←false;
3: Begin.
4: loop
5:    when Channel.indication(frame) do
6:       frame_header ←MAC.get.header(frame);
7:       if MAC.CRC.check(frame) is OK then
8:          crc_error_status ← false;
9:          MAC.indication(frame);
10:      else
11:         crc_error_status ← true;
12:         MAC.frame.discard(frame);
13:      end if
14:      MAC.Mgmt.indication(time_slot, frame_header, crc_error_status);
15:   end when
16: end loop
17: End.
```

of the status of the communication channel, being directly utilized by our complementary mechanisms to monitor and evaluate the network status properly. Additionally, the indication generated in line 14 also reports in which *time_slot* the frame was received, which can be utilized to identify if the sender information present within the $frame\_header$ matches the owner of the *time_slot*, in case of transmissions within CFP. The implementation of this extension to the classical functionality is not hard to implement off-the-shelf using modern commercially available wireless communication platforms [2].

This means that we now have a simple mechanism allowing to accurately detect and account for the omission degree of a logical channel, O_d, defined as the number of consecutive logical channel omissions. Considering only accidental transient faults, the omission degree of 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 of the network error characteristics and environment conditions [4]. The IEEE 802.15.4 standard indirectly defines a fixed value of $k = 3$ in its error handling mechanisms. Having the ability to determine an adequate omission degree bound is a worthwhile feature.

Therefore, making use of the indication provided by the CRC error check procedure, the pseudo-code specified in Algorithm 2 accounts for the locally-observed logical channel omission degree, which is represented by O_d in line 2. The reception of a frame with a correct CRC (lines 9 and 10) indicates that the current logical channel is correct. However, when a frame with a CRC error is received the procedure increments O_d (line 8). If the O_d value exceeds k, the violation of the assumed omission degree bound is signalled to MAC management entities (line 14). This may be an indication of an heavily disturbed channel or

Algorithm 2 Omission degree monitoring

```
1: Initialization phase.
2: O_d ← 0;
3: k ← 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 in-
   directly defines k ← 3;
4: Begin.
5: loop
6:    when MAC.Mgmt.indication(time_slot, frame_header, crc_error_status) do
7:       if crc_error_status is true then
8:          O_d ← O_d + 1;
9:       else if crc_error_status is false then
10:         O_d ← 0;
11:      end if
12:   end when
13:   if O_d > k then
14:      MLA.Mgmt.indication(logical_channel, O_d_exceeds_k);
15:   end if
16: end loop
17: End.
```

it may be a result of the underestimation of the omission degree bound. In any case, the logical channel should be considered failed.

Our network dependability awareness policy can now be formulated by the following proposition:

Policy proposition. 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 3, 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 inaccessibility times associated to the *multiple beacon frame loss* and *synchronization loss* scenarios.

5.4 Logical channel diversity policy

The violation of the logical channel omission degree bound locally-perceived by each node should be interpreted as a failure indication. To restore communication one must resort to the two following propositions:

Table 3: Network parametrization in function of dependability metrics

IEEE 802.15.4 Parameter	**IEEE 802.15.4 Standard Configuration**	**Dependable Adaptation**
$nrLost$	4	$k+1$
$nrWait$	32	$(k+1).2^{BO}$

Policy proposition. There are multiple and redundant logical channels, where a frame is only transmitted in the active one.

Policy proposition. 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 in the use of redundant logical channels, specifying the following procedure: *when a node (including the coordinator) detects that a given logical channel O_d exceeds k a node switches to the next logical channel*. To avoid the occurrence of a permanent physical partitioning of the network, the same channel configuration is utilized by all nodes, defining a deterministic order utilized to switch from one logical channel to another one. Furthermore, we assume: *each node must transmit at least one (heartbeat) frame during its allocated GTS to signal node liveness*.

Upon logical channel switch it may happen that a node detects no traffic activity because it is the only node in that channel. The standard MAC protocol of non coordinator nodes has mechanisms to detect such situations, signaled to MAC management entities through a *synchronization loss* indication. However, the MAC protocol of the coordinator must be enhanced to detect logical channel idleness, as specified in Algorithm 3: a superframe is delimited by two consecutive beacons, and when the MAC layer does not indicate any traffic within the superframe (line 13), an idle period is signaled (line 9). Algorithm 3 also specifies the switch operation to the next logic channel (line 17) upon receiving an indication that the current channel has failed or is idle (line 16). In case of a channel switch performed by the coordinator the number of associated nodes becomes 0 (line 18), avoiding a wrong idleness detection within the new channel.

Algorithm 4, to be executed at non coordinator nodes controls how logical channel switch is performed: no channel switch is due if no indication of channel failure has been received and some beacon (either correct or incorrect) have been heard, which means the node has become *orphan*. A non failed channel is able to deliver correct beacon frames, which implies that *orphan* procedure is executed within only one channel (line 13), which is the current one. It results in a quick synchronization reestablishment between the coordinator and a non coordinator node, even if compared with our channel utilization awareness policy.

Otherwise, the node should switch channel and execute a *re-association* procedure (line 18). The *re-association* procedure is performed within only two

Algorithm 3 Logical channel diversity procedure - COORDINATOR

```
1: Initialization phase.
2: nrAssocNodes ← 0;
3: idle_status ← false;
4: Begin.
5: loop
6:    when MAC.Mgmt.request(Beacon) do
7:       nrAssocNodes ← MLA.Mgmt.get(NR_ASSOC_NODES);
8:       if idle_status is true ∧ nrAssocNodes > 0 then
9:          MLA.Mgmt.indication(logical_channel, idle_status);
10:      end if
11:      idle_status ← true;
12:   end when
13:   when MAC.indication(frame) do
14:      idle_status ← false;
15:   end when;
16:   when MLA.Mgmt.indication(logical_channel, O_d_exceeds_k) ∨
      MLA.Mgmt.indication(logical_channel, idle_status) do
17:      MLA.Mgmt.request(Change_Channel);
18:      MLA.Mgmt.request(RESET_NR_ASSOC_NODES);
19:   end when
20: end loop
21: End.
```

logical channels, the new channel and the previous one. The use of only two channels is justified by: (a) the coordinator remains in the non failed previous channel with other non coordinator nodes; or (b) the coordinator detects an idle period and switches to the new channel before a non coordinator node finishes the execution of the *re-association* procedure in that new channel. It is important to emphasize that the execution of the *re-association* procedure within a logical channel is k times greater than the detection of a logical channel idleness.

Figure 6 shows the impact of the logical channel diversity policy in the reduction of network inaccessibility of IEEE 802.15.4 wireless communications.

6 Conclusion and future work

This paper presented a set of policies to reduce the negative effects of network inaccessibility within IEEE 802.15.4 wireless communications. Our results shows the efficiency of the approach proposed by our policies, which enhances the MAC layer operation extending the IEEE 802.15.4 standard with advanced omission detection and signalization mechanisms. Such mechanisms improve the timeliness and dependability properties of communications, being an important step to enhance the real-time communication support of IEEE 802.15.4 WSNs.

Future research directions of this work includes the study of new techniques, which exploit multiple communication channels to enhance the reliability of communications; the study of network inaccessibility in the presence of malicious at-

Algorithm 4 Logical channel diversity procedure - NON COORDINATOR

1: Initialization phase.
2: $changeState \leftarrow false$;
3: $receivedBeacons \leftarrow false$;
4: Begin.
5: **loop**
6: **when** *MLA.Mgmt.indication(logical_channel, O_d_exceeds_k)* **do**
7: *MLA.Mgmt.request(Change_Channel)*;
8: $changeState \leftarrow true$;
9: **end when**
10: **when** *MAC.Mgmt.indication(SYNC_LOSS)* **do**
11: $receivedBeacon \leftarrow$ *MLA.Mgmt.get(RECEIVED_BEACON)*;
12: **if** *changeState is false* $\wedge$ *receivedBeacon is true* **then**
13: *MLA.Mgmt.request(ORPHAN,*$\#C_a = 1$*)*
14: **else**
15: **if** *changeState is false* **then**
16: *MLA.Mgmt.request(Change_Channel)*;
17: **end if**
18: *MLA.Mgmt.request(RE_ASSOCIATION,*$\#C_a = 2$*)*;
19: $changeState \leftarrow false$;
20: **end if**
21: **end when**
22: **end loop**
23: End.

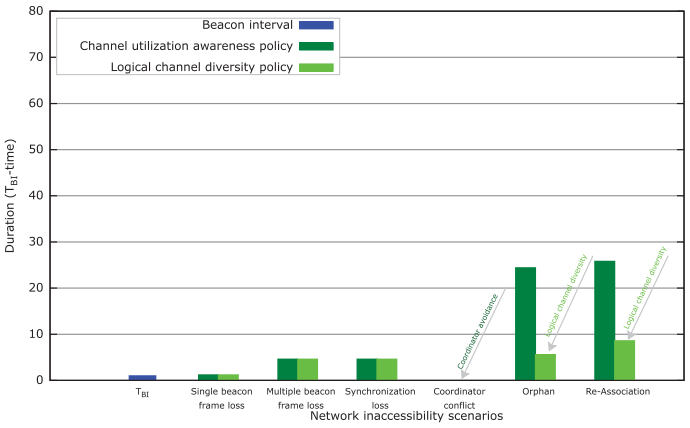


Figure 6: Impact of the logical channel diversity policy in the IEEE 802.15.4 network inaccessibility ($\#C_a = 1$ and $\#C_a = 2$ for the *orphan* and *re-association* scenario, respectively).

tacks; the incorporation of the effects of network inaccessibility in the timeliness model of wireless communications.

References

1. Aad, I., Hofmann, P., Loyola, L., Riaz, F., Widmer, J.: E-MAC: Self-organizing 802.11-compatible MAC with elastic real-time scheduling. In: IEEE Internatonal Conference on Mobile Ad hoc and Sensor Systems (MASS).(October 2007)
2. ATMEL: ATMEL AVR2025: IEEE 802.15.4 MAC Software Package - User guide. ATMEL Coorporation (May 2012).
3. Bartolomeu, P., Ferreira, J., Fonseca, J.: Enforcing flexibility in real-time wireless communications: A bandjacking enabled protocol. In: IEEE Conference on Emerging Technologies Factory Automation (ETFA). pp. 1 –4 (September 2009)
4. Eckhardt, D., Steenkiste, P.: Measurement and analysis of the error characteristics of an in-building wireless network. In: SIGCOMM '96: Conf. Proc. on Applications, Tech., Arch. and Protocols for Computer Comm. New York, NY, USA (1996)
5. Egea-López, E., Vales-Alonso, J., Martínez-Sala, A.S., García-Haro, J., Pavón-Mariño, P., Bueno Delgado, M.V.: A wireless sensor networks MAC protocol for real-time applications. Personal Ubiquitous Computing 12, 111–122 (January 2008)
6. Fujiwara, T., Kasami, T., Lin, S.: Error detecting capabilities of the shortened hamming codes adopted for error detection in IEEE standard 802.3. IEEE Transactions on Communications 37(9), 986–989 (Sep 1989)
7. Hameed, M., Trsek, H., Graeser, O., 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). pp. 1016 –1022 (September 2008)
8. Huang, Y.K., Pang, A.C., Hung, H.N.: An adaptive GTS allocation scheme for IEEE 802.15.4. IEEE Trans. on Parallel and Distributed Systems 19(5) (May 2008)
9. 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
10. Koubâa, A., Cunha, A., Alves, M., Tovar, E.: i-GAME: An implicit GTS allocation mechanism in IEEE 802.15.4, theory and practice. Springer Real-Time Systems Journal 39(1-3), 169–204 (August 2008)
11. Kuhn, F., Lynch, N., Newport, C.: The abstract MAC layer. In: 23rd International Symposium On Distributed Computing (DISC). Spain (September 2009)
12. Peng, J., Cheng, L., Sikdar, B.: A wireless MAC protocol with collision detection. IEEE Transactions on Mobile Computing 6(12), 1357 –1369 (December 2007)
13. Sahoo, A., Baronia, P.: An energy efficient MAC in wireless sensor networks to provide delay guarantee. In: 15th IEEE Workshop on Local Metropolitan Area Networks (LANMAN). pp. 25 –30 (June 2007)
14. Sen, S., Roy Choudhury, R., Nelakuditi, S.: CSMA/CN: Carrier sense multiple access with collision notification. In: Sixteenth International Conference on Mobile Computing and Networking (MOBICOM). pp. 25–36. MobiCom '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1859995.1859999
15. Shuai, X.Y., Zhang, Z.C.: Research of real-time wireless networks control system MAC protocol. Journal of Networks 5(4), 419–426 (April 2010)
16. From the same authors: Characterization of inaccessibility in wireless networks-a case study on IEEE 802.15.4 standard.
17. Veríssimo, P., Rodrigues, L., Baptista, M.: AMp: A Highly Parallel Atomic Multicast Protocol. SIGCOMM Comput. Commun. Rev. 19(4), 83–93 (1989)

A.1.5 Self-Stabilizing TDMA algorithms for Dynamic Wireless Ad-hoc Networks

"Self-Stabilizing TDMA algorithms for Dynamic Wireless Ad-hoc Networks". Pierre Leone and Elad Michael Schiller. The 8th International Symposium on Algorithms for Sensor Systems, Wireless Ad Hoc Networks and Autonomous Mobile, 2012.

This page is intentionally left blank.

Self-Stabilizing TDMA Algorithms for Dynamic Wireless Ad-hoc Networks *

Pierre Leone † Elad M. Schiller ‡

October 12, 2012

Abstract

In dynamic wireless ad-hoc networks (DynWANs), autonomous computing devices set up a network for the communication needs of the moment. These networks require the implementation of a medium access control (MAC) layer. We consider MAC protocols for DynWANs that need to be autonomous and robust as well as have high bandwidth utilization, high predictability degree of bandwidth allocation, and low communication delay in the presence of frequent topological changes to the communication network. Recent studies have shown that existing implementations cannot guarantee the necessary satisfaction of these timing requirements. We propose a self-stabilizing MAC algorithm for DynWANs that guarantees a short convergence period, and by that, it can facilitate the satisfaction of severe timing requirements, such as the above. Besides the contribution in the algorithmic front of research, we expect that our proposal can enable quicker adoption by practitioners and faster deployment of DynWANs, such as the `IEEE 802.11p` for mobile ad hoc networks (MANETs) and vehicular ad-hoc network (VANETs).

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

†Computer Science Department, University of Geneva, Geneva Switzerland. Email: pierre.leone@unige.ch

‡Chalmers University of Technology, Göteborg Sweden. Email: elad.schiller@chalmers.se

1 Introduction

Dynamic wireless ad-hoc networks (DynWANs) are autonomous and self-organizing systems where computing devices require networking applications when a fixed network infrastructure is not available or not preferred to be used. In these cases, computing devices may set up a short-lived network for the communication needs of the moment, also known as, an ad-hoc network. Ad-hoc networks are based on wireless communications that require implementation of a *Medium Access Control* (MAC) layer. We consider MAC protocols for DynWANs that need to be autonomous, robust, and have high bandwidth utilization, a high predictability degree of bandwidth allocation, and low communication delay [22] in the presence of frequent changes to the communication network topology. Existing implementations cannot guarantee the necessary satisfaction of timing requirements [6, 7]. This work proposes an algorithmic design for self-stabilizing MAC protocols that guarantees a short convergence period, and by that, can facilitate the satisfaction of severe timing requirements. The proposed algorithm possesses a greater degree of predictability, while maintaining low communication delays and high throughput.

The dynamic and difficult-to-predict nature of wireless ad-hoc networks gives rise to many fault-tolerance issues and requires efficient solutions. DynWANs, for example, are subject to transient faults due to hardware/software temporal malfunctions or short-lived violations of the assumed settings for modeling the location of the mobile nodes. Fault tolerant systems that are *self-stabilizing* [16] can 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), or the model of dynamic networks in which communication links and nodes may fail and recover during normal operation [17]. The proof of self-stabilization requires convergence from an arbitrary starting system state. Moreover, once the system has converged and followed its specifications, it is required to do so forever. The self-stabilization design criteria liberate the application designer from dealing with low-level complications, such as bandwidth allocation in the presence of topology changes, and provide an important level of abstraction. Consequently, the application design can easily focus on its task – and knowledge-driven aspects.

The `IEEE 802.11` standard is widely used for wireless communications. Nonetheless, the research field of MAC protocols is very active and requires further investigation. In fact, the `IEEE 802.11` amendment, `IEEE 802.11p`, for wireless access in vehicular environments (WAVE), has just being published. It was shown that the standard's existing implementations cannot guarantee channel access before a finite deadline [6, 7]. Therefore, applications with severe timing requirements cannot predictably meet their deadlines, e.g., safety-critical applications for vehicular systems.

ALOHAnet and its synchronized version Slotted ALOHA [1] are pioneering wireless systems that employ a strategy of "random access". Time division multiple access (TDMA) [41] is another early approach, where nodes transmit one after the other, each using its own timeslot, say, according to a defined schedule. Radio transmission analysis in ad-hoc networks [20] and relocation analysis of mobile nodes [34] show that there are scenarios in which MAC algorithms that employ a scheduled access strategy have lower throughput than algorithms that follow the random access strategy. However, the scheduled approach offers greater predictability of bandwidth allocation and communication delay, which can facilitate fairness [23] and energy conservation [50].

Our design choices have basic radio technology in mind, whilst aiming at satisfying applications that have severe timing requirements. We consider TDMA frames with fixed number of fixed length timeslots. The design choice of TDMA frames with fixed-length radio time fits well applications that have severe delay requirements. By avoiding the division of fixed length frames into timeslots of non-equal length, as in [23, 10], we take into consideration the specifications of basic radio technology.

In the context of the above design choices, there are two well-known approaches for dealing with contention (timeslot exhaustion): (1) employing policies for administering message priority (for meeting timing requirements while maintaining high bandwidth utilization, such as [39]), or (2) adjusting the nodes' individual transmission signal strength or carrier sense threshold [43]. The former approach is widely accepted and adopted by the IEEE `802.11p` standard, whereas the latter has only been evaluated via computer simulations [43].

The proposed algorithm facilitates the implementation of both of the above approaches (more details appear in Section 7). For the sake of presentation simplicity, we consider a single priority MAC protocol and base the timeslot allocation on straightforward vertex-coloring. The proposed algorithm allocates timeslots to a number of nearby transmitters, i.e., a number that is bounded by the TDMA frame size, whereas other nearby transmitters receive busy channel indications. The analysis considers saturated situations in which the node degree in the message collision graph is smaller than the TDMA frame size. As explained above, this analysis assumption does not restrict the number of concurrent transmitters when implementing the proposed MAC algorithm.

Related work We are not the first to propose a MAC algorithm for DynWANs that follows the TDMA's scheduled approach. STDMA [51] and Viqar and Welch [48] consider GNSS-based scheduling (Global Navigation Satellite System [44]) according to the nodes' geographical position and their trajectories. Autonomous systems cannot depend on GNSS services, because they are not always available, or preferred not to be used, due to their cost. Arbitrarily long failure of signal loss can occur in underground parking lots and road tunnels. We propose a self-stabilizing TDMA algorithm that does not require GNSS accessibility or knowledge about the node trajectories. Rather it considers an underlying self-stabilizing local pulse synchronization, such as [14, 38], which can be used for TDMA alignment, details appear in [38].

When using collision-detection at the receiving-side [43, 11, 51, 47, 31], it is up to the receiving-side to notify the sender about collisions via another round of collision-prone transmissions, and by using FI (frame information) payload fields that includes T entries, where T is the TDMA frame size. Thus far, FI-based protocols study the stochastic resolution of message collision via computer network simulations [51, 2, 45, 12, 47, 31]. Simulations are also used for evaluating the heuristics of MS-ALOHA [43] for dealing with contention (timeslot exhaustion) by adjusting the nodes' individual transmission signal strength and / or carrier sense threshold. We do not consider lengthy frame information (FI) fields, which significantly increase the control information overhead, and yet we provide provable guarantee regarding the convergence time. Further analysis validation of the proposed algorithm via simulations and test bed implementation can be found in Section 8, and respectively, in [38].

The proposed algorithm does *not* consider collision-detection mechanisms that are based on signal processing or hardware support, as in [15]. Rather, it employs a variation on a well-known strategy for eventually avoiding concurrent transmissions among neighbors. This strategy allows the sending-side to eventually observe the existence of interfering transmissions. Before sending, the sender waits for a random duration while performing a clear channel assessment. A channel is considered to be used once the detected energy levels reach a threshold in which the radio unit is expected to succeed in carrier sense locking (details appear in Section 3).

The proposed MAC algorithm can be entirely based on the carrier sensing of message transmission, as in [10], which focuses on fair bandwidth allocation, but does not consider dynamic networks or self-stabilization. Our algorithm uses carrier sensing of message transmission and the above collision avoidance strategy for coloring vertices, i.e., single-hop-distance broadcasting. It can facilitate unicast functionally with no significant overheads. Two-hop-distance vertex coloring is often used by unicast MAC protocols for mitigating hidden terminal phenomena. Naturally, the proposed algorithm facilitates the implementation of

a self-stabilizing two-hop-distance vertex coloring, such as [8].

An abstract MAC layer was specified for DynWANs in [28]. The authors mention algorithms that can satisfy their specifications. However, they do not consider predictable broadcasting schedules.

Local algorithms [21, 19] considers both theoretical and practical aspects of MAC algorithms [**?**,]and references therein]Wattenhofer2010Theory and the related problem of clock synchronization, see [32] and references therein. For example, the first partly-asynchronous self-organizing local algorithm for vertex-coloring in wireless ad-hoc networks is presented in [42]. However, this line currently does not consider dynamic networks and predictable bandwidth allocation.

Two examples of self-stabilizing TDMA algorithms are presented in [23, 27]. The algorithms are based on vertex-coloring and consider (non-dynamic) ad-hoc networks. Recomputation and floating output techniques ([16], Section 2.8) are used for converting deterministic local algorithms to self-stabilization in [33]. The authors focus on problems that are related to MAC algorithms. However, deterministic MAC algorithms are known to be inefficient in their bandwidth allocation when the topology of the communication network can change frequently [34]. There are several other proposals related to self-stabilizing MAC algorithms for sensor networks, e.g., [29, 5, 4, 30]; however, none of them consider dynamic networks and their frame control information is quite extensive.

The MAC algorithms in [34, 36, 35, 38] *have no proof* that they are self-stabilizing. The authors of [34] present a MAC algorithm that uses convergence from a random starting state (inspired by self-stabilization). In [36, 35, 38], the authors use computer network simulators for evaluating self-$\star$ MAC algorithms.

Our contribution This work proposes a self-stabilizing MAC algorithm that demonstrates rapid convergence without the extensive use of frame control information. Our analysis shows that the algorithm facilitates the satisfaction of severe timing requirements for DynWANs.

We start by considering transient faults and topological changes to the communication network, i.e., demonstrating self-stabilization in Theorem 4.2. We then turn to focus on bounding the algorithm's convergence time after an arbitrary and unbounded finite sequence of transient faults and changes to the network topology. Theorem 5.1 shows that the expected local convergence time is brief, and bounds it in equation (7). Theorem 6.2 formulates the expected global convergence time in equation (21). Moreover, for a given probability, the global convergence time is calculated in equation (22).

For discussion (Section 8), we point out the algorithm's ability to facilitate the satisfaction of severe timing requirements for DynWANs. Moreover, the analysis conclusions explain that when allowing merely a small fraction of the bandwidth to be spent on frame control information and when considering any given probability to converge within a bounded time, the proposed algorithm demonstrates a low dependency degree on the number of nodes in the network (Fig. 4 and Fig. 6).

Due to the space limit, some of the proofs appear in the Appendix.

2 Preliminaries

The system consists of a set, P, of N anonymous communicating entities, which we call *nodes*. Denote every node $p_i \in P$ with a unique index, i.

Synchronization Each node has fine-grained, real-time clock hardware. We assume that the MAC protocol is invoked periodically by synchronized *common pulse* that aligns the starting time of the TDMA frame. This can be based, for example, on TDMA alignment algorithms [38], GPS [24] or a distributed pulse synchronization algorithm [14]. The term *(broadcasting) timeslot* refers to the period between two consecutive common pulses, t_x and t_{x+1}, such that $t_{x+1} = (t_x \bmod T) + 1$, where T is a predefined constant named the

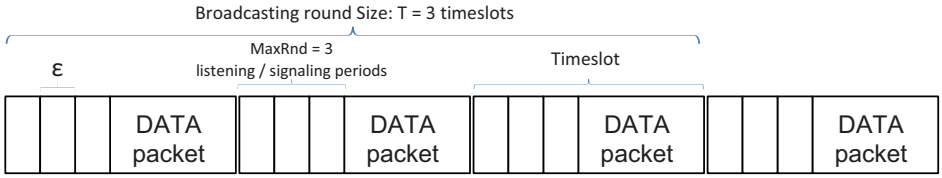


Figure 1: An example of TDMA frame, with 3 timeslots and 3 listening/signaling periods of size ε (signal exposure time).

frame size. Throughout the paper, we assume that $T \geq 2$. In our pseudo-code, we use the event timeslot(t) that is triggered by the common pulse. We assume that the timeslots are aligned.

Communications and interferences At any instance of time, the ability of any pair of nodes to directly communicate is defined by the set, $N_i \subseteq P$, of *(direct) neighbors* that node $p_i \in P$ can communicate with directly. Wireless transmissions are subject to interferences (collisions). We consider the potential of nodes to interfere with each other's communications.

Nodes raise the event carrier_sense() when they detect that the received energy levels have reached a threshold in which the radio unit is expected to succeed in carrier sense locking, see [25].

We consider a graph-based interference model, where the set $\mathcal{N}_i \supseteq N_i$ is the set of nodes that may interfere with p_i's communications when any nonempty subset of them, $I \subseteq \mathcal{N}_i : I \neq \emptyset$, transmit concurrently with p_i. We call $\mathcal{N}_i$ the *(extended) neighborhood* of node $p_i \in P$ and $d_i = |\mathcal{N}_i|$ is named the (extended) degree of node p_i. We assume that at any time, for any pair of nodes, $p_i, p_j \in P$ it holds that $p_j \in \mathcal{N}_i$ implies that $p_i \in \mathcal{N}_j$. Given a particular instance of time, we define the *(interference) graph* as $G = (P, E)$, where $E = \cup_{i \in P}\{(p_i, p_j) : p_j \in \mathcal{N}_i\}$ represents the interference relationships among nodes.

Communication schemes Timeslots allow the transmission of DATA packets using the transmit() and receive() primitives after fetching (MAC_fetch()) a new packet from the upper layer, and respectively, before delivering (MAC_deliver()) the packet to the upper layer. A *beacon* is a short packet that includes no data load, rather its carrier sense delivers important information. Before the transmission of the DATA packet in timeslot t, the scheme uses beacons for singling the node intention to transmit a DATA packet within t.

Fig. 1 depicts a TDMA frame with three timeslots. Each timeslot has a constant number, $MaxRnd = 4$, of *listening/signaling periods* in which beacons can be] sent. Each listening/signaling period take a period of ε (signal exposure time); the period during which a beacon that is sent by node $p_i \in P$ is transmitted and received by all neighbors $p_j \in \mathcal{N}_i$. Namely, the period between p_i's transition and the rise of the carrier sense event, carrier_sense(), by $p_j \in \mathcal{N}_i$.

System Settings We consider the interleaving model [16]. Every node, $p_i \in P$, executes a program that is a sequence of *atomic steps*. The *state* st_i of a node p_i consists of the value of all the variables of the node (including messages in transit for p_i). Variables are associated with individual node states by using the subscript notation, i.e., x_i is the variable x in the state of node p_i. The term *configuration* is used for a tuple of the form $(G, \{st_i\}_{i=1}^N)$, where G is the (interference) graph, and $\{st_i\}_{i=1}^N$ are the nodes' states (including the set of all incoming communications). An *execution* (run) $R = (c(0), c(1), \ldots)$ is an unbounded sequence of system configurations $c(x)$, such that each configuration $c(x+1)$ (except the initial configuration $c(0)$) is obtained from the preceding configuration $c(x)$ by the execution of steps, $\{a_i(x)\}_{p_i \in P}$, taken by all nodes.

Let τ (task) be a set of specifications and LE a set of all executions that satisfy task τ. We consider TDMA-based MAC protocols for which the task τ_{TDMA} and the set LE_{TDMA} of legal executions specify that every node has its own broadcasting timeslot that is unique within its neighborhood. We say that configuration c_{safe} is *safe* if there is an execution $R \in LE$, such that c_{safe} is R's starting configuration. Let R be an

```
   Constants, variables, macros and external functions
 2 MaxRnd (n in the proofs) : integer = bound on round number
   s : [0, T-1] ∪ {⊥} = next timeslot to broadcast or null, ⊥
 4 signal : boolean = trying to acquiring the channel
   unused[0,T-1] : boolean = marking unused timeslots
 6 unused_set = { k : unused[k] = true } : unused timeslot set (macro)
   MAC_fetch()/MAC_deliver() : MAC layer interface
 8 transmit/receive/carrier_sense : communication primitives

10 Upon timeslot(t)
     if t = 0 ∧ s = ⊥ then s := select_unused(unused_set)
12   (unused[t], signal) := (true, false) (* remove stale info. *)
     if s ≠ ⊥∧ t = s then send(MAC_fetch())
14
   Upon receive(< DATA, m> ) do MAC_deliver(< m> )

   Function send(m)  (* send message m to p_i's neighbors *)
18 for ((signal, k) := (true, 0); k := k + 1; k ≤ MaxRnd) do
     if signal then with probability ρ(k) = 1/(MaxRnd − k) do
20     signal := false (* quit the competition *)
       transmit(< BEACON> ) (* try acquiring the channel *)
22 wait until the end of competition round (* exposure period alignment *)
   if s ≠ ⊥ then transmit(< DATA, m> ) (* send the data packet *)
24
   Upon carrier_sense(t) (* defer transmission during t *)
26 if s = t ∧signal then s := ⊥ (* mark that the timeslot is not unique *)
     (signal, unused[t]) := (false, false) (* quit the competition *)
28
   Function select_unused(set) (* select an empty timeslot *)
30 if set = ∅ then return ⊥ else return uniform_select(set)
```

Figure 2: Self-stabilizing TDMA-based MAC algorithm, code of node p_i.

execution and $c \in R$ its arbitrary starting configuration. We say that R *converges* with respect to τ if within a bounded number of steps from c, the system reaches a *safe* configuration c_{safe}. The *closure* property requires that for any execution, R, that starts form c_{safe} implies that $R \in LE$. An algorithm is said to be *self-stabilizing* if it satisfies both the convergence and the closure properties.

We describe execution R as an unbounded number of concatenated finite sequences of configurations. The finite sequences, $R(x) = (c_0(x), \ldots\ c_{T-1}(x))$, $x > 0$, is a *broadcasting round* if (1) configuration $c_0(x)$ has a clock value, t, of 0 and immediately follows a configuration in which the clock value is $T-1$, and (2) configuration $c_{T-1}(x)$ has a clock value of $T-1$ and immediately precedes a configuration in which the clock value is 0.

3 Algorithm Description

The MAC algorithm in Fig. 2 assigns timeslots to nodes. Suppose that the ratio between the extended degree and the frame size is less than one, i.e., $\forall p_i \in P : 1 \lneq T/d_i$. After the convergence period, every node p_i is assigned with a broadcasting timeslot, $s_i \in [1,T]$, i.e., $\forall p_i \in P : ((s_i \in [1,T]) \wedge (p_j \in \mathcal{N}_i)) \rightarrow s_i \neq s_j$. Systems that do not satisfy the condition $\forall p_i \in P : 1 \lneq T/d_i$, eventually run into timeslots exhaustion. The algorithm indicates that the channel is busy to the nodes for which there was no timeslot left (cf. the $s_i = \perp$ assignment in line 30).

During the convergence period several nodes can be assigned to the same timeslot. Namely, we may have $p_i \in P : p_j \in \mathcal{N}_i \wedge s_i = s_j$. The algorithm solves such timeslot allocation conflicts by letting the node p_i and p_j to go through a (listening/signaling) competition before transmitting in its broadcasting timeslot. The competition rules require each node to choose one out of *MaxRnd* listening/signaling period for its broadcasting timeslot, see Fig. 1. This implies that among all the nodes that attempt to broadcast in the same timeslot, the ones that select the earliest listening/signaling period win this broadcasting timeslot and access the communication media. Before the winners access their timeslots, they signal to their neighbors that they won by transmitting beacon. The signal is sent during their choice of listening/signaling periods, see Fig. 1. When a node receives a beacon, it does not transmit during that timeslot, because it lost this (listening/signaling) competition. Instead, it randomly selects another broadcasting timeslot and competes for it on the next broadcasting round.

In detail, the MAC algorithm in Fig. 2 is invoked at the start of every timeslot, t. When t is the first timeslot, the algorithm tries to allocate the broadcasting timeslot, s_i, to p_i (line 11) by randomly selecting a timeslot for which there is no indication to be used by its neighbors. Later, when the timeslot t becomes p_i's broadcasting timeslot, s_i, the node attempts to broadcast (by calling the function send() in line 13). We note that the start of timeslot t also requires the marking of t as an unused timeslot and the removal of stale information (line 12). This indication is changed when the carrier_sense(t) event is raised (line 27) due to a neighbor transmission. Namely, when the detected energy levels reach a threshold in which the radio unit is expected to succeed in carrier sense locking, see [25].

When a node attempts to broadcast it uses the (listening/signaling) competition mechanism for deciding when to signal to its neighbors that it is about to transmit a DATA packet. The competition has *MaxRnd* rounds and it stops as soon as the node transmits a BEACON or a neighbor succeeds in signaling earlier (lines 18 to 23). We note that this singling is handle by the carrier_sense(t) event (line 27). Moreover, BEACONs are not required to carry payloads or any other information that is normally stored in packet headers. They are rather used to invoke the carrier sense event in $\mathcal{N}_i$.

The carrier sense in timeslot t indicates to each node that it needs to defer from transmission during t (line 25). In particular, it should stop using timeslot t for broadcasting, stop competing and mark t as a used timeslot. Lastly, arriving DATA packets are delivered to the upper layer (line 15).

4 Correctness Proof: Outline and Notation

The MAC task requires that every node can successfully broadcast infinitely often. We start our proof by considering wireless ad hoc networks that do not change its topology and for which the ratio between the extended degree of node and the frame size is less than one, i.e., $\forall p_i \in P : 1 \lneq T/d_i$. For these settings, we consider the task, τ_{TDMA}, in which the nodes can access successfully the media once in every broadcasting round. After showing that the MAC algorithm in Fig. 2 is self-stabilizing with respect to task τ_{TDMA} (sections 10 to 9 of the Appendix), we consider the time it take to converge within a single neighborhood (Section 5) and the entire neighborhood (Section 6). These convergence estimations facilitate the exploration of important properties, such as predictability, and dealing with changes in the network topology (Section 8).

Proof outline The exposition of the proof outline refers to Definition 4.1, which delineates the different states at which a node can be in relation to its neighbors. Definition 4.1 groups these states into three categories of *relative states*: (1) Ready to be allocated, when the node state depicts correctly its neighbor states, (2) Obtaining a timeslot, when the node is competing for one, but there is no agreement with its neighbor states, and (3) Allocated to a timeslot, when the node is the only one to be allocated to a particular timeslot in its neighborhood. The correctness proof shows that the MAC algorithm in Fig. 2 implements τ_{TDMA} in a self-stabilizing manner by showing that eventually all nodes are allocated with timeslots, i.e., all nodes are in the relative state Allocated, see Definition 4.1.

Let R be an execution of the MAC algorithm in Fig. 2 and $R(x)$ is the x-th complete broadcasting round of R, where $x > 0$ is an integer. We simplify the presentation by using uppercase notation for the configurations, $c_t^{\text{name}}(x)$, where $t \in [1,T]$ is a timeslot. This notation includes the *name* of the first event to be triggered immediately after configuration c, i.e., $R(x) = (c_0^{\text{timeslot}}(x), \ldots c_{T-1}^{\text{carrier\_sense/receive}}(x))$.

Definition 4.1. *We say that node $p_i \in P$ is* Ready (to be allocated) *to a timeslot in configuration $c_0^{timeslot}(x)$, if properties* (1), (2) *and* (3) *hold for node p_i but Property* (4) *does not. We say that p_i is* Obtaining *timeslot s_i in configuration $c_0^{timeslot}(x)$, if properties* (1) *to* (4) *hold for node p_i, but Property* (5) *does not. We say that*

node $p_i \in P$ is in Allocated *state, with respect to timeslot s_i in configuration $c_0^{timeslot}(x)$, if properties* (1) *to* (5) *hold for node p_i.*

$$signal_i = \mathit{false} \tag{1}$$

$$(t \in unused_i \wedge t \neq s_i) \leftrightarrow (\forall p_k \in \mathcal{N}_i : s_k \neq t) \tag{2}$$

$$s_i \neq \perp \vee unused\_set_i \setminus \{s_i\} \neq \emptyset \tag{3}$$

$$s_i \neq \perp \tag{4}$$

$$\forall p_j \in \mathcal{N}_i : ((s_i \neq s_j) \wedge (unused_j[s_i] = \mathit{false})) \tag{5}$$

Property (1) implies that node p_i finishes any broadcast attempts within a timeslot. Properties (2) to (3) consider the case in which p_i's internal state represents correctly the timeslot allocation in its neighborhood. In particular, property (2) means that processor p_i views timeslot t as an unused one if, and only if, it is indeed unused. Property (3) implies that when node p_i is not using any timeslot, there is an unused timeslot at its disposal. Property (4) says that node p_i is using timeslot s_i. Property (5) refers to situations in which p_i's neighbors are not using p_i's timeslot during the next broadcasting round.

Starting from an arbitrary configuration, we show that node p_i becomes Ready within two broadcasting rounds (or one complete broadcasting round), see Section 10 of the Appendix. Then, we consider the probability, $OnlyOne_i(x)$, that a node enters the relative state Allocated from either Ready or Obtaining, see equation (6) (and sections 11 and 12 of the Appendix). Namely, equation (6) considers the probability that node p_i is the *only one* to use its broadcasting timeslot in its neighborhood, where $\rho_k = 1/MaxRnd = 1/n$ is p_i's probability to selects the k-th listening/signaling period for transmitting its BEACON.

$$\boxed{OnlyOne_i(x) \geq \sum_{k=1}^{n} \rho_k \left(1 - \sum_{\ell=1}^{k} \rho_k\right)^{\frac{d_i}{T}}} \tag{6}$$

Theorem 4.2 demonstrates self-stabilization, rather than convergence from an randomized starting configuration, as in [34].

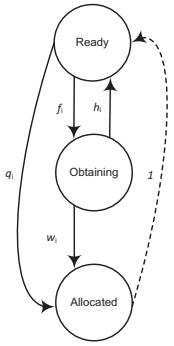
Theorem 4.2 (Self-Stabilization, the proof appears in Section 9 of the Appendix)**.** *The MAC algorithm in Fig. 2 is self-stabilizing with respect to the task τ_{TDMA}.*

We bound the time it takes the MAC algorithm in Fig. 2 to converge by considering the relative states, Ready, Obtaining, and Allocated, and describe a state machine of a Markovian process. This process is used for bounding the convergence time of a single node (Section 5), and the entire network (Section 6).

In detail, give node $p_i \in P$, its neighborhood, $\mathcal{N}_i$, we define a random environment of a Markov chain, see Fig. 3. By looking at this random environment, we can focus our analysis on p_i's relative states while avoiding probability dependencies and considering average probabilities [9]. Suppose that p_i's environment, e, is known. Theorem 5.1 estimates two bounds on the expectation of probability, $q_i \mid_e$, which is literally the probability q_i given that the environment is e.

In order to do that, we consider a set, $\mathcal{R}$, of executions of the MAC algorithm, such that each execution $R \in \mathcal{R}$ starts in a configuration, $c \in R$, in which: (I) for any node $p_j \in P$, properties (1), (2) and (3) holds, and (II) node p_i is in the relative state Ready, which implies that (III) eventually, node p_i arrives to the relative state Allocated.

With this convention, we can add a probability 1 to transit from the relative state Allocated to Ready, see the dashed line in the state-machine diagram of Fig. 3. This allows us to estimate the expected time to reach



We look at p_i's state transition with relation to its neighbors, see Definition 4.1. The figure on the right defines p_i's relative states as a 3-state Markov chain. The probabilities, q_i, w_i, f_i, and h_i (solid lines arrows), that node p_i change its relative state depends on its neighbor's state. For instance, q_i is the probability that p_i goes from the relative state Ready to Allocated. It is environment dependent, i.e., the states of p_i's neighbors are random as well. We add the dotted edge between the state Allocated and the state Ready in order to make the Markov chain irreducible and to allow working with the invariant probability. Namely, once node p_i arrives to Allocated, it returns to Ready with probability 1. With this convention, we can estimate the expected time to reach the final relative state Allocated from relative state Ready by the expectation of the first hitting time of the irreducible chain [3]

Figure 3: Markov chain describing p_i's relative state transitions.

the final relative state Allocated from relative state Ready by the expectation of the first hitting time of the irreducible Markov chain [3].

When computing the expected time for node p_i to reach state Allocated within its neighborhood, we see that it is sufficient to consider the lower bound of the probability $OnlyOne_i(x)$ to obtain an upper bound on the expected time to converge, see section 5. Moreover, when considering the network convergence time, i.e., the expected convergence time of all nodes in the network, we see that the most dominant parameter is the mean neighborhood size. We do that by applying the AM-GM (Arithmetic Mean vs Geometric Mean) inequality and bounding the expected network convergence time, see Section 6.

Notation Throughout the paper, we denote the states of the Markov chain by $\{X_t\}_{t\geq 0}$, $T_i^+ = \min\{t > 0 \text{ such that } X_t = i\}$ and $E_i(\cdot)$ is the expectation given that we start in relative state i, $E_i\left(T_i^+\right) = E\left(T_i^+ \mid X_0 = i\right)$. In this paper, the states 1, 2, and 3 of the Markovian process correspond respectively to states Ready, Obtaining and Allocated, and the time $t = 0,1,\ldots$ corresponds to configuration $c_0^{timeslot}(x+t) \in R(x+t)$, where $R(x)$ is the first complete broadcasting round in R that starts in a configuration, $c_0^{timeslot}(x)$, in which all nodes are in the relative state Ready. For example, $E_3\left(T_3^+\right)$ is the expected time to reach the Allocated state.

Let $p_i \in P$ be a node for which $s_i \neq \perp \wedge \exists p_j \in \mathcal{N}_i : s_j = s_i$ in configuration $c_0^{\text{timeslot}}(x)$. We define $M_i(x) = \{p_j \in \mathcal{N}_i : s_i = s_j\}$ to be the set of p_i's (broadcasting timeslot) *matching* neighbors, which includes all of p_i's neighbors that, during broadcasting round $R(x)$, are attempting to broadcast in p_i's timeslot. In our proofs, we use n as the number of *listening/signaling periods*, $MaxRnd$.

5 Convergence within a Neighborhood

Theorem 5.1 bounds the expected time, $\mathcal{S}_i$, for a node to reach the relative state Allocated, and follows from Proposition 5.3 and equation (12). Note that $\mathcal{S}_i \leq 4$ when the number of listening/signaling periods is $n \geq 2$, and considering saturated situations in which the extended node degree $d_i < T$ is smaller than the TDMA

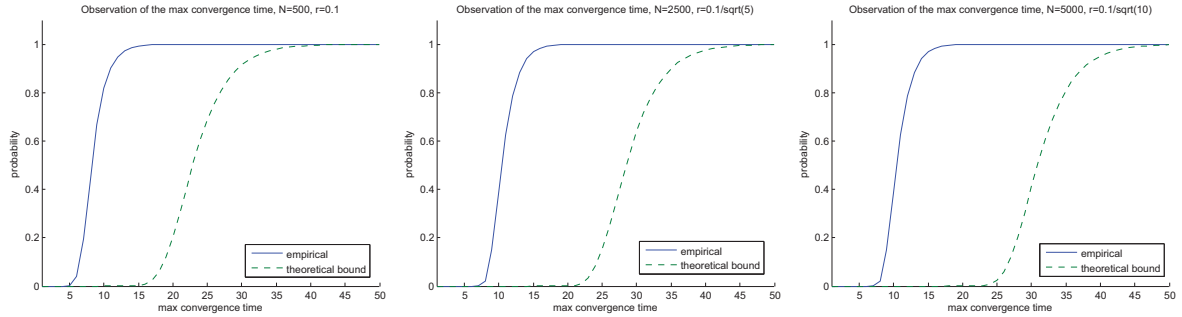


Figure 4: Numerical validation of Theorem 6.2's bound on the network-wise convergence time. We compare the bound, $P(t_{\max} < k) = (1-(1-q)^k)^N$, with the numerical results, which consider random geometric graphs in which the nodes are randomly placed on the unit square. The charts considers $N \in \{500, 2500, 5000\}$ nodes (from left to right). All experiments considered 2 listening/signaling periods, interference range of $0.1/\sqrt{(\frac{N}{500})}$, which result in an average extended degree of 15, $d_i/T = 1$ on average, and $q_i = 1/4$.

frame size. Namely, the proposed algorithm convergence with a neighborhood is brief.

Theorem 5.1 (Local Convergence)**.** *The expected time, $\mathcal{S}_i$, for node $p_i \in P$ to reach the relative state Allocated satisfies equation* (7)*, where n is the number of listening/signaling periods, T the TDMA frame size, and d_i is p_i's extended degree.*

$$\boxed{\mathcal{S}_i \leq \min\{\left(\frac{2n}{n-1}\right)^{\frac{d_i}{T}}, \frac{\frac{d_i}{T}+1}{n}\left(\frac{n}{n-1}\right)^{\frac{d_i}{T}+1}\}} \tag{7}$$

We look into the transition probability among relative states by depicting the diagram of Fig. 3 as an homogeneous Markov chain. We estimate the diagram transition probabilities in a way that maximizes the expected time for reaching the diagram's final state, Allocated. It is known that the first hitting time is given by $E_i\left(T_i^+\right) = \frac{1}{\pi_i}$, where $\pi = (\pi_1, \pi_2, \pi_3)$ is the invariant probability vector [3]. Let $\mathcal{S}_i$ be the expected time it takes node p_i that starts at the relative state Ready to reach Allocated. It is clear that $\mathcal{S}_i = T_3^+ - 1$, because $T_3^+ - 1$ is the return time of the relative state Allocated. In our case, the transition matrix P is given by equation (8).

$$P = \begin{pmatrix} 1-f_i-q_i & f_i & q_i \\ h_i & 1-h_i-w_i & w_i \\ 1 & 0 & 0 \end{pmatrix} \tag{8}$$

The invariant probability vector π satisfying $\pi P = \pi$ is given by equation (9).

$$\pi = \frac{\left(h_i+w_i, f_i, q_ih_i+q_iw_i+f_iw_i\right)}{h_i+w_i+f_i+h_iq_i+q_iw_i+f_iw_i} \tag{9}$$

The estimation of the maximal expected time necessary to assign the node p_i to a timeslot requires to compute bounds on the probabilities f_i, h_i, q_i and w_i that maximize equation (10).

$$E_3\left(T_3^+\right) = \frac{1}{\pi_3} = \frac{h_i+w_i+f_i+h_iq_i+q_iw_i+f_iw_i}{q_ih_i+q_iw_i+f_iw_i} \tag{10}$$

The expected time for p_i to reach the relative state Allocated is bounded in equation (11).

$$\boxed{\mathcal{S}_i = E_3\left(T_3^+\right) - 1 = \frac{h_i + w_i + f_i}{q_i h_i + q_i w_i + f_i w_i}} \tag{11}$$

Equation (7) has a compact and meaningful bound for equation (11). We achieve that by studying the impact of the parameters T and n on the MAC algorithm in Fig. 2. Lemma 5.2 and equation (11) imply equation (12).

$$\boxed{\mathcal{S}_i \leq \frac{h_i + w_i + f_i}{q_i h_i + q_i w_i + f_i q_i} = \frac{1}{q_i}} \tag{12}$$

Lemma 5.2. *Suppose that $n \geq 2$ is the number of listening/signaling periods, see line 2 of the code in Fig. 2. Then $w_i \geq q_i$.*

Proof. Let us consider node $p_i \in P$ that is in relative state Ready. Given that p_i has v_i neighbors that compete for the same timeslot, the probability that p_i gets allocated, $q_i \mid_{v_i}$, is given by equation (13).

$$q_i \mid_{v_i} = \sum_{k=1}^{n-1} \rho_k \left(1 - \rho_1 - \ldots - \rho_k\right)^{v_i} \tag{13}$$

Consider next that p_i is in relative state Obtaining, and thus we know that p_i transmitted during the preceding broadcasting round and transited from relative state Ready to Obtaining. Moreover, p_i is using the same timeslot for the current broadcasting round. The only neighbors of p_i that are using the same timeslot are the neighbors that are also in relative state Obtaining and, have chosen the same listening/signaling period as p_i during the preceding broadcasting round. Let us denote by ℓ_i the number of such neighbors. Given ℓ_i the probability $w_i \mid_{\ell_i}$ that p_i is allocated to the timeslot is given by equation (14).

$$w_i \mid_{\ell_i} = \sum_{k=1}^{n-1} \rho_k \left(1 - \rho_1 - \ldots\ldots - \rho_k\right)^{\ell_i} \tag{14}$$

We have that ℓ_i is stochastically dominated by v_i [40], i.e., $E(\ell_i) \leq E(v_i)$. Indeed, v_i is a random variable that counts the number of neighbors that choose the same timeslot as p_i while ℓ_i counts the number of neighbors that choose the same timeslot *and* listening/signaling period as p_i. For $n \geq 2$, ℓ_i's expected value is smaller than v_i's expected value. To conclude, we remark that expressions (13) and (14) are the same decreasing function, $f_i \to \sum_{k=1}^{n-1} \rho_k \left(1 - \rho_1 - \ldots - \rho_k\right)^{f_i}$, that is evaluated at two different point, v_i and ℓ_i respectively. Moreover, since ℓ_i is stochastically dominated by v_i, equation (15) holds.

$$w_i = E\left(w_i \mid_{\ell_i}\right) \geq E\left(q_i \mid_{v_i}\right) = q_i \tag{15}$$

□

Proposition 5.3 demonstrates equation (16) and leads us toward the proof of Theorem 5.1.

Proposition 5.3. *Let $\rho_i = 1/MaxRnd$. Equation (16) bounds from below the probability q_i, see Section 12 of the Appendix.*

$$q_i \geq \max\{\left(\frac{n-1}{2n}\right)^{\frac{d_i}{T}}, \frac{1}{\frac{d_i}{T}+1}\left(1-\frac{1}{n}\right)^{\frac{d_i}{T}+1}\} \tag{16}$$

The first bound, $\frac{1}{q_i} \leq \left(\frac{2n}{n-1}\right)^{\frac{d_i}{T}}$ (equation (7)), has a simple intuitive interpretation. Let us consider first that two nodes compete for a same timeslot. The two nodes choose independently any of the n listening/signaling periods and there are n^2 different possible outcomes. Among these outcomes n correspond to the situation where the two nodes choose the same listening/signaling period and there is no winner. We then have $n^2-n=n(n-1)$ outcomes that lead to a winner. There is then a probability of $n(n-1)/n^2=(n-1)/n$ that one of the node wins the (listening/signaling) competition. Since the game is symmetric, the probability that p_i wins is $(n-1)/(2n)$. The fact that we have T timeslots divides the number of competing nodes, d_i, and imply that there are d_i/T competing nodes to the same timeslot. If we interpret the game as a collection of d_i/T independent games, where for each game p_i wins with probability $(n-1)/(2n)$. Thus, the probability q_i that p_i wins is $\left(\frac{n-1}{2n}\right)^{\frac{d_i}{T}}$. The inverse of this expression gives the average time for the event to occur and is the bound by equation (7).

6 Network Convergence

We estimate the expected time for the entire network to reach a safe configuration in which all nodes are allocated with timeslots. The estimation is based on the number of nodes that are the earliest to signal in their broadcasting timeslot. These nodes are winners of the (listening/signaling) competition and are allocated to their chosen timeslots. However, counting only these nodes leads to under-estimate the number of allocated nodes, which then results in an over-estimation of the convergence time. Indeed, node $p_i \in P$ might have a neighbor $p_j \in \mathcal{N}_i$ that selects the earliest listening/signaling period in $\mathcal{N}_i$, but p_j does not transmit because one of its neighbors, $p_k \in \mathcal{N}_j \setminus \mathcal{N}_i$, had transmitted in an earlier listening/signaling period. Our bound consider only p_k while both p_i and p_k transmit, became p_j is inhibited by p_k's BEACON.

Lemma 6.1 shows that the assumption that the nodes are allocated independently of each other's is suitable for bounding the network convergence time, $\mathcal{S}$. Theorem 6.2 uses Lemma 6.1 for bounding the network convergence time, $\mathcal{S}$.

In Section 5, we prove a bound on the expected time, $\mathcal{S}_i$, for a *single* node to be allocated to a timeslot. We observe that the bound depends uniquely on the number of listening/signaling periods, n, as well as the ratio between the extended degree and the frame size, d_i/T. In order to obtain a bound valid for all nodes, we bound this ratio with x/T where x is as defined in Lemma 6.1. We note that the time needed for the allocation of timeslots to *all* the nodes depends on N, the total number of nodes.

In detail, the convergence time estimation considers the (fixed and independent) bound, q_i, for the probability that a node reach the relative state Allocated within a broadcasting round. Then, the convergence time, t, is a random variable with geometric probability, i.e., $P(t=k)=(1-q)^{k-1}q$. Let us denote $t_1,\ldots t_N$ the time it takes for the nodes $p_1,\ldots p_N$ to respectively reach the relative state Allocated. The convergence time, $\mathcal{S}$, for all the nodes is given by $\max(\{t_1,\ldots t_N\})$, which depends on N.

Lemma 6.1. *The expected number of nodes, $E(W)$, that win the (listening/signaling) competition after one broadcasting round satisfies equation (17), where $x=\frac{2A}{N}$, T is the number of timeslots, A the number of edges in the interference graph, G, and $N=\mid P \mid$ the number of nodes that attempt to access the communi-*

cation media.

$$E(W) \geq N \sum_{j=1}^{n} \rho_j \left(1 - (\rho_1 + \ldots + \rho_j)\right)^{\frac{x}{T}} \tag{17}$$

Proof. The nodes that are allocated to a timeslot can previously being on relative state Ready or Obtaining. The probability of a transition from relative state Obtaining to Allocated is w_i, and, a transition from relative state Ready to Allocated is q_i. As proved in Lemma 5.2, we always have $w_i \geq q_i$. To bound the number of nodes that get allocated during a broadcasting round, we use the lower bound on the probability q_i that a node gets allocated to a timeslot. Moreover, in the computations, we use the AM-GM bound [46], which says that if $\sum b_k = 1$ then $\prod a_k^{b_k} \leq \sum b_k a_k$ and, denote d_i the number of neighbors of node p_i. As proved in Proposition 12.1, since there are T timeslots the number of neighbors of i that choose the same timeslot as i and compete for it is bounded by d_i/T . This lemma is proved by equation (18), where the last line of the expression holds because $\sum_i d_i = 2A$.

$$\begin{aligned}
E(W) \geq & \qquad (18)\\
E\left(\sum_{i=1}^{N} 1_{|p_i \text{ selects the earliest signaling period}}\right) &= \\
\sum_{i=1}^{N}\left(\rho_1 (1-\rho_1)^{\frac{d_i}{T}} + \ldots \rho_{n-1}\left(1 - \sum_{k=1}^{n-1}\rho_k\right)^{\frac{d_i}{T}}\right) &= \\
\sum_{j=1}^{n} N \sum_{i=1}^{N} \frac{1}{N} \rho_j \left(1 - \sum_{k=1}^{j}\rho_k\right)^{\frac{d_i}{T}} &\geq
\end{aligned}$$

$$\begin{aligned}
N \sum_{j=1}^{n} \prod_{i=1}^{N} \rho_j^{\frac{1}{N}} \left(1 - \sum_{k=1}^{j}\rho_k\right)^{\frac{d_i}{NT}} &= \qquad (19)\\
N \sum_{j=1}^{n} \rho_j \left(1 - \sum_{k=1}^{j}\rho_k\right)^{\frac{1}{TN}\sum d_i} &= \\
N \sum_{j=1}^{n} \rho_j \left(1 - \sum_{k=1}^{j}\rho_k\right)^{\frac{x}{T}} &
\end{aligned}$$

We note that we use the AM-GM bound to reach the 4-th raw of equation (18).

□

By arguments similar to the ones used in the proof of Proposition 5.3, we deduce that if N nodes compete, the expected number $E(W)$ of nodes that get allocated to a timeslot is lower bounded in equation (20).

$$E(W) \geq N \max\left\{\left(\frac{n-1}{2n}\right)^{\frac{x}{T}}, \frac{\left(\frac{n-1}{n}\right)^{\frac{x}{T}+1}}{\frac{x}{T}+1}\right\} \tag{20}$$

Theorem 6.2 bounds the system convergence time (see the proof in Section 13 of the Appendix).

Theorem 6.2 (Global Convergence)**.** *The expected number of retransmissions is smaller than* $\left(\frac{2n}{n-1}\right)^{d/T} - 1$*, where* $d = \max(\{d_i : p_i \in P\})$*. Hence, we have that the expected number of broadcasting rounds,* $\mathcal{S}$*, that guarantee that all nodes to reach the relative state* Allocated *satisfies equation* (21)*.*

$$\mathcal{S} \leq \left(\frac{2n}{n-1}\right)^{d/T} \tag{21}$$

Moreover, given that there are N nodes in the network and $\alpha \in (0,1)$*, the network convergence time is bounded by equation* (22) *with probability* $1-\alpha$*.*

$$k = 1 + \frac{log\left(1 - \sqrt[N]{1-\alpha}\right)}{log\left(1 - \left(\frac{n-1}{2n}\right)^{\frac{d}{T}}\right)} \tag{22}$$

This means that with probability α *all nodes are allocated with timeslots in maximum k broadcasting rounds, see Fig.* (6)*.*

We numerically validate Theorem 6.2, see Fig. 4. Moreover, our experiments showed that the average convergence time of the network is below the upper bound of equation (21).

7 Implementation

Existing MAC protocols offer mechanisms for dealing with contention (timeslot exhaustion) via policies for administering message priority, such as [39]. In particular, the IEEE `802.11p` standard considers four priories and techniques for facilitating their policy implementation. We explain similar techniques that can facilitate the needed mechanisms.

Peritonized listening/signaling periods One can consider listening period parameters, $[LSP_{start}, LSP_{end}]$, that refer to the first, and respectively, the last listening/signaling periods that a node can use when attempting to acquire a broadcasting timeslot. E.g., suppose that there are six listening/signaling periods, and that nodes with the highest priory may use the first three listening/signaling periods, $[0,2]$, and nodes with the lowest priority may use the last three, $[3,5]$. In the case of two neighbors with different listening period parameters, say $[0,2]$ and $[3,5]$, that attempt to acquire the same broadcasting timeslot, the highest priority node always attempts to broadcast before the lowest priority one.

TDMA-based back-off Let us consider two back-off parameters, CW_{start} and CW_{end}, that refer to the maximal and minimal values of the contention window. Before selecting an unused timeslot, the procedure counts a random number of unused ones. Fig. 5 presents an implementation of the select_unused() function that facilitates back-off strategies as an alternative to the implementation presented in line 29 of Fig. 2.

The statically allocated variable *count* records the number of backoff steps that node p_i takes until it reaches the zero value. Whenever the function select_unused() is invoked with $count_i = 0$, node p_i assigns to $count_i$ a random integer from $[CW_{start}, CW_{end}]$ (cf. line 7). Whenever the value of $count_i$ is not greater than the number of unused timeslots, the returned timeslot is selected uniformly at random (cf. lines 8 to 9). Otherwise, a $\perp$-value is returned after deducting the number of unused timeslots during the previous broadcasting round (cf. lines 6 and 10).

8 Discussion

Thus far, both schedule-based and non-schedule-based MAC algorithms could not consider timing requirements within a provably short recovery period that follows (arbitrary) transient faults and network topology

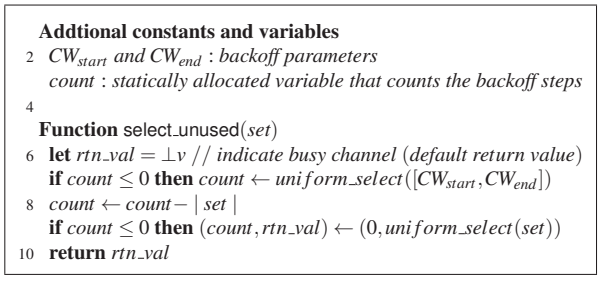


Figure 5: select_unused() with TDMA-based back-off

changes. This work proposes the first self-stabilizing TDMA algorithm for DynWANs that has a provably short convergence period. Thus, the proposed algorithm possesses a greater degree of predictability, while maintaining low communication delays and high throughput.

In this discussion, we would like to point out the algorithm's ability to facilitate the satisfaction of severe timing requirements for DynWANs by numerically validating Theorem 6.2. As a case study, we show that, for the considered settings of Fig. 4, the global convergence time is brief and definitive. Fig. 6 shows that when allowing merely a small fraction of the bandwidth to be spent on frame control information, say three listening/signaling periods, and when considering 99% probability to convergence within a couple of dozen TDMA frames, the proposed algorithm demonstrates a low dependency degree on the number of nodes in the network even when considering $10,000$ nodes. We have implemented the proposed algorithm, extensively validated our analysis via computer simulation, and tested it on a platform with more than two dozen nodes. (This technical report can be made available.) These results indeed validate that the proposed algorithm can indeed facilitate the implementation of MAC protocols that guarantee satisfying these severe timing requirements.

The costs associated with predictable communications, say, using base-stations, motivate the adoption of new networking technologies, such as MANETs and VANETs. In the context of these technologies, we expect that our proposal would contribute to the development of MAC protocols that can be used by applications that needs guarantees for severe timing requirements.

References

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

[2] Fabrizio Abrate, Andrea Vesco, and Riccardo Scopigno. An analytical packet error rate model for wave receivers. In *VTC Fall*, pages 1–5. IEEE, 2011.

[3] David Aldous and Jim Fill. Reversible Markov chain and random walks on graph. unpublished, available at `http://www.stat.berkeley.edu/~aldous/RWG/book.html`, 1999.

[4] M. Arumugam and S.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.

[5] Mahesh Arumugam and Sandeep S. Kulkarni. Self-stabilizing deterministic TDMA for sensor networks. In Goutam Chakraborty, editor, *2nd Inter. Conf. Distributed Computing and Internet Technology (ICDCIT)*, volume 3816 of *LNCS*, pages 69–81. Springer, 2005.

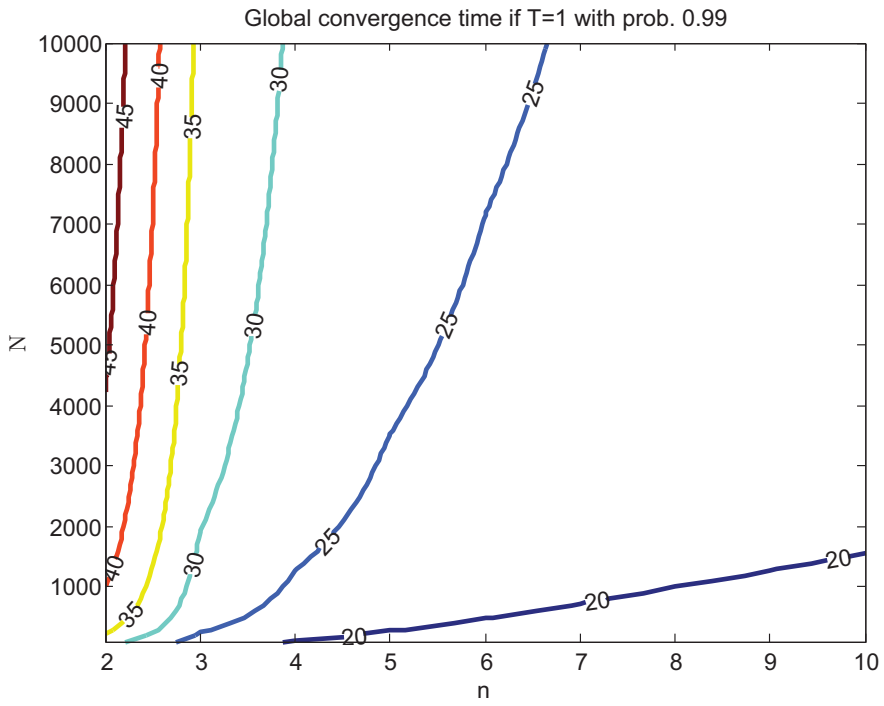


Figure 6: Contour plot of equation (22) for $s = d/T = 1$. *Contour charts* [13] present two parameter functions, e.g., the convergence time function, $k(n,N)$ presented in equation (22). Contour lines in Fig. 6 connect values of $k(n,N)$ that are the same (see the text tags along the line). When N nodes attempt to access the medium, the conversance time, $\mathcal{S}$ (cf. the contour lines), is stable in the presence of a growing number, n, of listening/signaling periods.

[6] Katrin Bilstrup, Elisabeth Uhlemann, Erik G. Ström, and Urban Bilstrup. Evaluation of the IEEE 802.11p MAC method for vehicle-to-vehicle communication. In *VTC Fall*, pages 1–5. IEEE, 2008.

[7] Katrin Bilstrup, Elisabeth Uhlemann, Erik G. Ström, and Urban Bilstrup. On the ability of the 802.11p MAC method and STDMA to support real-time vehicle-to-vehicle communication. *EURASIP Journal on Wireless Comm. & Net.*, 2009:1–13, 2009.

[8] Jean R. S. Blair and Fredrik Manne. An efficient self-stabilizing distance-2 coloring algorithm. In Shay Kutten and Janez Zerovnik, editors, *SIROCCO*, volume 5869 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 2009.

[9] Robert Cogburn. The Ergodic theory of Markov chains in random environments. *Probability Theory and Related Fields (Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete)*, 66(1):109–128, 1984.

[10] Alejandro Cornejo and Fabian Kuhn. Deploying wireless networks with beeps. In Nancy A. Lynch and Alexander A. Shvartsman, editors, *24th Inter. Symp. on Distributed Computing (DISC'10)*, volume 6343 of *LNCS*, pages 148–162. Springer, 2010.

[11] Hector Agustin Cozzetti and Riccardo Scopigno. Rr-aloha+: A slotted and distributed mac protocol for vehicular communications. In *Vehicular Networking Conference (VNC), 2009 IEEE*, pages 1 –8, Oct. 2009.

[12] Hector Agustin Cozzetti, Riccardo Scopigno, Luca Casone, and Giuseppe Barba. Comparative analysis of ieee 802.11p and ms-aloha in vanet scenarios. In Markus Kirchberg, Patrick C. K. Hung, Barbara Carminati, Chi-Hung Chi, Rajaraman Kanagasabai, Emanuele Della Valle, Kun-Chan Lan, and Ling-Jyh Chen, editors, *APSCC*, pages 64–69. IEEE, 2009.

[13] AM Crocker, WL Godson, and CM Penner. Frontal contour charts. *Journal of the Atmospheric Sciences*, 4(3), 1947.

[14] Ariel Daliot, Danny Dolev, and Hanna Parnas. Self-stabilizing pulse synchronization inspired by biological pacemaker networks. In Shing-Tsaan Huang and Ted Herman, editors, *6th Inter. Symp. on Self-Stabilizing Systems (SSS)*, volume 2704 of *LNCS*, pages 32–48. Springer, 2003.

[15] Murat Demirbas and Muzammil Hussain. A MAC layer protocol for priority-based reliable multicast in wireless ad hoc networks. In *3rd Inter. Conf. on Broadband Comm., Net., and Systems (BROADNETS)*. IEEE, 2006.

[16] Shlomi Dolev. *Self-Stabilization*. MIT Press, 2000.

[17] Shlomi Dolev and Ted Herman. Superstabilizing protocols for dynamic distributed systems. *Chicago J. Theor. Comput. Sci.*, 1997.

[18] Shlomi Dolev, Amos Israeli, and Shlomo Moran. Analyzing expected time by scheduler-luck games. *IEEE Trans. Software Eng.*, 21(5):429–439, 1995.

[19] Olga Goussevskaia, Roger Wattenhofer, Magnús M. Halldórsson, and Emo Welzl. Capacity of arbitrary wireless networks. In *INFOCOM*, pages 1872–1880. IEEE, 2009.

[20] Martin Haenggi. Outage, local throughput, and capacity of random wireless networks. *Trans. Wireless. Comm.*, 8(8):4350–4359, 2009.

[21] Magnús M. Halldórsson and Roger Wattenhofer. Wireless comm. is in apx. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *ICALP (1)*, volume 5555 of *LNCS*, pages 525–536. Springer, 2009.

[22] H. Hartenstein and K. Laberteaux. *VANET: Vehicular Applications and Inter-Networking Technologies*. Wiley, 2010.

[23] Ted Herman and Sébastien Tixeuil. A distributed TDMA slot assignment algorithm for wireless sensor networks. In *5th Inter. Workshop on Algo. Aspects of Wireless Sensor Net. (ALGOSENSORS)*, volume 3121 of *LNCS*, pages 45–58. Springer, 2004.

[24] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global positioning System. Theory and Practice*. Springer-Verlag, 1993.

[25] Kyle Jamieson, Bret Hull, Allen K. Miu, and Hari Balakrishnan. Understanding the real-world performance of carrier sense. In *ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis (E-WIND)*, Philadelphia, PA, August 2005.

[26] Johan Ludwig William Valdemar Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30(1):175–193, 1906.

[27] Arshad Jhumka and Sandeep S. Kulkarni. On the design of mobility-tolerant TDMA-based media access control (MAC) protocol for mobile sensor networks. In Tomasz Janowski and Hrushikesha Mohanty, editors, *ICDCIT*, volume 4882 of *LNCS*, pages 42–53. Springer, 2007.

[28] Fabian Kuhn, Nancy A. Lynch, and Calvin C. Newport. The abstract MAC layer. In Idit Keidar, editor, *23rd Inter. Symp. on Distributed Computing (DISC)*, volume 5805 of *LNCS*, pages 48–62. Springer, 2009.

[29] Sandeep S. Kulkarni and Mahesh Umamaheswaran Arumugam. *Sensor Network Operations*, chapter SS-TDMA: A self-stabilizing MAC for sensor networks. IEEE Press, 2006.

[30] Andreas Lagemann, Jörg Nolte, Christoph Weyer, and Volker Turau. Mission statement: Applying self-stabilization to wireless sensor networks. In *8th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze" (FGSN)*, pages 47–49, August 2009.

[31] M. Lenoble, K. Ito, Y. Tadokoro, M. Takanashi, and K. Sanda. Header reduction to increase the throughput in decentralized TDMA-based vehicular networks. In *Vehicular Networking Conference (VNC), 2009 IEEE*, pages 1–4. IEEE, 2009.

[32] Christoph Lenzen, Thomas Locher, Philipp Sommer, and Roger Wattenhofer. Clock Synchronization: Open Problems in Theory and Practice. In *36th Inter. Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM), Ŏpindleruv Mlýn, Czech Republic*, January 2010.

[33] Christoph Lenzen, Jukka Suomela, and Roger Wattenhofer. Local algorithms: Self-stabilization on speed. In Rachid Guerraoui and Franck Petit, editors, *11th Inter. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 5873 of *LNCS*, pages 17–34. Springer, 2009.

[34] Pierre Leone, Marina Papatriantafilou, and Elad Michael Schiller. Relocation analysis of stabilizing MAC algorithms for large-scale mobile ad hoc networks. In *5th Inter. Workshop on Algo. Aspects of Wireless Sensor Net. (ALGOSENSORS)*, pages 203–217, 2009.

[35] Pierre Leone, Marina Papatriantafilou, Elad Michael Schiller, and Gongxi 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)*, October 2009. Hamburg, Germany.

[36] Pierre Leone, Marina Papatriantafilou, Elad Michael Schiller, and Gongxi Zhu. Chameleon-mac: Adaptive and self-$\star$ algorithms for media access control in mobile ad hoc networks. In Shlomi Dolev, Jorge Arturo Cobb, Michael J. Fischer, and Moti Yung, editors, *12th Inter. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS'10)*, volume 6366 of *LNCS*, pages 468–488. Springer, 2010.

[37] Torgny Lindvall. *Lectures on the Coupling Method*. Dover Publications, Inc., 1992.

[38] Mohamed Mustafa, Marina Papatriantafilou, Elad Michael Schiller, Amir Tohidi, and Philippas Tsigas. Autonomous TDMA alignment for VANETs. In *IEEE 76th Vehicular Technology Conference (VTC'12-Fall)*, 2012.

[39] Raphael Rom and Fouad A. Tobagi. Message-based priority functions in local multiaccess communication systems. *Computer Networks*, 5:273–286, 1981.

[40] Sheldon M. Ross. *Stochastic Processes*. John Wiley & Sons, Inc., 1996.

[41] William G. Schmidt. Satellite time-division multiple access systems: Past, present and future. *TeleComm.*, 7:21–24, 1974.

[42] Johannes Schneider and Roger Wattenhofer. Coloring unstructured wireless multi-hop networks. In Srikanta Tirthapura and Lorenzo Alvisi, editors, *28th Annual ACM Symp. on Principles of Distributed Computing (PODC)*, pages 210–219. ACM, 2009.

[43] Riccardo Scopigno and Hector Agustin Cozzetti. Mobile slotted aloha for VANETs. In *VTC Fall*, pages 1 – 5. IEEE, 2009.

[44] Riccardo Scopigno and Hector Agustin Cozzetti. GNSS synchronization in VANETs. In Khaldoun Al Agha, Mohamad Badra, and Gregory B. Newby, editors, *NTMS*, pages 1–5. IEEE, 2009.

[45] Riccardo Scopigno and Hector Agustin Cozzetti. Evaluation of time-space efficiency in CSMA/CA and slotted VANETs. In *VTC Fall*, pages 1–5. IEEE, 2010.

[46] J.M. Steele. *The Cauchy-Schwartz Master Class*. Cambridge University Press, 2004.

[47] Yukihiro Tadokoro, Kenji Ito, Junji Imai, Noriyoshi Suzuki, and Nobuo Itoh. Advanced transmission cycle control scheme for autonomous decentralized TDMA protocol in safe driving support systems. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 1062–1067, june 2008.

[48] Saira Viqar and Jennifer L. Welch. Deterministic collision free communication despite continuous motion. In *5th Inter. Workshop on Algo. Aspects of Wireless Sensor Net. (ALGOSENSORS)*, pages 218–229, 2009.

[49] Roger Wattenhofer. Theory Meets Practice, It's about Time! In *36th Inter. Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM), Špindleruv Mlýn, Czech Republic*, January 2010.

[50] Wei Ye, John S. Heidemann, and Deborah Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *INFOCOM*, 2002.

[51] Fan Yu and Subir 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.

Appendix

9 Theorem 4.2

Theorem 4.2 shows that all nodes are allocated eventually with timeslots (convergence) and once all nodes are allocated, they stay this way (closure). We note that Theorem 4.2 refers to Proposition 11.2.

Theorem 4.2 (Self-Stabilization) *The MAC algorithm in Fig. 2 is a self-stabilizing algorithm with respect to the task* τ_{TDMA}.

Proof. • **Convergence** We need to show that properties (1) to (5) eventually hold in configuration $c_0^{\text{timeslot}}(x+y)$ for a finite value of $y > 0$. Propositions 10.1, 10.2 and 10.3 imply that properties (1), (2), and respectively, (3) within two broadcasting round.

Propositions 11.1, 11.2 and 11.3 show that there is a nonzero probability that node p_i enters the relative state Allocated from either Ready or Obtaining within one broadcasting round. Thus, by the analyzing the expected time of the scheduler-luck games [16, 18], we have y has a finite value.

• **Closure** Suppose that $c_0^{\text{timeslot}}(x) \in R$ is a safe configuration and let $p_i \in P$ be any node. By the assumption that $c_0^{\text{timeslot}}(x)$, we have that p_i is in the relative state Allocated, i.e., properties (1) to (5) hold for any node p_i. We need to show that properties (1) to (5) holds in configuration $c_0^{\text{timeslot}}(x+1)$.

Propositions 10.1, 10.2 and 10.3 imply that properties (1), (2), and respectively, (3) (within one complete broadcasting round).

Properties (4) to (5) are implied by Proposition 11.3 and the fact that Properties (4) to (5) holds in $c_0^{\text{timeslot}}(x)$, i.e., $M(x) = \emptyset$. □

10 Properties (1) to (3)

Propositions 10.1, 10.2 and 10.3 imply that properties (1), (2), and respectively, (3) hold within two broadcasting rounds (or one complete broadcasting round). Let R be an execution of the MAC algorithm in Fig. 2, $x > 0$ an integer, and $c_0^{\text{timeslot}}(x)$ the first configuration in a complete broadcasting round $R(x) = (c_0^{\text{timeslot}}(x), \ldots c_{T-1}^{\text{carrier\_sense/receive}}(x))$. We note that $c_0^{\text{timeslot}}(x)$ follows an arbitrary starting configuration.

Proposition 10.1 shows that, within a broadcasting round from $c_0^{\text{timeslot}}(x)$, Property (1) holds.

Proposition 10.1. *In* $c_0^{\text{timeslot}}(x+1)$*, it holds that* $signal_i = \text{false}$.

Proof. The value of $signal_i$ is updated in line 18 (assigned to true) and in lines 12, 20, and 27 (assigned to false). Let us look into these assignments.

In every timeslot, the value false is assigned to $signal_i$ (cf. line 12). Suppose that the function send() is called, and thus, true is assigned to $signal_i$ (line 18). We proposition that before returning from the function send() and after true is assigned to $signal_i$ (line 18), node p_i must assign false to $signal_i$ either in line 20 or 27. To see that, let us look at lines 18 and 19. Eventually either $signal_i = \text{false}$ (because of an assignment in line 27) or $\rho(k) = \text{true}$ (line 19) holds (note the condition when $k = MaxRnd$). The latter case implies the execution of line 20. □

Proposition 10.2 shows that, within a broadcasting round from $c_0^{\text{timeslot}}(x)$, Property (2) holds.

Proposition 10.2. $(\exists t \in unused\_set_i \setminus \{s_i\}) \leftrightarrow (\nexists p_k \in \mathcal{N}_i : s_k = t)$ *in* $c_0^{\text{timeslot}}(x+1)$.

Proof. Recall that $unused\_set_i = \{k : unused_i[k] = \mathsf{true}\}$ (see line 6) and that the proposition statement does not consider the cases in which: (1) $s_i = s_k$ (because $t \neq s_i$) in $c_0^{\mathsf{timeslot}}(x+1)$, or (2) There exists a configuration $c \in R(x)$, such that $s_k \neq \perp$ in c and $s_k = \perp$ in $c_0^{\mathsf{timeslot}}(x+1)$ (because by *unused_set*'s definition, $\perp$ is never in $unused\_set_i$).

We note that in every broadcasting round, node $p_k \in P$ *at most once*: (1) Allocates the broadcasting timeslot s_k (when $t_k = 0$, see line 11), (2) Transmits a packet (when $t_k = s_k$, see line 13), and (3) Deallocates the broadcasting timeslot s_k (by assigning $\perp$ to s_k when $t_k = s_k$ and the $\mathsf{carrier\_sense}(t)$ event is raised, see line 26). Moreover, node p_i updates $unused_i[t]$ only in lines 12 (true) and 27 (false), when p_i removes stale information just before timeslot t, and respectively, when the event $\mathsf{carrier\_sense}(t)$ is raised.

Line 12 is executed at the start of every timeslot, whereas line 27 is executed after, and only when the event $\mathsf{carrier\_sense}(t)$ is raised. The event $\mathsf{carrier\_sense}(t)$ is raised after, and only when the node $p_k \in \mathcal{N}_i$ transmits in timeslot t. In other words, *none* of p_i's neighbors, $p_k \in \mathcal{N}_i$, that transmits in timeslot $s_k = t$, can avoid causing the event $\mathsf{carrier\_sense}(t)$ to be raised, and timeslot t to be included in $unused\_set_i \setminus \{s_i\}$. □

Proposition 10.3 shows that, within a broadcasting round from $c_0^{\mathsf{timeslot}}(x)$, Property (3) holds.

Proposition 10.3. $(s_i \neq \perp) \vee (unused\_set_i \setminus \{s_i\} \neq \emptyset)$ *holds in* $c_0^{\mathsf{timeslot}}(x+1)$.

Proof. If $s_i \neq \perp$ in $c_0^{\mathsf{timeslot}}(x+1)$, we are done. Let us suppose that $s_i = \perp$ in $c_0^{\mathsf{timeslot}}(x+1)$ and show that $unused\_set_i \setminus \{s_i\} \neq \emptyset$ in $c_0^{\mathsf{timeslot}}(x+1)$.

Let us assume, in the way of proof by contradiction that, $unused\_set_i \setminus \{s_i\} = \emptyset$ and show that $d_i/T > 1$, i.e., a contradiction with the assumption that $\forall p_i \in P : d_i/T \lesssim 1$.

Recall that $unused\_set_i = \{k : unused_i[k] = \mathsf{true}\} \subseteq [1,T]$ (see line 6). Therefore, the assumption that $s_i = \perp$ implies that $unused\_set_i = unused\_set_i \setminus \{s_i\} \subseteq [1,T]$, because by *unused_set*'s definition, $\perp$ is never in $unused\_set_i$.

By Proposition 10.2, we can say that $\forall t \in [1,T] : (\nexists t \in unused\_set_i) \leftrightarrow (\exists p_k \in \mathcal{N}_i : s_k = t)$. Since $unused\_set_i \subseteq [1,T]$, we can write $[1,T] \setminus unused\_set_i \subseteq \{s_k \in [1,T] : p_k \in \mathcal{N}_i\}$. By the fact that $unused\_set_i = \emptyset$, we have that $T \leq \mid \{s_k \in [1,T] : p_k \in \mathcal{N}_i\} \mid$. Since $d_i = \mid \mathcal{N}_i \mid$ (by definition), we have that $\mid \{s_k \in [1,T] : p_k \in \mathcal{N}_i\} \mid \leq d_i$, which implies $T \leq d_i$: a contradiction with the assumption that $d_i/T \lesssim 1$. □

11 Properties (4) to (5)

Section 10 of this Appendix shows that, starting from an arbitrary configuration, node $p_i \in P$ enters the relative state Ready within two broadcasting rounds. This section shows considers the probability for p_i to enter the relative states Obtaining and Allocated.

Let $x > 0$ and R be an execution of the MAC algorithm in Fig. 2. Suppose that $c_0^{\mathsf{timeslot}}(x)$ is the first configuration in a complete broadcasting round $R(x)$ for which properties (1) to (3) hold in configuration $c_0^{\mathsf{timeslot}}(x)$ with respect to node $p_i \in P$, i.e., p_i is in relative state Ready, Obtaining or Allocated. Propositions 11.1, 11.2 and 11.3 show that there is a nonzero probability that node p_i enters the relative state Allocated from either Ready or Obtaining in configuration $c_0^{\mathsf{timeslot}}(x+1)$.

Proposition 11.1 shows that p_i attempts to broadcast once in every round.

Proposition 11.1. *During broadcasting round* $R(x)$, p_i *executes line 13 and calls the function* $\mathsf{send}()$.

Proof. If $s_i \neq \perp$ in $c_0^{\mathsf{timeslot}}(x)$, we are done by lines 11 and 13. Let us consider the case of $s_i = \perp$ in $c_0^{\mathsf{timeslot}}(x)$. By Property (4), $unused\_set_i \neq \emptyset$ and thus when line 11 is executed, the function $\mathsf{select\_unused}()$ returns a non-$\perp$ element from $unused\_set_i$ and $s_i \neq \perp$ when executing line 13. □

Defining optimal transmission probabilities for any choices of T, n, d_i is not possible. We choose to consider and look for optimal choices when $d_i \simeq T$ (the 'hard' case) and make a case for a uniform probability $\rho_i = \frac{1}{n} : i \in [1,n]$.
Let us consider node $p_i \in P$ that competes, together with $k-1$ other neighbors, for the same unique timeslot. The probability that node p_i wins the (listening/signaling) competition is $\rho_1(1-\rho_1)^{k-1}$, where ρ_1 is the probability of choosing the first listening/signaling period. The value $\rho_1 = \frac{1}{k}$ maximizes this probability. In the more general case where there is more than one timeslot, we consider a strategy that aims at guessing the number, k, of competing neighbors, which the optimal probability of transmission depends on. During the first listening/signaling period, the strategy considers the case in which there are $n = MaxRnd$ singling nodes, and thus, the transmission probability is $1/MaxRnd$, where $MaxRnd \simeq T$. During the second listening/signaling period, the strategy considers the case in which there are $MaxRnd - 1$ neighbors, and thus, the transmission probability is $1/(MaxRnd-1)$, and so on. This *sequential* selection of the listening/signaling period leads to a uniform choice of a listening/signaling neighbor. The above strategy is driven by a heuristic in which nodes signal with probability that is optimal for the case of $n \simeq T$, and thus, it depends on the number of competing neighbors.

Figure 7: Transition probability, ρ_i, for listening/signaling periods (line 19 in Fig. 2)

Propositions 11.2 and 11.3 consider the set $M_i(x+1) = \{p_k \in \mathcal{N}_i : s_k = t'\}$ and the number $m_i = |M_i(x+1)|$ of p_i's neighbors that attempt to broadcast during p_i's timeslot, t', of broadcasting round $R(x)$.

Let ρ_j be the probability for p_i to transmit in the j-th listening/signaling period of timeslot t' (cf. line 19). This paper considers the concrete transmission probability $\rho_i = 1/MaxRnd$. We motivate our implementation choice of the transmission probability, ρ_i, in Fig. 7. Note that the *sequential* selection of the broadcasting rounds with probability $1/(MaxRnd-k+1)$ leads to the uniform selection $\rho_k = 1/MaxRnd$.

Proposition 11.2 considers p_i's chances to be the only one to transmit in its neighborhood.

Proposition 11.2. *There is a nonzero probability, $OnlyOne_i(x)$ (cf. equation (23)), that only node p_i transmits in its broadcasting timeslot, t', of broadcasting round $R(x)$.*

$$\begin{aligned} OnlyOne_i(x) \mid_{m_i>0} = \rho_1(1-\rho_1)^{m_i} + \rho_2(1-\rho_1-\rho_2)^{m_i} \\ + \ldots + \rho_{n-1}(1-\sum_{\ell=1}^{n-1}\rho_k)^{m_i} \end{aligned} \tag{23}$$

Proof. We show that there is a nonzero probability that only node p_i transmits in its broadcasting timeslot, t', of broadcasting round $R(x)$. Let us look at p_i and the nodes in $M_i(x)$ while they attempt to broadcast in the steps $a_i^{\mathsf{timeslot},t'}(x)$ and $a_k^{\mathsf{timeslot},t'}(x)_{|k\in M_i(x)}$. All of these steps include the execution of line 19, viz., each node chooses to transmit in listening/signaling period $\ell \in [0, MaxRnd]$ with probability $\rho_\ell = 1/(MaxRnd-\ell)$. Therefore, for any $MaxRnd > 0$, there is a nonzero probability, $OnlyOne_i(x)$ that, during timeslot t', node p_i transmits in the listening/signaling period $a \in MaxRnd$ and no node in $M_i(x)$ transmits in round a (or in an earlier one).

We note that the fact that p_i transmits first during timeslot t' implies that it is the only to transmit during t'. This is because once p_i transmits a BEACON in step $a_i^{\mathsf{timeslot},t'}(x)$ (which includes the execution of line 21) node $p_j \in \mathcal{N}_i \supseteq M_i(x)$ raises the event carrier_sense(t') immediately after $a_i^{\mathsf{timeslot},t'}(x)$. Thus, $\forall p_j \in M_i(x)$ we have that immediately after step $a_i^{\mathsf{timeslot},t'}(x)$, node p_j takes step $a_j^{\mathsf{carrier\_sense},t'}(x)$, which includes the execution

of lines 26 and 27 that assigns $\perp$ to s_j and $false$ to $signal_j$. Thus, p_j leaves the (listening/signaling) competition for timeslot t' (see line 18) and does not transmits its DATA packet (see line 23).

We now turn to calculate $OnlyOne_i(x)$. Let the variable $m_i = \mid M_i(x) \mid$ denote the number of nodes that select the same timeslot as p_i in configuration $c_0^{\text{timeslot: s}\neq\perp}(x)$. The value of $OnlyOne_i(x)$ depends on the value of m_i and we denote this dependence with the notation $q(i) \mid_{m_i}$ (conditional probability). It means the value of $OnlyOne_i(x)$ depends on the value of m_i. The value of $OnlyOne_i(x)$ for $m_i = 0$ is $OnlyOne_i(x) \mid_{m_i=0} = 1$. For the case of $m_i > 0$, $OnlyOne_i(x)$'s value is given by equation (23)) (that appears again below), where ρ_j is the probability for transmitting in the j-th listening/signaling period.

$$OnlyOne_i(x) \mid_{m_i>0} = \rho_1(1-\rho_1)^{m_i} + \rho_2(1-\rho_1-\rho_2)^{m_i}$$
$$+ \ldots + \rho_{n-1}(1-\textstyle\sum_{\ell=1}^{n-1}\rho_k)^{m_i} \quad \text{[clone of equation (23)]}$$

We note that the j-th term in equation (23), is the probability that node p_i selects the j-th listening/signaling period and all its neighbors select a later listening/signaling period. □

Proposition 11.3 shows that once a node is the only one in its neighborhood to transmit during its broadcasting timeslot, it enters the relative state Allocated.

Proposition 11.3. *$M_i(x) = \emptyset$ (or having none of the nodes in $M_i(x)$ transmitting during timeslot t') implies that node p_i is in the relative state Allocated in $c_0^{timeslot}(x+1)$.*

Proof. We need to show that, in $c_0^{\text{timeslot}}(x+1)$, we have that $s_i = t' \neq \perp$ and $\forall p_j \in \mathcal{N}_i : s_i \neq s_j$.
Showing that $s_i = t' \neq \perp$ in $c_0^{\text{timeslot}}(x+1)$ The proposition assumes that $t' \neq \perp$ in $c_0^{\text{timeslot}}(x)$. We wish to show that $s_i = t'$ in $c_0^{\text{timeslot}}(x+1)$, which implies that $s_i \neq \perp$ holds in $c_0^{\text{timeslot}}(x+1)$ and throughout $R(x+1)$.

Since the variable s_i is assigned only in lines 11 (when $t_i = 0$) and 26 (when $t_i = t'$), it is sufficient to show that line 26 is not executed by any step during timeslot t' of broadcasting round $R(x)$, i.e., $a_i^{\text{carrier\_sense},t'}(x) \notin R(x)$.

Node p_i raises the event carrier_sense only during timeslots in which p_i's neighbor, p_j, transmits. By the proposition assumptions that, during timeslot t' of broadcasting round $R(x)$, none of p_i's neighbors transmits, we have $a_i^{\text{carrier\_sense},t'}(x) \notin R(x)$. Moreover, $a_i^{\text{timeslot},t'}(x+1)$ does not includes an execution of line 11 that changes the value of s_i, because $s_i = t' \neq \perp$ in $c_0^{\text{timeslot}}(x+1)$.
Showing that $\forall p_j \in \mathcal{N}_i : s_i \neq s_j$ in $c_0^{\text{timeslot}}(x+1)$ The proposition assumes that $\forall p_j \in \mathcal{N}_i : s_i \neq s_j$ in $c_0^{\text{timeslot}}(x)$. We wish to show that the same holds in $c_0^{\text{timeslot}}(x+1)$. Since the variable s_j is assigned to a non-$\perp$ value only in line 11 when $t_i = 0$, it is sufficient to show that when line 11 is executed in step $a_j^{\text{timeslot},0}(x+1)$ the function select_unused() considers a set that does not includes p_i's timeslot, s_i. This is implied by the facts that $\forall p_j \in \mathcal{N}_i : unused_j[t'] = \text{false}$ (Claim 11.1) and $s_i = t'$ (first item of (II) of this proof) in $c_0^{\text{timeslot}}(x+1)$. □

12 Bounding $OnlyOne_i(x)$

Propositions 5.3 and 12.2 bound $OnlyOne_i(x)$'s value, where $R(x)$ is the x-th broadcasting round in execution R of the MAC algorithm in Fig. 2. We assume that properties (1) to (5) holds in the first configuration, $c_0^{\text{timeslot}}(x)$, of $R(x)$. These bounds are obtained by computing the expectation of $q_i \mid_{m_i}$ with respect to m_i, where $M_i(x) = \{p_k \in \mathcal{N}_i : s_k = t'\}$ in $c_0^{\text{timeslot}}(x)$ and $m_i = |M_i(x)|$. The reason is that m_i is a random variable, i.e., $q_i = E\left(OnlyOne_i(x) \mid_{m_i}\right)$, where the expectation is computed with respect to the random variable m_i.

We note that all the terms in equation (23) are convex functions of m_i. This means that by Jensen's inequality, we obtain a lower bound of q_i in equation (24) by evaluating the expression $q_i\mid_{m_i}$ at m_i's expectation, $E(m_i)$ [26].

$$q_i \quad = \quad E\left(q_i\mid_{m_i}\right) \geq q_i\mid_{E(m_i)} \tag{24}$$

The expression on the right side of the inequality can be again lower bounded if we estimate an upper bound for $E(m_i)$. We proceed to the computations in the proof of the Proposition 12.2 after demonstrating Proposition 12.1 which shows that $E(m_i)$ is bounded by the ratio d_i/T, which is rather intuitive but, needs to be proved.

Proposition 12.1. *In configuration $c_0^{\text{timeslot}}(x)$ in holds that $E(m_i) \leq d_i/T$, where $m_i = |M_i(x)|$.*

Proof. We show that $E(m_i) = d_i/T$ by considering configuration $c_0^{\text{timeslot}}(x)$. The maximal number of p_i's neighbors that might choose the same timeslot as p_i in configuration $c_0^{\text{timeslot}}(x)$ is $\sum_{p_j\in\mathcal{N}_i} 1_{\{s_j=\perp\}}$, because any node, $p_j \in \mathcal{N}_i$, that chooses a new broadcasting timeslot immediately before $c_0^{\text{timeslot}}(x)$ must have $s_j = \perp$ in configuration $c_0^{\text{timeslot}}(x)$. We compute the expected value of m_i in equation (25) as a function of the number of empty timeslots, e_i, that p_i selects from when choosing a new broadcasting timeslot, where $e_i = |\ \mathit{unused\_set}_i\ |$ in configuration $c_0^{\text{timeslot}}(x)$.

$$\begin{aligned} E(m_i) = & \\ \sum_{t\in E_i} E(m_i \mid s_i = t)\Pr(s_i = t) = & \\ \sum_{t\in E_i} \frac{1}{e_i} E(m_i \mid s_i = t) = & \\ \sum_{t\in E_i} \frac{1}{e_i} E\left(\sum_{p_j\in\mathcal{N}_i} 1_{\{p_j \text{ chooses timeslot } t\}} \mid s_i = t\right) = & \\ \sum_{t\in E_i} \frac{1}{e_i} \sum_{p_j\in\mathcal{N}_i} \frac{1}{|\ E_j\ |} 1_{\{t\in E_j\}} 1_{\{s_j=\perp\}} & \end{aligned} \tag{25}$$

Our assumption that $d_i \leq T-1$ implies that $e_i > 0$. Using that $d_i = \sum_{p_j\in\mathcal{N}_i}\left(1_{\{s_j\neq\perp\}} + 1_{\{s_j=\perp\}}\right)$ and, $e_i \geq T - \sum_{p_j\in\mathcal{N}_i} 1_{\{s_j\neq\perp\}}$, we obtain equation (26).

$$\begin{aligned} E(m_i) \leq & \sum_{t\in E_i} \frac{1}{T - d_i + \sum_{p_j\in\mathcal{N}_i} 1_{\{s_j=\perp\}}} \sum_{p_j\in\mathcal{N}_i} \frac{1_{\{t\in E_j\}} 1_{\{s_j=\perp\}}}{|\ E_j\ |} \\ = & \frac{1}{T - d_i + \sum_{p_j\in\mathcal{N}_i} 1_{\{s_j=\perp\}}} \sum_{p_j\in\mathcal{N}_i} \frac{1_{\{s_j=\perp\}}}{|\ E_j\ |} \underbrace{\sum_{t\in E_i} 1_{\{t\in E_j\}}}_{|E_i\bigcap E_j|} \\ \leq & \frac{\sum_{p_j\in\mathcal{N}_i} 1_{\{s_j=\perp\}}}{T - d_i + \sum_{p_j\in\mathcal{N}_i} 1_{\{s_j=\perp\}}} \leq \frac{d_i}{T} \end{aligned} \tag{26}$$

□

Proposition 12.2.

$$q_i \geq \sum_{k=1}^{n} \rho_k \left(1 - \sum_{\ell=1}^{k} \rho_k\right)^{\frac{d_i}{T}}$$
[clone of equation (6)]

Proof. Proposition 12.1 shows that $E(m_i) \leq d_i/T$. The proposition is demonstrated by evaluating expression (23) at $E(m_i) = d_i/T$, see equation (24). □

Proposition 5.3 considers the concrete transmission probability $\rho_i = 1/MaxRnd$.
Proposition 5.3 *Let* $\rho_i = 1/MaxRnd$. *Equation* (16) *bounds from below the probability* q_i.

Proof. In this proof, we use the letter n instead of $MaxRnd$ for reason of space. We replace ρ_i with $1/n$ in equation (6) to obtain equation (27).

$$q_i \geq \sum_{k=1}^{n} \frac{1}{n} \left(1 - \frac{k}{n}\right)^{\frac{d_i}{T}} \tag{27}$$

Equation (28) is more compact than equation (27) and it is obtained by the fact that the function $(1-x)^s$ is convex.

$$\begin{aligned} q_i \geq & \\ \sum_{k=1}^{n} \frac{1}{n} \left(1 - \frac{k}{n}\right)^{\frac{d_i}{T}} = & \\ \frac{1}{2n} \sum_{k=1}^{n} \left[\left(1 - \frac{k}{n}\right)^{\frac{d_i}{T}} + \left(1 - \frac{n-k+1}{n}\right)^{\frac{d_i}{T}} \right] \geq & \\ \text{(convexity)} \ \frac{1}{n} \sum_{k=1}^{n} \left(1 - \frac{n+1}{2n}\right)^{\frac{d_i}{T}} = & \\ \left(1 - \frac{n+1}{2n}\right)^{\frac{d_i}{T}} & \end{aligned} \tag{28}$$

Another way to bound equation (27) is by considering the decreasing function $y \to (1-y)^x$, as in equation (29).

$$\begin{aligned} q_i \geq \sum_{k=1}^{n} \frac{1}{n} \left(1 - \frac{j}{n}\right)^{\frac{d_i}{T}} \geq \int_{\frac{1}{n}}^{1} (1-y)^{\frac{d_i}{T}} \, dy & \\ = \frac{1}{\frac{d_i}{T}+1} \left(1 - \frac{1}{n}\right)^{\frac{d_i}{T}+1} & \end{aligned} \tag{29}$$

□

13 Theorem 6.2

Theorem 6.2 bounds the system convergence time.

Theorem 6.2 (Global Convergence) *The expected number of retransmissions, is smaller than* $\left(\frac{2n}{n-1}\right)^{d/T} - 1$, *where* $d = \max_i\{d_i : d_i \in P\}$. *Hence, we have that the expected number of broadcasting rounds,* $\mathcal{S}$, *that guarantee that all nodes to reach the relative state* Allocated *satisfies equation* (21).

$$\mathcal{S} \leq \left(\frac{2n}{n-1}\right)^{d/T} \quad \text{[clone of equation (21)]}$$

Moreover, given that there are N nodes in the network and $\alpha \in (0,1)$, *the network convergence time is bounded by equation* (22) *with probability* $1-\alpha$.

$$k = 1 + \frac{log\left(1 - \sqrt[N]{1-\alpha}\right)}{log\left(1 - \left(\frac{n-1}{2n}\right)^{\frac{d}{T}}\right)} \quad \text{[clone of equation (22)]}$$

This means that with probability α *all nodes are allocated with timeslots in maximum k broadcasting rounds, see Fig.* (6).

Proof. Theorem 5.1 bounds the convergence time of a particular processor, see equation (7). Lemma 6.1, see equation (20) $E(W) \geq N(\frac{n-1}{2n})^{x/T}$, proves that this bound is still valid if we replace the term d_i/T with x/T, i.e., we consider the average degree instead of the particular degree of a node. If we replace x/T by $\max\{d_i\}/T$ in expression (20) we obtain a larger bound because $x/T \leq \max\{d_i\}/T$, i.e. $E(W) \geq N(\frac{n-1}{2n})^{x/T} \geq N(\frac{n-1}{2n})^{\max\{d_i\}/T}$.

The bound $E(W) \geq N(\frac{n-1}{2n})^{\max\{d_i\}/T}$ and the discussion in the 1st paragraph of section 6, show that the number of processors that are allocated during a broadcasting round is bounded by the random variable $\sum_{i=1}^{N} z_i$, where z_i are identically and independently distributed random variables that are 1 with probability $(\frac{n-1}{2n})^{\max\{d_i\}/T}$ and 0 with probability $1-(\frac{n-1}{2n})^{\max\{d_i\}/T}$ (the second random variable dominate the first one, see [37]). This means that we lower bound the number of processors that are allocated if we consider that they are allocated independently with probability $(\frac{n-1}{2n})^{\max\{d_i\}/T}$.

While the processors get allocated to a timeslot, the parameters d_i and T change because some timeslots are no longer available (T decreases and some nodes are allocated d_i decreases). Actually the ratio becomes $\frac{\max\{d_i\}-h_i}{T-f_i}$, where $h_i \geq f_i$ because if a timeslot is allocated or sensed used by processor p_i then T, the number of available timeslots decreases by 1 and d_i, the number of competing nodes, must decrease at least by one since there must be at least one processor that uses the busy timeslot (there may be multiple that are in state Obtaining). Under these circumstances we always have $\frac{\max\{d_i\}}{T} \geq \frac{\max\{d_i\}-h_i}{T-f_i}$. Thus, we can obtain a lower bound for the expected time to reach the relative state Allocated by assuming that all nodes are allocated independently with probability $x = (\frac{n-1}{2n})^{\max\{d_i\}/T}$. We simplify the following arguments by using this define of x.

To bound the number of broadcasting rounds we consider the following game. The bank pays 1 unit to the nodes that get in state Allocated (get allocated to a timeslot), and receives $x/(1-x)$ units per nodes that fails to get in state Allocated. The game is fair because in each round the expected gain is $1 \times x - x/(1-x) \times (1-x) = 0$. If we denote by W_i the number of processors that get in state Allocated during the i-th broadcasting round and by L_i the number of processors that fail we have that the gain is given by

equation 30, where t denotes the total number of rounds.

$$\text{gain} = \sum_{i=1}^{t} \left(\frac{x}{1-x} L_i - W_i \right) \tag{30}$$

The expected gain is 0 because the game is fair ($E(gain) = 0$) and $\sum_{i=1}^{t} W_i = N$ because eventually all the nodes get in state Allocated and the bank pays 1 unit for each such processors. If we compute the expectation on both sides of equation (30), we then obtain equation 31.

$$N = \frac{x}{1-x} E\left(\sum_{i=1}^{t} L_i \right) \tag{31}$$

We observe that $E(\sum_{i=1}^{t} L_i)$ is the expected total number of retransmissions and $E(\sum_{i=1}^{t} L_i)/N$ is the average expected number of retransmissions whose value is $(1-x)/x$. replacing x with its expression, we obtain that the average number of retransmission is bounded by $(2n/(n-1))^{\max\{d_i\}/T} - 1$ and, this leads to the bound equation (21).

To prove the second assertion, let $t_1, \ldots, t_N$ be the convergence time of nodes $1, \ldots N$, respectively. The random variables, t_i, are bound by random variables with geometric random distribution with expectation of $(2n/(n-1))^{d/T}$, with $d = max\{d_i : d_i \in P\}$. We require that $t_{\max} = \max\{t_1, \ldots, t_N\}$ in order to ensure that all nodes are allocated with timeslots. The fact that the random variables, t_i, are independent and identically distributed, implies equation (32), where t is a random geometrical random variable, i.e., $\Pr(t = k') = (1-q)^{k'-1} q$ and $\Pr(t \geq k') = (1-q)^{k'-1}$.

$$\begin{aligned} &\Pr\left(t_{\max} \leq k'\right) = P\left(t_1 \leq k', \ldots, t_N \leq k'\right) = \\ &\Pr\left(t_1 \leq k'\right) \cdot \ldots \cdot P\left(t_N \leq k'\right) = P\left(t \leq k'\right)^N \end{aligned} \tag{32}$$

Which $t_{\max} \leq k'$ satisfies equation (33) with probability α?

$$\begin{aligned} \Pr(t_{\max} < k') = \Pr\left(t < k'\right)^N = \\ \left(1 - (1-q)^{k'-1}\right)^N \geq 1 - \alpha \end{aligned} \tag{33}$$

By solving equation (33), we observe that equation (33) is satisfied for any $k' \geq k$, where k satisfies equation (22). This proves that, with probability $1 - \alpha$, the network convergence time is bounded by equation (22). □

A.1.6 Autonomous TDMA alignment for VANETs

"Autonomous TDMA alignment for VANETs". Mohamed Hassan Mustafa, Marina Papatriantafilou, Elad M. Schiller, Amir Tohidi, and Philippas Tsigas, In IEEE 76th Vehicular Technology Conference (VTC'12-Fall), 2012.

This page is intentionally left blank.

Autonomous TDMA Alignment for VANETs *

Mohamed Mustafa Marina Papatriantafilou Elad M. Schiller Amir Tohidi Philippas Tsigas
Chalmers University of Technology, Sweden {mohmus, ptrianta, elad, tohidi, tsigas}@chalmers.se

***Abstract*—The problem of local clock synchronization is studied in the context of media access control (MAC) protocols, such as time division multiple access (TDMA), for dynamic and wireless ad hoc networks. In the context of TDMA, local pulse synchronization mechanisms let neighboring nodes align the timing of their packet transmissions, and by that avoid transmission interferences between consecutive timeslots. Existing implementations for Vehicular Ad-Hoc Networks (VANETs) assume the availability of common (external) sources of time, such as base-stations or geographical positioning systems (GPS). This work is the first to consider autonomic design criteria, which are imperative when no common time sources are available, or preferred not to be used, due to their cost and signal loss. We present self-$\star$ pulse synchronization strategies. Their implementing algorithms consider the effects of communication delays and transmission interferences. We demonstrate the algorithms via extensive simulations in different settings including node mobility. We also validate these simulations in the MicaZ platform, whose native clocks are driven by inexpensive crystal oscillators. The results imply that the studied algorithms can facilitate autonomous TDMA protocols for VANETs.**

I. Introduction

Recent work on vehicular systems explores a promising future for vehicular communications. They consider innovative applications that reduce road fatalities, lead to greener transportation, and improve the driving experience, to name a few. The prospects of these applications depend on the existence of predictable communication infrastructure for dynamic networks. We consider time division multiple access (TDMA) protocols that can divide the radio time regularly and fairly in the presence of node mobility, such as Chameleon-MAC [8]. The studied problem appears when neighboring nodes start their broadcasting timeslots at different times. It is imperative to employ autonomous solutions for timeslot alignment when no common (external) time sources are available, or preferred not to be used, due to their cost and signal loss. We address the timeslot alignment problem by considering the more general problem of *(decentralized) local pulse synchronization*. Since TDMA alignment is required during the period in which communication links are being established, we consider non-deterministic communication delays, the effect of transmission interferences and local clocks with arbitrary initial offsets, see Section II. We propose autonomous and self-$\star$ algorithmic solutions that guarantee robustness and provide an important level of abstraction as they liberate the system designer from dealing with low-level problems, such as availability and cost of common time sources, see Section III. Our contribution also facilitates autonomous TDMA protocols for Vehicular Ad-Hoc Networks (VANETs), see Section IV.

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

Let us illustrate the problem and the challenges of possible strategies using an example. Consider three neighboring stations that have unique timeslot assignment, but their timeslots are not well-aligned, see Fig. 1. Packet transmissions collide in the presence of such concurrent transmissions. Suppose that the stations act upon the intuition that gradual pairwise adjustments are most preferable. Station p_k is the first to align itself with its closest neighbor, p_j, see Fig. 2. Next, p_j aligns itself with p_i and by that it opens a gap between itself and p_k. Then, p_k aligns itself with p_i and p_j. The end result is an all aligned sequence of timeslots. We call this algorithmic approach the *cricket* strategy.

Observe that the convergence process includes chain reactions, i.e., node p_k aligns itself before and after p_j's alignment. One can foresee the outcome of such chain reactions and let p_j and p_k to concurrently adjust their clock according to p_i. This algorithmic approach, named the *grasshopper*, is faster than the cricket, see Section IV. This improvement comes at the cost of additional memory and processing requirements. We integrate the proposed algorithms with the Chameleon-MAC [8], which is a self-$\star$, mobility resilient, TDMA protocol. After extensive simulations with and without mobility, we observe tight alignment among the timeslots, and high MAC throughput. Additional testbed experiments appear in [12].

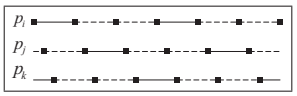


Fig. 1. Unaligned TDMA timeslots. Solid and dashed lines stand for transmission, and respectively, idle radio times.

Biologically-inspired synchronization mechanisms are proposed in [3, 10, 11]. They, and others such as [14], do not consider wireless communication environments with communication delays or disruptions. More practical communication environments are considered in [5, 15, 16], but they do not have TDMA MAC in mind. In [5], Byzantine-tolerance and self-stabilization properties are considered after communication establishment. We are the first to consider TDMA timeslot alignment during the period in which communication links are being established.

II. Preliminaries

The system consists of a set, $N = \{p_i\}$, of n anonymous communicating entities, which we call *nodes*. The radio time is divided into fixed size TDMA frames and then into fixed size timeslots [as in 8]. The nodes' task is to adjust their local clocks so that the starting time of frames and timeslots is aligned. They are to achieve this task in the presence of: (1)

978-1-4673-1881-5/12/$31.00 ©2012 IEEE

a MAC layer that is in the process of assigning timeslots, (2) network topology changes, and (3) message omission, say, due to topological changes, transmission interferences, unexpected change of the ambient noise level, etc.

Time, clocks, and synchrony bounds We consider three notations of time: *real time* is the usual physical notion of continuous time, used for definition and analysis only; *native time* is obtained from a native clock, implemented by the operating system from hardware counters; *local time* builds on native time with an additive adjustment factor in an effort to facilitate a neighborhood-wise clock. Applications require the clock interface to include the READ operation, which returns a *timestamp* value of the local clock. Let C_k^i and c_k^i denote the value $p_i \in N$ gets from the k^{th} READ of the native or local clock, respectively. Moreover, let r_k^i denote the real-time instance associated with that k^{th} READ operation. Pulse synchronization algorithms adjust their local clocks in order to achieve synchronization, but never adjust their native clocks. Namely, the operation ADJUST(add) adds a positive integer value to the local clock. This work considers solutions that adjust clocks forward, because such solutions simplify the reasoning about time at the higher layers. We define the native clocks *offset* $\delta_{i,j}(k,q) = C_k^i - C_q^j$, and the local clocks offset $\Lambda_{i,j}(k,q) = c_k^i - c_q^j$; where $\Delta_{i,j}(k,q) = r_k^i - r_q^j = 0$. Given a real-time instance t, we define the *(local clock) synchrony bound* $\psi(t) = \max(\{\Lambda_{i,j}(k,q) : p_i, p_j \in N \wedge \Delta_{i,j}(k,q) = 0\})$ as the maximal clock offset among the system nodes.

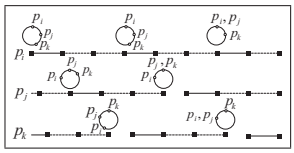


Fig. 2. The cricket strategy. Solid and dashed lines stand for transmission, and respectively, idle radio times. The circles above the solid boxes represents the node's view on its neighbors' TDMA alignment at the start of its broadcasting timeslot. Gaps between two solid boxes represent alignment events.

One may consider p_i's *(clock) skew*, $\rho_i = \lim_{\Delta_{i,i}(k,q)\to 0} \delta_{i,i}(k,q)/\Delta_{i,i}(k,q) \in [\rho_{\min}, \rho_{\max}]$, where $\rho_{\min}$ and $\rho_{\max}$ are known constants [4, 6]. The clock skew of MicaZ nodes is bounded by a constant that is significantly smaller than the communication delays. Therefore, our simulations assume a zero skew. We validate these simulations in the MicaZ platform.

Pulses Each node has hardware supported timer for generating *(periodic) pulses* every P (phase) time units. Denote by $c_{q_k}^i$ the $k-th$ time in which node p_i's timer triggers a pulse, immediately after performing the READ operation for the $q_k - th$ time. The term *timeslot* refers to the period between two consecutive pulses at times $c_{q_k}^i$ and $c_{q_{k+1}}^i$. We say that $t_i = c_{q_k}^i \bmod P$ is p_i's *(pulse) phase* value. Namely, whenever $t_i = 0$, node p_i raises the event $timeslot(s_i)$, where $s_i = k \bmod T$ is p_i's (broadcasting) timeslot number and $T > 1$ is the *TDMA frame size*.

The MAC layer The studied algorithms use packet transmission schemes that employ communication operations for receiving, transmitting and carrier sensing. Our implementation considers merely the latter two operations, as in the Beeps model [2], which also considers the period prior to communication establishment. We denote the operations' time notation (timestamp) in the format $\langle timeslot, phase\rangle$, where $timeslot \in [0, T-1]$ and $phase \in [0, P-1]$. We assume the existence of efficient mechanisms for timestamping packets at the MAC layer that are executed by the transmission operations, as in [4]. We assume the existence of an efficient upper-bound, $\alpha \ll P$, on the communication delay between two neighbors, that, in this work, has no characterized and known distribution.

Task definition The problem of *(decentralized) local pulse synchronization* considers the rapid reduction of all *local synchrony bounds* $\psi \geq \max(\{\Lambda_{i,j}(k,q) : p_i, p_j \in N \wedge p_j \in \mathcal{N}_i^T \wedge \Delta_{i,j}(k,q) = 0\})$, where $\mathcal{N}_i^T$ refers to p_i's recent neighbors, see Fig. 3 for definition. Given the synchrony bound $\psi \geq 0$, we look at the *convergence (rate bound)*, ℓ_ψ, which is the number of TDMA frames it takes to reach ψ. Recall that we consider only forward clock adjustments. We also study local pulse synchronization's relation to MAC-layer, network scalability and topological changes.

III. Pulse Synchronization Strategies

Pulse synchronization solutions require many considerations, e.g., non-deterministic delays and transmission interferences. Before addressing the implementation details, we simplify the presentation by first presenting *(algorithmic) strategies* in which the nodes learn about their neighbors' clock values without delays and interferences.

We present two strategies that align the TDMA timeslots by calling the function ADJUST(aim) immediately before their broadcasting timeslot, see Fig. 2. The first strategy, named *Cricket*, sets aim's value according to neighbors that have the most similar phase values. The second strategy, named *Grasshopper*, looks into a greater set of neighbors before deciding on aim's value. Both strategies are based on the relations among nodes' phase values, see Fig. 3 for definitions.

Cricket strategy This strategy acts upon the intuition that gradual pairwise adjustments are most preferable. Node p_i raises the event timeslot(s_i), when $t_i = 0$, and adjusts its local clock according to Eq. (1). At this time, $PhaseOrder_{\gamma_i}$'s first item has zero value, because it refers to p_i's own pulse, the second item refers to p_i's successor and the last item refers to p_i's predecessor.

$$aim_{\gamma_i} = \begin{cases} head_{\gamma_i} & : head_{\gamma_i} < tail_{\gamma_i} \quad \text{JUMP} \\ 0 & : head_{\gamma_i} > tail_{\gamma_i} \quad \text{WAIT} \\ head_{\gamma_i} \text{ or } 0; \text{ each with probability } \frac{1}{2} & : head_{\gamma_i} = tail_{\gamma_i} \quad \text{MIX} \end{cases} \tag{1}$$

The cricket strategy considers both pure deterministic actions (JUMP and WAIT) and a non-deterministic one (MIX).

- JUMP**:** Whenever node p_i is closer to its predecessor than to its successor ($head_{\gamma_i} < tail_{\gamma_i}$), it catches up with its predecessor by adding $head_{\gamma_i}$ to its clock value, which is the phase difference between itself and its predecessor.
- WAIT**:** Whenever p_i is closer to its successor than to its predecessor ($head_{\gamma_i} > tail_{\gamma_i}$), p_i simply waits for its successor.
- MIX**:** Node p_i breaks symmetry whenever it is as close to its predecessor as it is to its successor ($head_{\gamma_i} = tail_{\gamma_i}$). In

Learning about neighbors' clock values At any real-time instance t, p_i's *reach set*, $R_i(t) = \{p_j\} \subseteq N$, represents the set of nodes, p_j, that receive p_i's transmissions. At the MAC layer, the real-time instance t refers to the time in which p_j raises the carrier sense event. The set *recent neighbors*, $\mathcal{N}_i^T = \{p_j \in N : \text{starting-time}(s_j) \in [t, t'] \wedge p_i \in R_j(t(s_j))\}$, refers to nodes whose broadcast in timeslot s_j, arrive to node p_i, where t is a real-time instance that happens T timeslots before the real-time instance t' and starting-time$(s_j) \in [t, t']$ refers to the starting time of p_j's timeslot.

Locally observed pulse profiles Given a real-time instance t and node $p_i \in N$, we denote the *locally observed pulse profile* by $\gamma_i(t) = (\langle s_j, t_j \rangle)_{p_j \in \mathcal{N}_i^T}$, as a list of p_i's recently observed timestamps during the passed T timeslots before t. We sometimes write γ_i, rather than $\gamma_i(t)$, when t refers to the starting time of p_i's timeslot.

Phase orders Let $Order = (p_{i_k})_{k=0}^{n-1}$ be an ordered list of nodes in N, where p_i's predecessor and successor in N are $p_{i_{k-1 \bmod n}}$, and respectively, $p_{i_{k+1 \bmod n}}$. The ordered list, $PhaseOrder_{\gamma_i}$, of the pulse profile, γ_i, is sorted by the phase field of γ_i's timestamp $\langle timeslot_j, phase_j \rangle \in \gamma_i$.

Predecessors, successors, heads, and tails Given a node, p_i, and its view on the pulse profile, γ_i, define the $predecessor_i$ and the $successor_i$ as p_i's predecessor, and respectively, successor in $PhaseOrder_{\gamma_i}$. Moreover, $head_{\gamma_i} = (t_i - t_{pr}) \bmod P$ and $tail_{\gamma_i} = (t_{su} - t_i) \bmod P$ is the phase difference between p_i's phase value, t_i and $predecessor_i = p_{pr}$, and respectively, $successor_i = p_{su}$. These imply that $predecessor_i$ is pulsed $head_{\gamma_i}$ time units before node p_i and $successor_i$ is pulsed $tail_{\gamma_i}$ time units after p_i.

Fig. 3. Pulse profiles and the relations among nodes' phase values

Set up: Given that the broadcasting schedule of nodes $N = \{p_1, \ldots, p_4\}$ is by their index value, the pulse profiles, $\{\gamma_i\}_{p_i \in N}$, encode pulses such that initially p_1 and p_3 have local dominant pulses.
Convergence: In the first TDMA frame, nodes $N(1) = \{p_2, p_4\}$ converge towards their respective local dominant pulses in $p_y \in \{p_1, p_3\}$. In the second TDMA frame, the local dominant pulses are nodes $p_y \in \{p_1, p_2\}$. Note that node p_3's pulse is no longer a local dominant and it adjusts its phase according to Eq. (1), i.e., $N(2) = \{p_3, p_4\}$

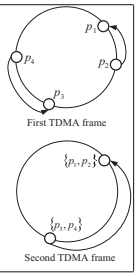


Fig. 4. Cricket strategy convergence during the first two TDMA frames.

this case, p_i randomly chooses between JUMP and WAIT.

Local dominant pulses Let us look into a typical convergence of the cricket strategy, see Fig. 4. Given two nodes, $p_i, p_j \in N$, and p_i's locally observed pulse profile, $\gamma_i(t)$, we say that p_j's pulse (phase value) *locally dominates* the one of p_i, if $head_{\gamma_i} < tail_{\gamma_i}$ and p_j is p_i's predecessor in γ_i. Observe that clock updates can result in a chain reaction, see Fig. 4. Lengthy chain reactions can prolong the convergence up to $\mathcal{O}(n)$ TDMA frames, see Fig. 5.

Global dominant pulses In Fig. 5, all nodes eventually align their timeslots with the one of p_1, because p_1's pulse immediately follows the maximal gap in γ_i. Pulse gaps provide

Set up: For $\xi > 0$ and $p_i \in N$, the pulse profiles $\{\gamma_i\}_{p_i \in N}$ encode pulses in which node p_{i+1}'s pulse occurs $(n-i)\xi$ clock units after p_i's pulse. **Convergence:** In the first TDMA frame, only node p_n can take JUMP action to align with its neighbor local clock, p_{n-1}, because its $head_{\gamma_n} < tail_{\gamma_n}$. In the second TDMA frame, nodes $N(2) = \{p_n, p_{n-1}\}$ adjust their clocks to be aligned with p_{n-2}. Thus, in the $(n-1)$-th TDMA frame, nodes $N(n-1) = \{p_n, p_{n-1}, \ldots, p_2\}$ align with node p_1. Therefore, $n-1$ TDMA frames are needed before convergence.

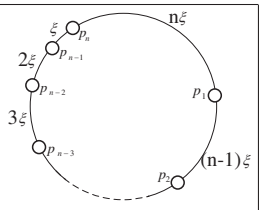


Fig. 5. Chain reactions of pulse updates: An example with $\mathcal{O}(n)$ TDMA frames before convergence.

useful insights into the cricket strategy convergence. Given node $p_i \in N$, its pulse profile γ_i and $k \in [1, |\mathcal{N}_i^T|]$, we obtain the *(pulse) gaps* between γ_i's consecutive pulses, $Gap_{\gamma_i}(k) = (PhaseOrder_{\gamma_i}[k].phase - PhaseOrder_{\gamma_i}[k-1].phase)$, see Fig. 3 for definitions. For the case of $k = 0$, we define $Gap_{\gamma_i}(0) = (P - PhaseOrder_{\gamma_i}[|\mathcal{N}_i^T|].phase)$. The set, $MaxGap_{\gamma_i}$, of pulses that immediately follow the maximal gap in γ_i are named *global dominates*.

$$MaxGap_{\gamma_i} = \underset{k \in [0, |\mathcal{N}_i^T|]}{\operatorname{argmax}} (Gap_{\gamma_i}(k)) \tag{2}$$

Given three nodes, $p_i, p_j, p_\ell \in N$, p_i's locally observed pulse profile, $\gamma_i(t)$, $j \in MaxGap_{\gamma_i}$ and $i, \ell \notin MaxGap_{\gamma_i}$, we say that p_j's pulse *globally dominates the one of* p_i, if at least one of the following holds: (1) $i = j$ (2) p_j's pulse locally dominates the one of p_i, or (3) p_j's pulse globally dominates the one of p_ℓ and p_ℓ's pulse locally dominates the one of p_i. We define p_i's clock offset towards its preceding global dominant pulse as $DomPulse_i = P - PhaseOrder_{\gamma_i}[k].phase$, where $k \in MaxGap_{\gamma_i}$ is p_i's global dominant pulse, see Eq. (2).

We define $OneGlobal(\gamma_i) = (|MaxGap_{\gamma_i}| = 1)$ to be true whenever γ_i encodes a single global dominant pulse. For the cases in which there is more than one, we define the term next (global) dominant pulse for node p_i, where $i \in MaxGap_{\gamma_i}$ refer to p_i's global dominant pulse. In this case, p_i's *next (global) dominant pulse*, $NextDomPulse_i = DomPulse_{pr}$, is p_i's predecessor's global dominant pulse, where $predecessor_i = p_{pr}$.

Next we present the grasshopper strategy, which uses the notion of global dominant pulses to avoid lengthy chain reactions in order to achieve a faster convergence.

Grasshopper strategy This strategy is based on the ability to see beyond the immediate predecessor and local dominant pulses. The nodes converge by adjusting their local clocks according to the phase value of their global dominant pulses, and by that avoid lengthy chain reactions of clock updates.

Eq. (3) defines the adjustment value, $aim(\gamma_i)$, for the grasshopper strategy. Whenever node $p_i \in N$ notices that its clock phase value is dominated by the one of node $p_j \in N$, node p_i aligns its clock phase value with the one of p_j, see the JUMP step. Thus, whenever a single global dominant pulse exists, the convergence speed-up is made simple, because all nodes adjust their clock values according to the dominant pulse of p_j. Thus, there are no chain reactions of clock updates. Note that node p_j does not adjust its clock, see the WAIT

Set up: Given nodes $N = \{p_1, \ldots, p_9\}$, the pulse profiles, $\{\gamma_i\}_{p_i \in N}$, encode pulses, such that initially p_1 and p_6 are global dominants.
Convergence: During the first TDMA frame, nodes $p_x \in \{p_2, \ldots, p_5\}$ and $p_y \in \{p_7, p_8, p_9\}$ take the JUMP action to align their local clocks with their respective preceding global dominant pulse, p_1 and p_6 (solid lines). Whereas, node p_1 and p_6 take the MIX action to either stay or align to next dominant pulse (dotted lines).

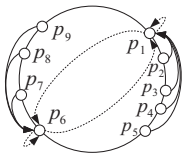


Fig. 6. Typical convergence process of the grasshopper strategy.

step. For the possible case of many global dominant pulses, we take the mixed strategy approach, see the MIX step. Here, chain reactions of clock updates can occur (Fig. 6). They occur only among the nodes whose clock phase values are global dominants.

$$aim_{\gamma_i} = \begin{cases} DomPulse_i & : DomPulse_i < P \quad \text{JUMP} \\ 0 & : Dom_i = P \\ & \wedge\ OneGlobal(\gamma_i) \quad \text{WAIT} \\ NextDomPulse_i \text{ or } 0; & : \textbf{else} \quad \text{MIX} \\ \text{each with probability } \frac{1}{2} & \end{cases} \tag{3}$$

IV. EXPERIMENTAL EVALUATION

Computer simulations and the MicaZ platform are used for showing that: (1) both proposed algorithms achieve a small synchrony bound, and (2) the grasshopper, which has a higher resource consumption cost, converges faster than the cricket.

Experiments design The proposed algorithms aim at aligning the TDMA timeslots during the MAC's timeslot assignment period. Since communication interruptions can occur, the nodes might not correctly observe their local pulse profiles. Therefore, we compare the result parameters, synchrony bound, ψ, and convergence time, ℓ_ψ, using both: (1) a MAC protocol that uses preassigned timeslots, and (2) Chameleon-MAC, a self-$\star$ TDMA protocol [8]. Moreover, the platform validation considers a control experiment using a *centralized pulse synchronizer*. This external time source is provided by a (base-station) node that periodically broadcasts. The centralized pulse synchronizer serves as a baseline for estimating the overheads imposed by the autonomous design.

The analysis considers the average over 8 experiments in which the simulations consider a timeslot size of, $P = 5\ msec$, and a communication delay bound of, $\alpha = 5\%$ of P.

Simulation experiments The proposed pulse synchronization algorithms are simulated using TOSSIM [9] on single-hop, multi-hop and mobile ad hoc networks. We observe the synchrony bound and convergence time, and study the proposed algorithms' relation to MAC-layer, network scalability and topological changes.

Single-hop Ad Hoc Network Both algorithms reduce the synchrony bound down to 1% of the timeslot size, see Fig. 7. Moreover, the synchrony bounds of the cricket and grasshopper are 24%, and respectively, 62% lower when using preassigned TDMA rather than Chameleon-MAC [8]. However, these values drop to 0.04%, and respectively, 0.4% after convergence. Furthermore, the grasshopper convergence is 5.4 times faster than of the cricket. In addition, the cricket and grasshopper converge 6.8%, and respectively, 40% times faster when using preassigned TDMA rather than Chameleon-MAC, see the cricket's lengthy chain reactions explained in Section III.

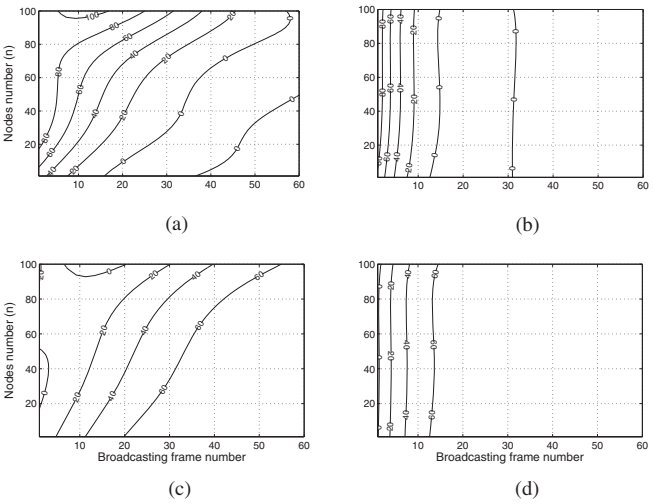


Fig. 8. Synchrony bounds (as timeslot percentage) and throughput levels (as radio time utilized percentage) for single-hop networks of $n \in \{10, 20, 30, \cdots, 100\}$ nodes. Top and bottom contour plots show the synchrony bounds, and respectively, throughput levels for cricket (a) and (c), and grasshopper (b) and (d). Given these plots, the number of TDMA frames needed to reach to a particular synchrony bound (or throughput) by a given number of nodes can be estimated. E.g., 60 nodes reach 20% synchrony within 25 frames using the cricket strategy.

We also study the algorithms' scalability by considering a variable number of nodes, $n \in \{10, 20, 30, \ldots, 100\}$. The grasshopper converges faster than the cricket as the number of nodes increases, cf. Fig. 8 (a) and (b). The convergence depends on the number of nodes. E.g., for 10% synchrony bound, $0.3n + 6.4$ and $0.0062n + 10.86$ are linear interpolations of the convergence time for the cricket, and respectively, grasshopper strategies. Moreover, $2(log_2(n) + 0.1)$ is a logarithmic interpolation of the grasshopper convergence time. This suggests that the grasshopper has lower dependency on the network size than the cricket. During the grasshopper executions, we often observed a single global dominant pulse that facilitates rapid TDMA alignment, rather than a lengthy chain reactions, see Section III. The proposed algorithms affect the MAC throughput, which is the radio time utilization percentage, cf. Fig. 8 (c) and (d). Both algorithms eventually reach a throughput of 70%.

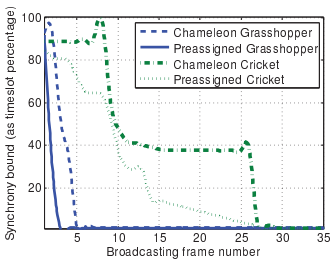


Fig. 7. 20 node single-hop network with two kind of MACs.

Multi-hop Ad Hoc Network Fig. 9-(left) considers networks with 45 nodes, and diameters of 6 hops. Often, synchrony bounds depend on the network diameter [7]. The observed synchrony bound increased to 3% of the timeslot

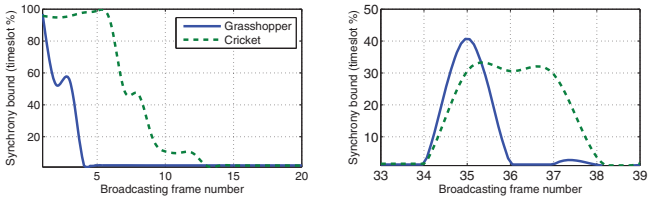


Fig. 9. (Left) Multi-hop network synchrony bound and convergence time the algorithms using Chameleon-MAC. (Right) Cricket and grasshopper convergence with mobile clusters.

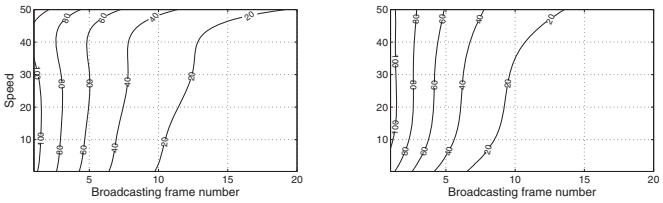


Fig. 10. The synchrony bound (as timeslot percentage) for the cricket (left) and grasshopper (right) using Chameleon-MAC and considering regular interferences. The neighborhood change rate increases with speed, causing the algorithms to spend longer time for convergence.

size and the grasshopper converged 3.25 times faster than the cricket.

Mobile Ad Hoc Network We borrow two mobility models from [8] for studying the algorithms. One in which radio interferences follow regular patterns when the nodes are placed in parallel lanes and move in opposite directions (72 node and diameter of 12). Both algorithms have quickly reached to a 10% synchrony bound, see Fig. 10, where the transmission (interference) radius was 22 distance units. The second model considers 2 clusters of 50 nodes each that pass by each other and thus they experience transient radio interferences, see Fig. 9-(right). Initially, the two clusters differ in their synchronized phase value. This difference results in timeslot misalignment and an increase in synchrony bound when the clusters are within each others interference range. We observed that the grasshopper was able to show a shorter recovery time and a greater resiliency degree.

V. Discussions

The prospects of safety-critical vehicular systems depend on the existence of predictable communication protocols that divide the radio time regularly and fairly. This paper presents autonomous and self-$\star$ algorithmic solutions for the problem of TDMA timeslot alignment by considering the more general problem of (decentralized) local pulse synchronization. The studied algorithms facilitate autonomous TDMA-based MAC protocols that are robust to transient faults, have high throughput and offer a greater predictability degree with respect to the transmission schedule. These properties are often absent from current MAC protocol implantations for VANETs, see [1, 13].

We saw that avoiding clock update dependencies can significantly speed up the convergence and recovery processes. In particular, the grasshopper algorithm foresees dependencies among the clock updates, which the cricket cannot. However, dependency avoidance requires additional resources.

Existing vehicular systems often assume the availability of common time sources, e.g., GPS. Autonomous systems cannot depend on GPS services, because they are not always available, or preferred not to be used, due to their cost. Arbitrarily long failure of signal loss can occur in underground parking lots and road tunnels. Moreover, some vehicular applications cannot afford accurate clock oscillators that would allow them to maintain the required precision during these failure periods.

By demonstrating the studied algorithms on inexpensive MicaZ motes, we have opened up the door for *hybrid-autonomous* designs (cf. centralized pulse synchronizer in Section IV). Namely, nodes that have access to GPS, use this time source for aligning their TDMA timeslots, whereas nodes that have no access to GPS, use the studied strategies as dependable fallback for catching up with nodes that have access to GPS.

We expect applicability of the hybrid-autonomous design criteria to other areas of VANETs. E.g., spatial TDMA [13] protocols base their timeslot allocation on GPS availability. As future work, we propose dealing with such dependencies by adopting the hybrid-autonomous design criteria.

References

[1] K. Bilstrup, E. Uhlemann, E. G. Ström, and U. Bilstrup. "Evaluation of the IEEE 802.11p MAC method for vehicle-to-vehicle communication," *IEEE VTC Fall*, pp. 1–5, 2008.

[2] A. Cornejo and F. Kuhn. "Deploying wireless networks with beeps," *Distributed Systems and Networks*, pp. 148–162, 2010.

[3] A. Daliot, D. Dolev, and H. Parnas. "Self-stabilizing pulse synchronization inspired by biological pacemaker networks," *Stabilization, Safety, and Security of Distributed Systems*, pp. 32–48, 2003.

[4] T. Herman and C. Zhang. "Best paper: Stabilizing clock synchronization for wireless sensor networks," *Stabilization, Safety, and Security of Distributed Systems*, pp. 335–349, 2006.

[5] E. N. Hoch. "Self-stabilizing byzantine pulse and clock synchronization," Master's thesis, CSE Hebrew Univ. of Jerusaelm, 2007.

[6] J.-H. Hoepman, A. Larsson, E. M. Schiller, and P. Tsigas. "Secure and self-stabilizing clock synchronization in sensor networks," *Stabilization, Safety, and Security of Distributed Systems*, v. 4838 of *LNCS*, pp. 340–356. Springer, 2007.

[7] C. Lenzen, T. Locher, and R. Wattenhofer. "Tight bounds for clock synchronization," *J. ACM*, 57(2), 2010.

[8] P. Leone, M. Papatriantafilou, E. M. Schiller, and G. Zhu. "Chameleon-mac: Adaptive and self-* algorithms for media access control in mobile ad hoc networks," *Stabilization, Safety, and Security of Distributed Systems*, pp. 468–488, 2010.

[9] P. Levis, N. Lee, M. Welsh, and D. E. Culler. "TOSSIM: accurate and scalable simulation of entire tinyos applications," *ACM SenSys*, pp. 126–137, 2003.

[10] D. Lucarelli and I.-J. Wang. "Decentralized synchronization protocols with nearest neighbor communication," *ACM SenSys*, pp. 62–68, 2004.

[11] R. E. Mirollo, Steven, and H. Strogatz. "Synchronization of pulse-coupled biological oscillators," *SIAM*, 50:1645–1662, 1990.

[12] M. H. Mustafa. "Self-$\star$ Pulse Synchronization for Autonomous TDMA MAC in VANETs," *CSE, Chalmers Univ. of Tech.*, 2012.

[13] K. Sjöberg, E. Uhlemann, and E. G. Ström. "Delay and interference comparison of CSMA and self-organizing TDMA when used in VANETs," *Wireless Comm. & Mobile Comp.*, pp. 1488–1493, 2011.

[14] R. Solis, V.S. Borkar, and P.R. Kumar. "A New Distributed Time Synchronization Protocol for Multihop Wireless Networks," IEEE Decision and Control, pp. 2734–2739. 2006.

[15] A. Tyrrell, G. Auer, and C. Bettstetter. "Fireflies as role models for synchronization in ad hoc networks," *ACM ICST BIONETICS*. 2006.

[16] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal. "Firefly-inspired sensor network synchronicity with realistic radio effects," *ACM SenSys*, pp. 142–153. 2005.

A.1.7 Self-Stabilizing End-to-End Communication in Bounded Capacity, Omitting, Duplicating and Non-FIFO Dynamic Networks

"Self-Stabilizing End-to-End Communication in Bounded Capacity, Omitting, Duplicating and Non-FIFO Dynamic Networks". S. Dolev, H. Ariel, E. M. Schiller and S. Sharma, 14th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'12), Toronto, Canada, October 2012.

This page is intentionally left blank.

Self-stabilizing End-to-End Communication in (Bounded Capacity, Omitting, Duplicating and non-FIFO) Dynamic Networks*

(Extended Abstract)

Shlomi Dolev[1], Ariel Hanemann[1],
Elad Michael Schiller[2], and Shantanu Sharma[1]

[1] Department of Computer Science, Ben-Gurion University of the Negev, Israel
{dolev,hanemann,sharmas}@cs.bgu.ac.il**
[2] Department of Computer Science and Engineering, Chalmers University of Technology, Sweden
elad@chalmers.se***

Abstract. End-to-end communication over the network layer (or data link in overlay networks) is one of the most important communication tasks in every communication network, including legacy communication networks as well as mobile ad hoc networks, peer-to-peer networks and mash networks. We study end-to-end algorithms that exchange packets to deliver (high level) messages in FIFO order without omissions or duplications. We present a self-stabilizing end-to-end algorithm that can be applied to networks of bounded capacity that omit, duplicate and reorder packets. The algorithm is network topology independent, and hence suitable for always changing dynamic networks with any churn rate.

1 Introduction

End-to-end communication is a basic primitive in communication networks. A *sender* must transmit messages to a *receiver* in an exactly once fashion, where no omissions, duplications and reordering are allowed. Errors occur in transmitting packets among the network entities – one significant source of error is noise in the transmission media. Thus, error detection and error correcting techniques are employed as an integral part of the transmission in the communication network. These error detection and correction codes function with high probability. Still,

* Also appears as a technical report in [10].

** Partially supported by Deutsche Telekom, Rita Altura Trust Chair in Computer Sciences, Lynne and William Frankel Center for Computer Sciences, Israel Science Foundation (grant number 428/11), Cabarnit Cyber Security MAGNET Consortium, Grant from the Institute for Future Defense Technologies Research named for the Medvedi of the Technion, Israeli Internet Association, and Israeli Defense Secretary (MAFAT).

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

A.W. Richa and C. Scheideler (Eds.): SSS 2012, LNCS 7596, pp. 133–147, 2012.

when there is a large volume of communication sessions, the probability that an error will not be detected becomes high, leading to a possible malfunction of the communication algorithm. In fact, it can lead the algorithm to an arbitrary state from which the algorithm may never recover unless it is *self-stabilizing* [8]. By using packets with enough distinct labels infinitely often, we present a self-stabilizing end-to-end communication algorithm that can be applied to dynamic networks of bounded capacity that omit, duplicate and reorder packets.

Contemporary communication and network technologies enhance the need for automatic recovery and interoperability of heterogeneous devices and the means of wired and wireless communications, as well as the churn associated with the totally dynamic communication networks. Having a self-stabilizing, predictable and robust basic end-to-end communication primitive for these dynamic networks facilitates the construction of high-level applications. Such applications are becoming extremely important nowadays where countries' main infrastructures, such as the electrical smart-grid, water supply networks and intelligent transportation, are based on cyber-systems. Defining the communication network as a bounded capacity network that allows omissions, duplications and reordering of packets and building (efficient) exactly once message transmission using packets, allows us to abstract away the exact network topology, dynamicity and churn.

The dynamic and difficult-to-predict nature of electrical smart-grid and intelligent transportation systems give rise to many fault-tolerance issues and require efficient solutions. Such networks are subject to transient faults due to hardware/software temporal malfunctions or short-lived violations of the assumed settings for the location and state of their nodes. Fault-tolerant systems that are *self-stabilizing* [8,7] can recover after the occurrence of transient faults, which can drive the system to an arbitrary system state. The system designers consider *all* configurations as possible configurations from which the system is started. The self-stabilization design criteria liberate the system designer from dealing with specific fault scenarios, the risk of neglecting some scenarios, and having to address each fault scenario separately.

Related Work and Our Contribution. End-to-end communication and data-link algorithms are fundamental for any network protocol [25]. End-to-end algorithms provide the means for message exchange between senders and receivers over unreliable communication links. Not all end-to-end communication and data-link algorithms assume initial synchronization between senders and receivers. For example, Afek and Brown [1] presented a self-stabilizing alternating bit protocol (ABP) for FIFO packet channels without the need for initial synchronization. Self-stabilizing token passing was used as the bases for self-stabilizing ABP over unbounded capacity and FIFO preserving channels in [17,11]. Spinelli [24] introduced two self-stabilizing sliding window ARQ protocols for unbounded FIFO channels. Dolev and Welch [15] considered tolerating network errors in dynamic networks with FIFO non-duplicating communication links, and use source routing over paths to cope with crashes. In

contrast, we do not consider known network topology nor base our algorithms on a specific routing policy. We merely assume bounded network capacity.

In [2], an algorithm for self-stabilizing unit capacity data link over a FIFO physical link is assumed. Flauzac and Villai [16] described a snapshot algorithm that uses bidirectional and FIFO communication channels. Cournier et al. [5] considered a snap-stabilizing algorithm [3] for message forwarding over message switched network. They ensure one time delivery of the emitted message to the destination within a finite time using destination based buffer graph and assuming underline FIFO packet delivery.

In the context of dynamic networks and mobile ad hoc networks, Dolev, Schiller and Welch [14,12,13] presented self-stabilizing algorithms for token circulation, group multicast, group membership, resource allocation and estimation of network size. Following [14,12,13], similar approaches to cope with constantly changing networks have been investigated [22] in addition to other fundamental problems such as clock synchronization [21], dissemination [18,20], leader election [19,6,4], and consensus [23] to name a few. In this paper, we investigate the basic networking tasks of end-to-end communication over the network layer (or overlay networks), that are required for the design of fundamental problems, such as the aforementioned problems considered in [21,22,18,20,19,6,4,23].

Recently, Dolev et al. [9] presented a self-stabilizing data link algorithm for reliable FIFO message delivery over bounded non-FIFO and non-duplicating channel. This paper presents the first, to the best of our knowledge, self-stabilizing end-to-end algorithms for reliable FIFO message delivery over bounded non-FIFO and *duplicating* channel.

Due to space limit, some of the proofs are omitted from this extended abstract and can be found in [10].

2 System Settings

We consider a distributed system that includes *nodes* (or processors), $p_1, p_2, \ldots, p_N$. We represent a distributed system by a *communication graph* that may change over time, where each processor is represented as a node. Two *neighboring* processors, p_i and p_j, that can exchange packets directly are connected by a link in the communication graph. Packet exchange between neighbors is carried via (directed) communication links, where packets are sent from p_i to p_j through the directed link (p_i, p_j) and packets are sent from p_j to p_i through (p_j, p_i), the opposite directed link. End-to-end communication among non-neighbor nodes, p_s and p_r, is facilitated by packet relaying from one processor to neighbors. Thus, establishing a (virtual) communication link between p_s and p_r in which p_s is the sender and p_r is the receiver. We assume the communication graph is dynamic, and is constantly changed, while respecting N as the upper bound on the number of nodes in the system. Packets are exchanged by the sender and the receiver in order to deliver (high level) messages in a reliable fashion. We assume that the entire number of packets in the system at any given time, does not exceed a known bound. We allow any churn rate,

assuming that joining processors reset their own memory, and by that assist in respecting the assumed bounded packet capacity of the entire network.

The communication links are bidirectional. Namely, between every two nodes, p_i and p_j, that can exchange packets, there is a unidirectional *channel (set)* that transfers packets from p_i to p_j and another unidirectional channel that transfer packets from p_j to p_i. We model the *(communication) channel*, from node p_i to node p_j as a (non-FIFO order preserving) packet set that p_i has sent to p_j and p_j is about to receive. When p_i sends a packet m to p_j, the operation *send* inserts a copy of m to the channel from p_i to p_j as long as the upper bound of packets in the channel is respected. Once m arrives, p_j triggers the *receive* event and m is deleted from the set. The communication channel is non-FIFO and has no reliability guarantees. Thus, at any time the sent packets may be omitted, reordered, and duplicated, as long as the link capacity bound is not violated. We note that transient faults can bring the system to consist of arbitrary, and yet capacity bounded, channel sets from which convergence should start. We assume that when node p_i sends a packet, *pckt*, infinitely often through the communication link from p_i to p_j, p_j receives *pckt* infinitely often. We intentionally do not specify (the possible unreliable) routing scheme that is used to forward a packet from the sender to the receiver, e.g., flooding, shortest path routing. We assume that the overall network capacity allows a channel from p_i to p_j to contain at most *capacity* packets at any time, where *capacity* is a known constant. However, it should be noted that although the channel has a maximal capacity, packets in the channel may be duplicated infinitely many times because even if the channel is full, packets in the channel may be either lost or received. This leaves places for other packets to be (infinitely often) duplicated and received by p_j.

Self-stabilizing algorithms do not terminate (see [8]). The non-termination property can be easily identified in the code of a self-stabilizing algorithm: the code is usually a do forever loop that contains communication operations with the neighbors. An *iteration* is said to be complete if it starts in the loop's first line and ends at the last (regardless of whether it enters branches).

Every node, p_i, executes a program that is a sequence of *(atomic) steps*. Where a step starts with local computations and ends with a single communication operation, which is either *send* or *receive* of a packet. For ease of description, we assume the interleaving model, where steps are executed atomically, a single step at any given time. An input event can either be the receipt of a packet or a periodic timer going off triggering p_i to send. Note that the system is totally asynchronous and the non-fixed spontaneous send of nodes and node processing rates are irrelevant to the correctness proof.

The *state*, s_i, of a node p_i consists of the value of all the variables of the node including the set of all incoming communication channels. The execution of an algorithm step can change the node state. The term *(system) configuration* is used for a tuple of the form $(s_1, s_2, \cdots, s_N)$, where each s_i is the state of node p_i (including packets in transit for p_i). We define an *execution (or run)*

$R = c[0], a[0], c[1], a[1], \ldots$ as an alternating sequence of system configurations $c[x]$ and steps $a[x]$, such that each configuration $c[x+1]$ (except the initial configuration $c[0]$) is obtained from the preceding configuration $c[x]$ by the execution of the step $a[x]$. We often associate the notation of a step with its executing node p_i using a subscript, e.g., a_i. An execution R is *fair* if every node, p_i, executes infinitely many steps in R. We represent the omissions, duplications and reordering using environment steps that are interleaved with the steps of the processors in the run R. In every fair run, the environment steps do not prevent communication, namely, infinite *send* operations of p_i of a packet, *pckt*, to p_j implies infinite *receive* operations of *pckt* by p_j.

The system is asynchronous and the notion of time, for example, when considering system convergence to legal behavior, is measured by the number of *asynchronous rounds*, where the first asynchronous round is the minimal prefix of the execution in which every node sends at least one packet to every neighbor and one of these packets is received by each neighbor. Thus, we nullify the infinite power of omissions, duplications and reordering when measuring the algorithm performance. Moreover, we ensure that packets sent are eventually received; otherwise the channel is, in fact, disconnected. The second asynchronous round is the first asynchronous round in the suffix of the run that follows the first asynchronous round, and so on. We measure the communication costs by the number of packets sent in synchronous execution in which each packet sent by p_s arrives to its destination, p_r, in one time unit, and before p_s sends any additional packet to p_r.

We define the system's task by a set of executions called *legal executions* (LE) in which the task's requirements hold. A configuration c is a *safe configuration* for an algorithm and the task of LE provided that any execution that starts in c is a legal execution (belongs to LE). An algorithm is *self-stabilizing* with relation to the task LE when every (unbounded) execution of the algorithm reaches a safe configuration with relation to the algorithm and the task.

The *self-stabilizing end-to-end communication* (S^2E^2C) algorithm provides FIFO guarantee for bounded networks that omit duplicate and reorder packets. Moreover, the algorithm considers arbitrary starting configurations and ensures error-free message delivery. In detail, given a system's execution R, and a pair, p_s and p_r, of sending and receiving nodes, we associate the message sequences $s_R = m_0, m_1, m_2, \ldots$, of messages fetched by p_s, with the message sequence $r_R = m'_0, m'_1, m'_2, \ldots$ of messages delivered by p_r. Note that we list messages according to the order they are fetched (from the higher level application) by the sender, thus two or more (consecutive or non-consecutive) messages can be identical. The S^2E^2C task requires that for every legal execution $R \in LE$, there is an infinite suffix, R', in which infinitely many messages are delivered, and $s_{R'} = r_{R'}$. It should be noted that packets are not actually received by the receiver in their correct order but eventually it holds that messages are delivered by the receiver (to higher level application) in the right order.

3 The End-to-End Algorithm

Dynamic networks have to overcome a wide range of faults, such as message corruption and omission. It often happens that networking techniques, such as retransmissions and multi-path routing, which are used for increasing robustness, can cause undesirable behavior, such as message duplications and reordering. We present a self-stabilizing end-to-end communication algorithm that uses the network's bounded capacity, to cope with packet corruptions, omissions, duplications, and reordering. We abstract the entire network by two directed channels, one from the sender to the receiver and one from the receiver to the sender, where each abstract channel is of a bounded capacity. These two abstract channels can omit, reorder and duplicate packets. We regard two nodes, p_s, p_r, as sender and receiver, respectively. Sender p_s sends packets with distinct labels infinitely often until p_s receives a sufficient amount of corresponding distinct acknowledgment labels from the receiver p_r.

For the sake of readability, we start describing the algorithm using large overhead, before showing ways to dramatically reduce the overhead. The sender repeatedly sends each message m with a three state *alternating index*, which is either 0, 1 or 2. We choose to discuss, without the loss of generality, the case of a message with alternating index 0, where $\langle 0, m\rangle$ is repeatedly sent in $(2 \cdot \mathit{capacity} + 1)$ packet types. Each type uses a distinct label in the range 1 to twice the capacity plus 1. Namely, the types are: $\langle 0, 1, m\rangle$, $\langle 0, 2, m\rangle$, ..., $\langle 0, 2 \cdot \mathit{capacity} + 1, m\rangle$. The sender waits for an acknowledgment of the packet arrival for each of the $(2 \cdot \mathit{capacity} + 1)$ distinct labels, and an indication that the receiver delivered a message due to the arrival of $(\mathit{capacity}+1)$ packets with alternating index 0. The receiver accumulates the arriving packets in an array of $(2 \cdot \mathit{capacity}+1)$ entries, where each entry, j, stores the last arriving packet with distinct label j. Whenever the receiver finds that $(\mathit{capacity} + 1)$ recorded array entries share the same alternating index, for example 1, the receiver delivers the message m encapsulated in one in-coming packet recorded in the array – this packet has the alternating index of the majority of recorded packets; 1 in our example. Then, the receiver resets its array and starts accumulating packets again, until $(\mathit{capacity} + 1)$ recorded copies, with the same alternating index reappear. The receiver always remembers the last delivered alternating index, ldai, that caused the reset of its array, and does not deliver two successive messages with the same alternating index. Each packet $\langle \mathit{ai}, \mathit{lbl}, m\rangle$ that arrives to the receiver is acknowledged by $\langle \mathit{lbl}, \mathit{ldai}\rangle$. The sender accumulates the arriving packet in an array of $(2 \cdot \mathit{capacity} + 1)$ entries and waits to receive a packet for each entry, and to have a value of ldai that is equal to the alternating index the sender is currently using in the sent packets in at least $(\mathit{capacity} + 1)$ of the recorded packets. Once such a packet set arrives, the sender resets its array, fetches a new message, m', to be delivered, and increments the alternating index by 1 modulo 3 for the transmission process of the next message, m'.

The correctness considers the fact that the receiver always acknowledges incoming packets, and hence the sender will infinitely often fetch messages. Following the first fetch of the sender, the receiver follows the sender's alternating

index, records it in $ldai$, and acknowledges this fact. We consider an execution in which the sender changes the alternating index in to x, $x+1$, $x+2$, x (all modulo 3). In this execution, the sender is acknowledged that the receiver changes $ldai$ to $x+1$ and then to $x+2$, while the sender does not send packets with alternating index x, thus, the last x delivery in the sequence must be due to fresh packets, packets sent after the packets with alternating index $x+2$ were sent, and cause a delivery.

In the preceding text a simplified algorithm with a large overhead was presented – a more efficient algorithm is described in the following. The basic idea is to enlarge the arrays to have more than $n > (2 \cdot capacity + 1)$ recorded packets. Roughly speaking, in such a case the minority of the distinct label packets accumulated in the arrays are erroneous, i.e., packet copies that were accumulated in the network prior to the current fetch (maximum *capacity*). The other $(n - capacity)$ distinct label accumulated packets are correct. Thus, as we know the maximal amount of unrelated packets, we can manipulate the data so that the $n - capacity$ correct packets, each of length pl will encode, by means of error correcting codes, pl messages each of length ml, a length slightly shorter than n. The sender fetches a window of pl messages each of length ml, where pl is the maximal packet length beyond the header. The sender then uses error correcting codes so that a message of length ml is coded by a word of length n, such that the encoded word can tolerate up to *capacity* erroneous bits. The pl encoded messages of length n are then converted to n packets of length pl in a way that the i^{th} message out of the ml fetched messages is encoded by the i^{th} bits of all the n distinct packets that are about to be transmitted. So eventually, the first bit of all distinct labeled packets, ordered by their distinct labels, encode, with redundancy, the first message, and the second bit of all distinct labeled packets, encode, with redundancy, the second message, etc. Fig. 1 shows the formation of the n packets from the pl messages. When the receiver accumulates n distinct label packets, the *capacity* of the packets may be erroneous. However, since the i^{th} packet, out of the n distinct packets, encodes the i^{th} bits of all the pl encoded messages, if the i^{th} packet is erroneous, then the receiver can still decode the data of the original pl messages each of length $ml < n$. The i^{th} bit in each encoded message may be wrong, in fact, capacity of packets maybe erroneous yielding capacity of bits that may be wrong in each encoded message, however, due to the error correction, all the original pl messages of length ml can be recovered, so the receiver can deliver the correct pl messages in the correct order.

In this case, the sender repeatedly sends n distinct packets and the receiver keeps sending $(capacity + 1)$ packets each with a distinct label in the range 1 to $(capacity + 1)$. In addition, each of these packets contains the receiver's current value of $ldai$. The packets from the receiver are sent infinitely often, not necessarily as a response to its received packets. When the receiver accumulates n distinct label packets with the same alternating index, it recovers the original pl messages, delivers them, resets its received packets array and changes its $ldai$ to the alternating index of the packets that it just delivered. We note that these

received packets must be different from its current *ldai* because the receiver does not accumulate packets if their alternating index is equal to its current *ldai*. The sender may continue sending the n packets with alternating index *ldai*, until the sender accumulates $(capacity + 1)$ distinct label acknowledging packets with alternating index *ldai*. However, because now the packets' alternating index is equal to its current *ldai*, the receiver does not accumulate them, and hence does not deliver a duplicate. Once the sender accumulates $(capacity + 1)$ packets with *ldai* equal to its alternating index, it will fetch *pl* new messages, encode and convert them to n distinct label packets and increase its alternating index by 1 modulo 3.

The correctness arguments use the same facts mentioned above in the majority based algorithm. Eventually, we will reach an execution in which the sender fetches a new set of messages infinitely often and the receiver will deliver the messages fetched by the sender before the sender fetches the next set of messages.

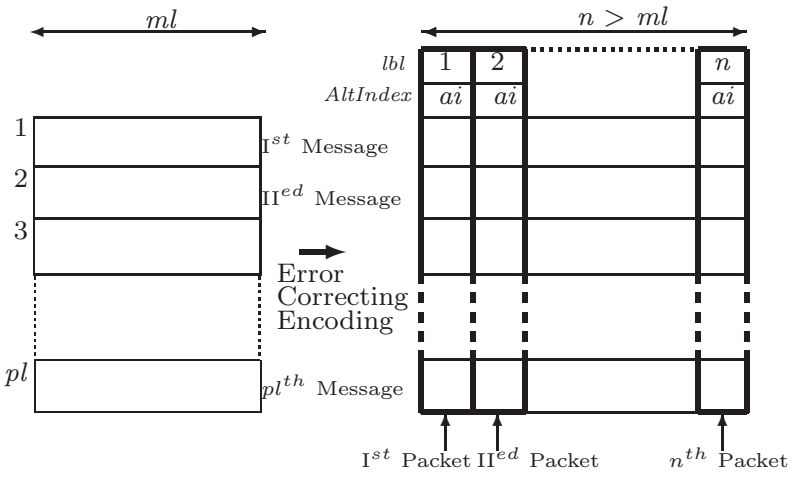


Fig. 1. Packet formation from messages

Eventually, every set of *pl* fetched messages is delivered exactly once because after delivery the receiver resets its packets record array and changes *ldai* to be equal to the senders alternating index. The receiver stops accumulating packets from the sender until the sender fetches new messages and starts sending packets with a new alternating index. Between two delivery events of the receiver, the receiver will accumulate n distinct label packets of an identical alternating index, where $(n - capacity)$ of them must be fetched by the sender after the last delivery of messages by the receiver. The fact, which reflects such behavior at the receiver node, is that the sender only fetches new messages after it gets $(capacity + 1)$ distinct packets with *ldai* equal to its current alternating index. When the receiver holds n distinct label packets with maximum capacity erroneous packets, it can convert the packets back to the original messages by applying the error correction code capabilities and deliver the original message correctly.

Algorithm Description. Algorithms 1 and 2 implement the proposed S^2E^2C sender-side and receiver-side algorithms, respectively. The two nodes, p_s and p_r, are the sender and the receiver nodes respectively. The Sender algorithm consists of a do forever loop statement (line 2 of the Sender algorithm), where the sender, p_s, assures that all the data structures comprises only valid contents. I.e., p_s checks that the ACK_set_s holds packets with alternating index equal to the senders current $AltIndex_s$ and the labels are between 1 and $(capacity + 1)$.

Algorithm 1. Self-Stabilizing End-to-End Algorithm (Sender)

Persistent variables:
$AltIndex$: an integer $\in [0, 2]$ that states the current alternating index value
ACK_set: at most $(capacity + 1)$ acknowledgment set, where items contain labels and last delivered alternating indexes, $\langle lbl, ldai \rangle$
$packet\_set$: n packets, $\langle AltIndex, lbl, dat \rangle$, to be sent, where $lbl \in [1, n]$ and dat is data of size pl bits

Interface:
$Fetch(NumOfMessages)$ Fetches $NumOfMessages$ messages from the application and returns them in an array of size $NumOfMessages$ according to their original order
$Encode(Messages[])$ receives an array of messages of length ml each, M, and returns a message array of identical size M', where message $M'[i]$ is the encoded original $M[i]$, the final length of the returned $M'[i]$ is n and the code can tolerate $capacity$ errors

```
1  Do forever begin
2      if (ACK_set ⊈ {AltIndex} × [1, capacity + 1]) then
       (ACK_set, messages) ← (∅, Encode(Fetch(pl)))
3      foreach pckt ∈ packet_set() do send pckt
4  Upon receiving ACK = ⟨lbl, ldai⟩ begin
5      if lbl ∈ [1, capacity + 1] ∧ ldai = AltIndex then
6          ACK_set ← ACK_set ∪ {ACK}
7          if capacity <| ACK_set | then begin
8              AltIndex ← (AltIndex + 1) mod 3
9              (ACK_set, messages) ← (∅, Encode(Fetch(pl)))
10 Function packet_set() begin
11     foreach (i, j) ∈ [1, n] × [1, pl] do let data[i].bit[j] = messages[j].bit[i]
12     return {⟨AltIndex, i, data[i]⟩}_{i∈[1,n]}
```

In case any of these conditions is unfulfilled, the sender resets its data structures (line 2 of the Sender algorithm). Subsequently, p_s triggers the *Fetch* and the *Encode* interfaces (line 2 of the Sender algorithm). Before sending the packets, p_s executes the $packet\_set()$ function (line 3 of the Sender algorithm).

The Sender algorithm, also, handles the reception of acknowledgments $ACK_s = \langle lbl, ldai \rangle$ (line 4 of the Sender algorithm). Each ACK_s has distinct labels, corresponding to already transmitted packets. On the reception of the $(capacity + 1)$ distinct label ACK_s, p_s keeps ACK_s in ACK_set_s (line 6 of the Sender algorithm), if ACK_s have the value of $ldai$ (last delivered alternating index) equals to $AltInex$ (line 5 of the Sender algorithm). When p_s gets an ACK_s packet $(capacity + 1)$ times (line 7 of the Sender algorithm), p_s changes $AltIndex_s$ (line 8 of the Sender algorithm). Afterwards, p_s does reset ACK_set_s and calls $Fetch()$ and $Encode()$ interfaces (line 9 of the Sender algorithm).

Algorithm 2. Self-Stabilizing End-to-End Algorithm (Receiver)

Persistent variables:
$packet\_set$: packets, $\langle AltIndex, lbl, dat\rangle$, received, where $label \in [1, n]$ and dat is data of size pl bits
$LastDeliveredIndex$: an integer $\in [0, 2]$ that states the alternating index value of the last delivered packets

Interface:
$Decode(Messages[])$ receives an array of encoded messages, M', of length n each, and returns an array of decoded messages of length ml, M, where $M[i]$ is the decoded $M'[i]$. The code is the same error correction coded by the sender and can correct up to *capacity* mistakes
$Deliver(messages[])$ receives an array of messages and delivers them to the application by the order in the array

Macros:
$P(ind) = \{\langle ind, *, *\rangle \in packet\_set\}$

```
1  Do forever begin
2      if {⟨ai, lbl⟩ : ⟨ai, lbl, ∗⟩ ∈ packet_set} ⊈
       {[0, 2] \ {LastDeliveredIndex}} × [1, n] × {∗}∨
       (∃⟨ai, lbl, dat⟩ ∈ packet_set : ⟨ai, lbl, ∗⟩ ∈ packet_set \ {⟨ai, lbl, dat⟩})∨
       (∃pckt = ⟨∗, ∗, data⟩ ∈ packet_set :| pckt.data |≠ pl)∨
       1 <| {AltIndex : n ≤| {⟨AltIndex, ∗, ∗⟩ ∈ packet_set} |} | then
       packet_set ← ∅
3      foreach i ∈ [1, capacity + 1] do send ⟨lbl, LastDeliveredIndex⟩
4  Upon receiving pckt = ⟨ai, lbl, dat⟩ begin
5      if ⟨ai, lbl, ∗⟩ ∉ packet_set∧
       ⟨ai, lbl⟩ ∈ ({[0, 2] \ {LastDeliveredIndex}} × [1, n])∧ | dat |= pl then
6          packet_set ← packet_set ∪ {pckt}
7          if ∃ ! ind : ind ≠ LastDeliveredIndex ∧ n ≤| P(ind) |: P(ind) =
           {⟨ind, ∗, ∗⟩ ∈ packet_set} then
8              foreach (i, j) ∈ [1, pl] × [1, n] do
9                  let messages[i].bit[j] = data.bit[i] : ⟨ind, j, data⟩ ∈ P(ind)
10             (packet_set, LastDeliveredIndex) ← (∅, ind)
11             Deliver(Decode(messages))
```

The Receiver algorithm executes at the receiver side, p_r. The receiver p_r assures its data structure, namely, $packet\_set_r$, in do forever loop (line 2 of the Receiver algorithm). The receiver p_r audits: (i) the $packet\_set_r$ holds packets with alternating index, $ai \in [0, 2]$, except $LastDeliveredIndex_r$, labels (lbl) between 1 and n and data of size pl; (ii) the $packet\_set_r$ holds exactly one group of ai that has at least n elements. When any of the aforementioned conditions are falsified, p_r assigns the empty set to $packet\_set_r$. In addition, p_r acknowledges p_s by $(capacity + 1)$ packets (line 3 of the Receiver algorithm).

Node p_r receives a packet $pckt_r = \langle ai, lbl, dat \rangle$, see line 4 of the Receiver algorithm. If $pckt_r$ has data (dat) in the size of pl bits and $pckt_r$ has alternating index (ai) in the range from 0 to 2, excluding the $LastDeliveredIndex$ and $pckt_r$ has a label (lbl) in the range of 1 to n (line 5 of the Receiver algorithm), p_r puts $pckt_r$ in $packet\_set_r$ (line 6 of the Receiver algorithm). When p_r gets n distinct label packets of identical ai (line 7 of the Receiver algorithm), p_r forms the message from the packets (line 9 of the Receiver algorithm). Subsequent steps include the reset of the $packet\_set_r$ data structure and change of $LastDeliveredIndex_r$ to ai (line 10 of the Receiver algorithm). Next, p_r decodes and delivers the message (line 11 of the Receiver algorithm).

Correction proof. The correct packet exchange between the sender and the receiver requires coordination. The sender should wait after fetching a new message batch, i.e., executing lines 8 to 9 of the Sender algorithm, until the receiver delivers a message batch, i.e., executing line 11 of the Receiver algorithm. We describe the set of legal executions for correct packet exchange before demonstrating that the Sender and the Receiver algorithms satisfy these requirements in Theorem 1, which says that the studied algorithms implement self-stabilizing end-to-end communication (S^2E^2C) task.

Let a_{s_α} be the α^{th} time that the sender is fetching a new message batch, i.e., executing lines 8 to 9 of the Sender algorithm. Let a_{r_β} be the β^{th} time that the receiver is delivering a message batch, i.e., executing line 11 of the Receiver algorithm. With respect to the self-stabilizing end-to-end communication (S^2E^2C) task and the algorithms of the Sender and the Receiver, the legal execution set includes executions, R, that interleave the a_{s_α} and the a_{r_β} steps in a manner that matches the alternating index labels. Namely, after the occurrence of $a_{s_\alpha} \in R$ in which the sender fetches a new message batch, the step $a_{s_{\alpha+1}}$ should not occur before $a_{r_\beta} \in R$ in which the receiver delivers *that* message batch (Lemma 3). Similarly, after the occurrence of $a_{r_\beta} \in R$ in which the receiver delivers a message batch, the step $a_{r_{\beta+1}}$ should not occur before $a_{s_\alpha} \in R$ in which the sender fetches the next message batch (Lemma 4).

In addition, the a_{s_α} and the a_{r_β} steps should have matching alternating indices. The proof shows that the sender, p_s, increments its $AltIndex_s = s\_index_\alpha$ value on every a_{s_α} in a modulo 3 fashion, and the receiver, p_r, adopts s_index_α and deliver its message batch in step a_{r_β} after receiving at least $(n - capacity)$ packets that are tagged by s_index_α. Similarly, p_r acknowledges the received packets using the tag $LastDeliveredIndex_r = r\_index_\beta$, and then p_s proceeds to fetch a next message batch in $a_{s_{\alpha+1}}$ after receiving at least more than $capacity$ acknowledgments.

We note that the proof implies that within a constant number of asynchronous rounds, the receiver, p_r, receives an entire batch of n packets from its incoming abstract channel out of which $(n - capacity)$ packets are from the sender, p_s. This is true because: (1) we assume that when the sender sends a packet infinitely often through the abstract channel, the receiver receives the packet infinitely often, and (2) the proof shows that the sender does not stop sending its current batch of messages, before guaranteeing that the current message batch

had arrived to the receiver, p_r, and p_r had delivered it. Moreover, analogous arguments to arguments (1) and (2) above imply the number of asynchronous rounds, in which the sender, p_s, receives an entire batch of $(capacity + 1)$ acknowledgments that at least one of them is from the receiver.

Lemmas 1 and 2 are needed for the proof of lemmas 3 and 4. Throughout we refer to R as an execution of the Sender and the Receiver algorithms, where p_s executes the Sender algorithm and p_r executes the Receiver algorithm.

Lemma 1. *Let $c_{s_\alpha}(x)$ be the x^{th} configuration between a_{s_α} and $a_{s_{\alpha+1}}$ and $ACK_\alpha = \{ack_\alpha(\ell)\}_{\ell\in[1,capacity+1]}$ be a set of acknowledgment packets, where $ack_\alpha(\ell) = \langle \ell, s\_index_\alpha\rangle$. For any given $\alpha > 0$, there is a single index value, $s\_index_\alpha \in [0, 2]$, such that for any $x > 0$, it holds that $AltIndex_s = s\_index_\alpha$ in $c_{s_\alpha}(x)$. Moreover, between a_{s_α} and $a_{s_{\alpha+1}}$ there is at least one configuration c_{r_β}, in which $LastDeliveredIndex_r = s\_index_\alpha$. Furthermore, between a_{s_α} and $a_{s_{\alpha+1}}$, the sender, p_s, receives from the channel from p_r to p_s, the entire set, ACK_α, of acknowledgment packets (each packet at least once), and between (the first) c_{r_β} and $a_{s_{\alpha+1}}$ the receiver must send at least one $ack_\alpha(\ell) \in ACK_\alpha$ packet, which p_s receives.*

Proof. We start by showing that s_index_α exists before showing that c_{r_β} exists and that p_s receives ack_α from p_r between a_{s_α} and $a_{s_{\alpha+1}}$.

The value of $AltIndex_s = s\_index_\alpha$ is only changed in line 8 of the Sender algorithm. By the definition of a_{s_α}, line 8 is not executed by any step between a_{s_α} and $a_{s_{\alpha+1}}$. Therefore, for any given α, there is a single index value, $s\_index_\alpha \in [0, 2]$, such that for any $x > 0$, it holds that $AltIndex_s = s\_index_\alpha$ in $c_{s_\alpha}(x)$.

We show that c_{r_β} exists by showing that, between a_{s_α} and $a_{s_{\alpha+1}}$, there is at least one acknowledge packet, $\langle lbl, ldai\rangle$, that p_r sends and p_s receives, where $ldai = s\_index_\alpha$. This proves the claim because p_r's acknowledgments are always sent with $ldai = LastDeliveredIndex_r$, see line 3.

We show that, between a_{s_α} and $a_{s_{\alpha+1}}$, the receiver p_r sends at least one of the $ack_\alpha(\ell) \in ACK_\alpha$ packets that p_s receives. We do that by showing that p_s receives, from the channel from p_r to p_s , more than *capacity* packets, i.e., the set ACK_α. Since *capacity* bounds the number of packets that, at any time, can be in the channel from p_r to p_s , at least one of the ACK_α packets, say $ack_\alpha(\ell')$, must be sent by p_r and received by p_s between a_{s_α} and $a_{s_{\alpha+1}}$. This in fact proves that p_r sends $ack_\alpha(\ell')$ after c_{r_β}.

In order to demonstrate that p_s receives the set ACK_α, we note that $ACK\_set = \emptyset$ in configuration $c_{s_\alpha}(1)$, which immediately follows a_{s_α}, see line 9 of the Sender algorithm. The sender tests the arriving acknowledgment packet, ack_α, in line 5 of the Sender algorithm. It tests ack_α's label to be in the range of $[1, capacity + 1]$, and that they are of ack_α's form. Moreover, it counts that $(capacity + 1)$ different packets are added to ACK_set by adding them to ACK_set, and not executing lines 8 to 9 of the Sender algorithm before at least $(capacity + 1)$ distinct packets are in ACK_set.

Lemma 2 (proof appears in [10]). *Let $c_{r_\beta}(y)$ be the y^{th} configuration between a_{r_β} and $a_{r_{\beta+1}}$, and $PACKET_\beta(r\_index'_\beta) = \{packet_\beta(\ell, r\_index'_\beta)\}_{\ell\in[1,n]}$ be*

a packet set, where $packet_{\beta,r\_index'_\beta}(\ell) = \langle r\_index'_\beta, \ell, * \rangle$. *For any given* $\beta > 0$, *there is a single index value,* $r\_index_\beta \in [0,2]$, *such that for any* $y > 0$, *it holds that* $LastDeliveredIndex_r = r\_index_\beta$ *in configuration* $c_{r_\beta}(y)$. *Moreover, between* a_{r_β} *and* $a_{r_{\beta+1}}$ *there is at least one configuration,* c_{s_α}, *such that* $AltIndex_s \neq r\_index_\beta$. *Furthermore, there exists a single* $r\_index'_\beta \in [0,2] \setminus \{r\_index_\beta\}$, *such that the receiver,* p_r, *receives all the packets in* $PACKET_\beta(r\_index'_\beta)$ *at least once between* c_{s_α} *and* $a_{r_{\beta+1}}$, *where at least* $n - capacity > 0$ *of them are sent by the sender* p_s *between* a_{r_β} *and* $a_{r_{\beta+1}}$.

Lemmas 3 and 4 borrow their notations from lemmas 1 and 2. Lemma 4 shows that between a_{s_α} and $a_{s_{\alpha+1}}$, there is exactly one a_{r_β} step.

Lemma 3. *Between* a_{s_α} *and* $a_{s_{\alpha+1}}$, *the receiver takes exactly one* a_{r_β} *step, and that between* a_{r_β}, *and* $a_{r_{\beta+1}}$, *the sender takes exactly one* $a_{s_{\alpha+1}}$ *step.*

Proof. We start by showing that between a_{s_α} and $a_{s_{\alpha+1}}$, there is at least one a_{r_β} step before showing that there is exactly one such a_{r_β} step when $\alpha > 2$. Then, we consider a proof for showing that between a_{r_β} and $a_{r_{\beta+1}}$, there is at least one a_{s_α} step before showing that between a_{r_β} and $a_{r_{\beta+1}}$, there is exactly one a_{s_α} step when $\beta > 2$.

By Lemma 1 and line 8 of the Sender algorithm, in any configuration, $c_{s_1}(x)$, that is between a_{s_1} and a_{s_2}, the sender is using a single alternating index, s_index_1, and in any configuration, $c_{s_2}(x)$, that is between a_{s_2} and a_{s_3}, the sender is using a single alternating index, s_index_2, such that $s\_index_2 = s\_index_1 + 1 \bmod 3$. In a similar manner, we consider configuration, $c_{s_\alpha}(x)$, that is between a_{s_α} and $a_{s_{\alpha+1}}$.

Lemma 1 also shows that for $\alpha \in (1, 2, \ldots)$, there are configurations, c_{r_α}, in which $LastDeliveredIndex_r = s\_index_\alpha$. This implies that between a_{s_α} and $a_{s_{\alpha+1}}$, the receiver changes the value of $LastDeliveredIndex_r$ at least once, where $\alpha \in (1, 2, \ldots)$. Thus, by a_{r_β}'s definition and line 10 of the Receiver algorithm, there is at least one a_{r_β} step between a_{s_α} and $a_{s_{\alpha+1}}$.

To see that when $\alpha > 2$ there is exactly one such a_{r_β} step between a_{s_α} and $a_{s_{\alpha+1}}$, we consider the case in which between a_{s_α} and $a_{s_{\alpha+1}}$, there are several a_{r_β} steps, i.e., $a_{r_{\beta_{first}}}, \ldots, a_{r_{\beta_{last}}}$. In particular we consider the $a_{s_{\alpha-1}}$, $a_{r_{\beta-1_{last}}}$, a_{s_α}, $a_{r_{\beta_{first}}}$, $a_{r_{\beta_{last}}}$, $a_{s_{\alpha+1}}$ steps and show that $a_{r_{\beta+1_{first}}} = a_{r_{\beta+1_{last}}}$. Let us assume, in the way of a proof by contradictions that $a_{r_{\beta+1_{first}}} \neq a_{r_{\beta+1_{last}}}$. We show that there is an $a_{s_{\alpha'}}$ step between $a_{r_{\beta+1_{first}}}$ and $a_{r_{\beta+1_{last}}}$.

By Lemma 2, between $a_{r_{\beta_{first}}}$ and $a_{r_{\beta_{last}}}$, there is at least one configuration, $c_{s_{\alpha'}}(x)$, for which $AltIndex_s \neq r\_index_{\beta-1_{last}}$, and at least one configuration, $c_{s_{\alpha''}}(x)$, for which $AltIndex_s \neq r\_index_{\beta+1_{first}}$.

Suppose that $\alpha' = \alpha''$. By a_{s_α}'s definition, line 3 of the Sender algorithm and the function *packet_set*(), the sender changes $AltIndex_s$'s value in step $a_{s_{\alpha'}}$ that occurs between $a_{r_{\beta+1_{first}}}$ and $a_{r_{\beta+1_{last}}}$. For the case of $\alpha' \neq \alpha''$, we use similar arguments and consider the sequence of all $c_{s_{\alpha'}}(x), c_{s_{\alpha''}}(x), \ldots$ configurations between $a_{r_{\beta_{first}}}$ and $a_{r_{\beta_{last}}}$ and their corresponding $AltIndex_s$'s values. By similar arguments to the case of $\alpha' = \alpha''$, any consecutive pair of

$AltIndex_s$ implies the existence of an a_{s_α} between $a_{r_{\beta_{first}}}$ and $a_{r_{\beta_{last}}}$. Thus, a contradiction.

Lemma 4 shows that between a_{r_β} and $a_{r_{\beta+1}}$, there is exactly one a_{s_α} step, and its proof follows similar arguments as the ones in Lemma 3.

Lemma 4 (proof appears in [10]). *Between a_{r_β} and $a_{r_{\beta+1}}$, the sender takes exactly one $a_{s_{\alpha+1}}$ step.*

Lemmas 3 and 4 facilitates the proof of Theorem 1.

Theorem 1 (S^2E^2C). *Within a constant number of asynchronous rounds, the system reaches a safe configuration (from which a legal execution starts). Moreover, following a safe configuration, Algorithm 2 delivers every new sent message batch within a constant number of asynchronous rounds.*

4 Conclusions

Self-stabilizing end-to-end data communication algorithms for bounded capacity dynamic networks have been presented in this extended abstract. The proposed algorithms inculcate error correction techniques for the delivery of messages to their destination without omissions, duplications or reordering. We consider two nodes, one as the sender and the other as the receiver. In many cases, however, two communicating nodes may act both as senders and receivers simultaneously. In such situations, acknowledgment piggybacking may reduce the overhead needed to cope with the capacity irrelevant packets that exist in each direction, from the sender to the receiver *and* from the receiver to the sender. Using piggybacking, the overhead is similar in both directions. The obtained overhead is proportional to the ratio between the number of bits in the original message, and the number of bits in the coded message, which is a code that withstands *capacity* corruptions. Thus, for a specific *capacity*, assuming the usage of efficient encoding, the overhead becomes smaller as the message length grows.

References

1. Afek, Y., Brown, G.M.: Self-stabilization over unreliable communication media. Distributed Computing 7(1), 27–34 (1993)
2. Awerbuch, B., Patt-Shamir, B., Varghese, G.: Self-stabilization by local checking and correction. In: FOCS, pp. 268–277. IEEE Computer Society (1991)
3. Bui, A., Datta, A.K., Petit, F., Villain, V.: State-optimal snap-stabilizing pif in tree networks. In: Workshop on Self-stabilizing Systems (ICDCS 1999), pp. 78–85. IEEE Computer Society (1999)
4. Chung, H.C., Robinson, P., Welch, J.L.: Brief Announcement: Regional Consecutive Leader Election in Mobile Ad-Hoc Networks. In: Scheideler, C. (ed.) ALGOSENSORS 2010. LNCS, vol. 6451, pp. 89–91. Springer, Heidelberg (2010)

5. Cournier, A., Dubois, S., Villain, V.: A snap-stabilizing point-to-point communication protocol in message-switched networks. In: 23rd IEEE International Symposium on Parallel and Distributed (IPDPS 2009), pp. 1–11 (2009)
6. Datta, A.K., Larmore, L.L., Piniganti, H.: Self-stabilizing Leader Election in Dynamic Networks. In: Dolev, S., Cobb, J., Fischer, M., Yung, M. (eds.) SSS 2010. LNCS, vol. 6366, pp. 35–49. Springer, Heidelberg (2010)
7. Dijkstra, E.W.: Self-stabilizing systems in spite of distributed control. Commun. ACM 17(11), 643–644 (1974)
8. Dolev, S.: Self-Stabilization. MIT Press (2000)
9. Dolev, S., Dubois, S., Potop-Butucaru, M., Tixeuil, S.: Stabilizing data-link over non-fifo channels with optimal fault-resilience. Inf. Process. Lett. 111(18), 912–920 (2011)
10. Dolev, S., Hanemann, A., Schiller, E.M., Sharma, S.: Self-stabilizing data link over non-fifo channels without duplication. Technical Report 2012:01, Chalmers University of Technology (2012) ISSN 1652-926X
11. Dolev, S., Israeli, A., Moran, S.: Resource bounds for self-stabilizing message-driven protocols. SIAM J. Comput. 26(1), 273–290 (1997)
12. Dolev, S., Schiller, E., Welch, J.L.: Random walk for self-stabilitzing group communication in ad hoc networks. In: PODC, p. 259 (2002)
13. Dolev, S., Schiller, E., Welch, J.L.: Random walk for self-stabilizing group communication in ad-hoc networks. In: 21st Symposium on Reliable Distributed Systems (SRDS 2002), pp. 70–79 (2002)
14. Dolev, S., Schiller, E., Welch, J.L.: Random walk for self-stabilizing group communication in ad hoc networks. IEEE Trans. Mob. Comput. 5(7), 893–905 (2006)
15. Dolev, S., Welch, J.L.: Crash resilient communication in dynamic networks. IEEE Trans. Computers 46(1), 14–26 (1997)
16. Flauzac, O., Villain, V.: An implementable dynamic automatic self-stabilizing protocol. In: ISPAN, pp. 91–97. IEEE Computer Society (1997)
17. Gouda, M.G., Multari, N.J.: Stabilizing communication protocols. IEEE Trans. Computers 40(4), 448–458 (1991)
18. Haeupler, B., Karger, D.R.: Faster information dissemination in dynamic networks via network coding. In: 30th Annual ACM Symposium on Principles of Distributed Computing (PODC 2011), pp. 381–390 (2011)
19. Ingram, R., Shields, P., Walter, J.E., Welch, J.L.: An asynchronous leader election algorithm for dynamic networks. In: 23rd IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2009), pp. 1–12 (2009)
20. Jelasity, M., Montresor, A., Babaoglu, Ö.: Gossip-based aggregation in large dynamic networks. ACM Trans. Comput. Syst. 23(3), 219–252 (2005)
21. Kuhn, F., Locher, T., Oshman, R.: Gradient clock synchronization in dynamic networks. Theory Comput. Syst. 49(4), 781–816 (2011)
22. Kuhn, F., Lynch, N.A., Oshman, R.: Distributed computation in dynamic networks. In: ACM Symposium on Theory of Computing (STOC 2010), pp. 513–522 (2010)
23. Kuhn, F., Oshman, R., Moses, Y.: Coordinated consensus in dynamic networks. In: 30th ACM Symposium on Principles of Distributed Computing (PODC 2011), pp. 1–10 (2011)
24. Spinelli, J.: Self-stabilizing sliding window arq protocols. IEEE/ACM Trans. Netw. 5(2), 245–254 (1997)
25. Tanenbaum, A.S.: Computer networks, 4th edn. Prentice-Hall (2002)

A.2 Adaptive Middleware for Advanced Control Systems

A.2.1 Lightweight Dependable Adaptation for Wireless Sensor Networks

"Lightweight Dependable Adaptation for Wireless Sensor Networks". L. Marques and A. Casimiro, Technical Report DI/FCUL, September 2012, Lisbon, Portugal.

This page is intentionally left blank.

Lightweight Dependable Adaptation for Wireless Sensor Networks

Luís Marques
lmarques@lasige.di.fc.ul.pt
FC/UL

António Casimiro
casim@di.fc.ul.pt
FC/UL

Abstract

Achieving dependable and real-time operation in Wireless Sensor Networks (WSNs) is a hard and open problem. This can be an obstacle for many applications, namely in the automotive and medical domains, particularly if safety-critical control is envisaged. To overcome the communication uncertainties that are intrinsic to wireless and dynamic environments, a generic approach is to constantly adapt to environment conditions. This requires appropriate solutions to characterize such conditions.

This paper contributes with a lightweight solution for a dependable characterization of network QoS metrics, which is appropriate to support dependable adaptation in WSNs. The proposed solution offers probabilistic guarantees, building on non-parametric stochastic analysis to achieve fast and effective results. The paper also provides an evaluation of the solution.

Keywords

Wireless Sensor Networks; dependability; adaptation; non-parametric; lightweight; QoS;

I. Introduction

Wireless Sensor Networks (WSNs) are used to sense and collect the state of physical entities. Applications that use this kind of networks can therefore obtain a representation of the state of such entities, which they use in monitoring and control functions. Each application has its own requirements regarding how faithful such representation must be, compared with the true state of the sensed entity. The problem that arises in WSNs is the lack of guarantees relative to its timeliness. This means that applications have no guarantees regarding the degree of synchronization, or the consistency, between the sensor data they are using and the real state of the environment. Consequently, they cannot also guarantee that monitoring and/or control functions are performed in a timely way.

The lack of real-time guarantees in WSNs results from a variety of factors. These include, among others, uncertainties introduced by medium access control protocols, transmission interferences common to wireless communication, dynamic changes occurring in the network, such as node failures or their movement, and the lack of real-time behavior of the nodes themselves.

In practice, some of these factors, which we may call architectural factors, are under the designer control and thus their influence on uncertainty can be minimized or eliminated. For instance, nodes can be designed and built using traditional real-time operating system techniques, with real-time scheduling, and communication protocols can be made deterministic with respect to a set of assumptions on the communication medium. Still, there are a number of factors out of the designer's control, which we may call external factors or perturbations, that occur with

This work was partially supported by EC, through project IST-STREP-288195 (KARYON) and by FCT through the Multiannual Funding and CMU-Portugal Programs and the Individual Doctoral Grant SFRH/BD/45344/2008.

uncertain patterns and lead to uncertainty, assumption violation and lack of real-time guarantees. Note that these perturbations can be characterized as part of the fault model. For instance, omission faults, caused by electromagnetic interference or by physical obstacles, or temporal faults, caused by network contention and transmission back-off.

One possible way of dealing with these perturbations and improving timeliness characteristics of a WSN is by employing redundancy. A given amount of perturbations can be tolerated through some combination of multiple nodes at different locations, communication over different channels, multiple copies of the same messages, etc. This is particularly true if fault independence can be assumed, which is often the case.

Fortunately, WSNs do typically possess great amounts of redundancy, which can be employed to increase the guarantees of successful communication. On the other hand, one of the main concerns in this type of networks is to save energy, and thus radio transmissions should be avoided as much as possible, as they are one of the main sources of energy consumption. That being, the operational objective of sensor networks should be that of satisfying application requirements through the use of the strictly necessary redundant resources. In other words, simply adding a fixed amount of redundant resources is not the best solution since it just pushes the bounds further, allowing a larger tolerance of perturbations, but not necessarily the adequate or sufficient one. Achieving some level of dependability and efficiency requires some effective way of dealing with the mentioned perturbations, whose distribution is a priori unknown and for which might not even be practical to define an arbitrary limit.

From a theoretical standpoint, not having a guaranteed limit on the amount of perturbations that can occur on WSNs makes it impossible to offer hard real-time guarantees. From a practical standpoint, even if the amount of perturbations never reaches the maximum amount that a sensor network can tolerate, it is not desirable to constantly use the maximum amount of redundant resources available. One alternative to offering strict hard real-time guarantees is to offer, instead, probabilistic guarantees. Therefore we consider that perturbations occur in a probabilistic manner. By analyzing the statistical behavior of such perturbations we can derive the necessary amount of redundancy that is necessary to satisfy application requirements, with a given probability. Since this behavior can change throughout time, as environment conditions change so must the system adapt, so that application requirements are continuously satisfied. In this paper we consider how to implement such statistical analysis and adaptation in a manner that is appropriate for WSNs.

We propose an approach for monitoring and raising awareness of communication latency that is based on non-parametric stochastic analysis. From a complexity perspective, the solution is extremely simple and thus appropriate for resource-constrained systems, such as WSNs. Despite its simplicity, it builds upon established theory on stochastic systems and allows probabilistic guarantees to be asserted to relevant temporal bounds. To conclude about the potential merits and the effectiveness of the proposed approach, we compared it with another solution, called Adaptare [1], which is based on complex and expensive stochastic analysis mechanisms. To ensure a fair comparison, we used the same data traces that were previously used for validating Adaptare, and we observed that our approach is very effective in general, although it exhibits some comparative limitations when the required probabilistic guarantees are very high. We believe that this trade-off may be acceptable in the context of WSNs, given that it still provides valuable monitoring data at an extremely low-cost.

The paper is thus organized. In the following section we provide some context and introduce various concepts. Then, Section III presents the idea of lightweight adaptation based on non-parametric analysis. In Section IV we evaluate and benchmark several aspects of the solution

and in Section V we present related work. Section VI concludes the paper.

II. Problem Context

Dealing with predictability or real-time requirements in WSNs is a difficult and open problem, which may be addressed from several perspectives. While the provision of strict real-time requires satisfying very stringent assumptions, focusing on the provision of some Quality of Service (QoS) is a reasonable alternative to address the problem. In fact, several authors have explored the issue of providing QoS guarantees in WSNs, which we review in more detail in Section V. Our work is also developed in this context.

Proposed solutions explore topological aspects, such as end-to-end path discovery, resource reservation along discovered paths and path recovery from topological changes, build on proper scheduling of real-time traffic (e.g. decentralized EDF scheduling), and define efficient network protocols to support the necessary QoS features. Such solutions deal mainly with problems that are architectural and internal to WSNs, namely how to match the low-power and dynamic nature of WSNs with the strict requirements of various types of real-time communication. Another body of work has explored how to make radio communication efficient and resilient. Employed techniques include detecting interferences [2] and dynamically changing channels to avoid them [3], [4].

But no matter the techniques and approaches that may be used to endow WSN-based systems with better predictability and greater ability of dealing with user QoS requirements, we believe that the likely occurrence of perturbations still makes it inappropriate to specify fixed upper bounds on system variables, such as the maximum number of omission faults, latencies, query load, and so on. Because of that, monitoring and adaptation appear to be relevant techniques to deal with this uncertainty, contributing to achieve systems that perform more closely to the allowed environment conditions and thus leading to improved QoS.

Quite clearly, one fundamental issue in this context is how QoS is defined. Different metrics (or estimators of metrics) can be considered, and they should be in some way related to application requirements. Possible metrics and estimators may include: rate of arrival of updates, number of duplicate packets received, jitter in the arrival of updates, packet loss rate, Packet Error Rate (PER), Received Signal Strength Indicator (RSSI), and single hop or end-to-end latency.

From the application perspective, QoS requirements tend to be less functional. For instance, for the correctness of monitoring and control applications what is essential is to ensure that real-time sensor data is accurate, close (within some error interval) to the real value of the monitored or controlled entity. We refer to this as a requirement for *perception quality*, that is, how accurately the application perceives the reality. Given the uncertainties affecting WSNs, and being inappropriate to assume fixed upper bounds for network latencies, the notion of perception quality encompasses both the acceptable error for sensor data and the probability that this error bound will be secured at run time. In summary, high perception quality means ensuring a very small perception error with a very high probability. We also say that the assumed error bound is secured with a certain *coverage*, i.e. the probably of the observed value being within the required error margin.

However, as mentioned above, in practice it is usually necessary to translate higher-level, possibly non-functional, application requirements to observable metrics. In this paper we consider that the propagation latency is the relevant QoS metric. Without loss of generality, we consider as a simplifying assumption that there is an inverse proportional relation between end-to-end latency and perception quality.

III. Lightweight Dependable Adaptation

We consider that the environment behaves as a stochastic process. That means that each relevant QoS property of the system is defined (at a given instant) by a random variable. Such random variable describes the values that the property can take, and with which probabilities — that is, its probability distribution. As time progresses and the environment changes, new random values will take the place of old ones, in ways characterized by the stochastic process.

For adaptation to occur it is necessary to characterize environment conditions. To deduce at a given time what is the state of the environment it is necessary to first sample its behavior. Using a sample we can then make inferences regarding the state of the system, and in particular estimate the probability distributions of the random variables.

In the same way that there are an infinity of numbers, there are also an infinity of possible probability distributions. In practice, some numbers are particularly common or important and thus become well-known, such as the numbers 1, 2, π or $\sqrt{2}$. Likewise, there are various well-known families of probability distributions, such as the Normal, Exponential and Poisson distributions. Despite their name, each well-known "distribution" does not specify a particular probability distribution function. Instead, they have parameters which control properties of the distribution, such as their average value or dispersion, giving rise to an infinity of fully specified probability distribution functions.

Traditionally, the process of statistical inference is done through parametric methods. That is, methods which assume that the sample values are the result of a random variable whose distribution function belongs to a well-known family, but for which the parameters must be estimated. This assumption, when valid, brings several benefits. It allows estimators to have more *statistical power*, producing estimates that are more accurate or that otherwise could not even be determined. The simplicity and benefits of parametric methods made them widely applied, and they are often used even when no distribution perfectly matches the studied phenomenon. However, when the assumptions made by parametric statistics do not hold the results can be extremely misleading. Because parametric methods strongly depend on their assumptions they are not statistically *robust*.

Prior work applying statistical adaptation [1] has tried to guarantee the parametric assumptions by employing runtime statistical diagnostics. That involves checking the *goodness of fit* of the observations of a random variable against the possible distributions. If the fit is good enough (i.e., very unlikely to be due to chance) then parametric methods can be used safely. Unfortunately, the disadvantages of such strategy are particularly onerous for WSNs. Most importantly, the necessary statistical tests are computationally expensive, making them unfeasible for limited devices such as sensor nodes.

As an alternative, we propose to achieve lightweight dependable adaptation through the use of non-parametric statistics — that is, methods which are distribution-free. Non-parametric methods are robust, since they apply to all probability distributions, making them fit for the wide variety of WSN scenarios and protocols. They are also lightweight, requiring only very simple computations, thus conserving energy. And, despite their simplicity, they can also be surprisingly effective.

A. Non-parametric statistical analysis

The fundamentals of how non-parametric statistics are used in the proposed solution, and how they differ from the parametric methods of previous work, can be explained by comparing the two histograms in Figure 1.

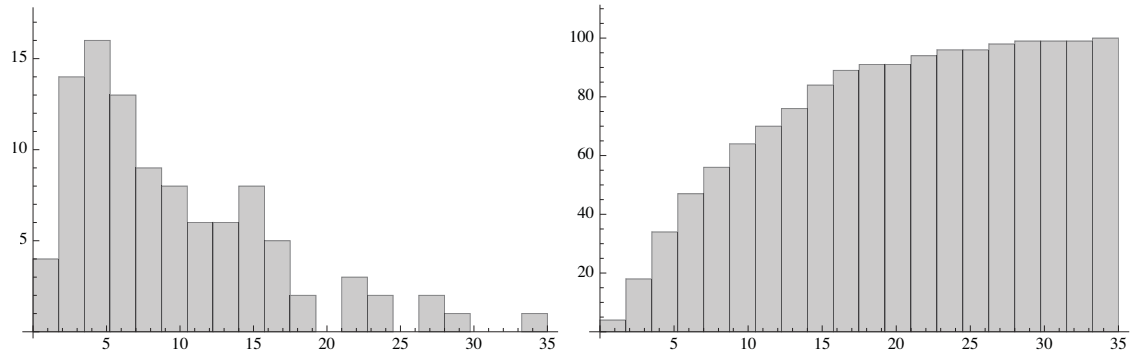


Figure 1: Two histograms for one same sample: normal and cumulative

The left histogram of Figure 1 shows twenty bins, each of which counts the frequency of values occurring in a particular interval. By glancing at such an histogram we can informally check if there is a good fit between the sample values and well-known distributions. In this case we do not immediately recognize some of the most well-known distributions: the distribution is asymmetric, so it is not a Normal distribution; after the third bin the histogram resembles an Exponential distribution, but the first two bins do not match; it is also not quite the shape of a Poisson distribution with a low λ value.

If we included a near-infinite amount of distributions against which to test the goodness of fit of our sample values then, eventually, we would find a good match. But such method would not achieve good results: the performance of testing huge amounts of distributions would not be sustainable on a sensor node. We would also match many possible distributions, so we would not know which to use to better predict yet unobserved values and their probabilities.

An alternative to matching the sample values to a theoretical model, and then using the properties of the theoretical model to drive adaptation, is to use only the statistical properties of collected sample itself, free of distribution assumptions. The right histogram of Figure 1 illustrates that approach, by virtue of being a *cumulative* histogram.

In a cumulative histogram the bins count the occurrences (or relative frequency) of values that fall in the range of those bins *or the preceding ones*. It thus becomes easier to see what percentage of the sample values is equal or smaller than some other value. For instance, we see that about 80% of the sample values are equal or smaller than 15. If this were a sample of latencies in the system then we could use this non-parametric analysis as statistical evidence to drive the adaptation. For instance, we could make sensor nodes wait for some event only up to 15 time units, and that way know that timing failures would occur only for approximately 20% of the events. That is, we would be using the empirical distribution to guide the process of adaptation.

Directly using the empirical distribution to make statistical inferences about the sampled system disregards the problem of sampling error. For instance, by using the empirical distribution present in Figure 1 we might be tempted to conclude that a time bound of 35 time units would be enough to receive 100% of events, and that way avoid timing failures. Of course, this ignores the possibility that there might be a small percentage of events which have higher latencies but, by chance, were not captured in the sample.

One possible solution then is to increase the sample size. As the number of sample values approaches infinity the statistics of the sample converge to the true values of the population.

Alas, it is not practical to use nearly infinite sample sizes. For one, it would be computationally expensive, particularly in sensor nodes. Also, as the environment changes the older values would no longer reflect its state.

The solution, therefore, must entail taking into account the sampling error and estimating how much of the population really is equal or less than a given sample value. In fact, this is not done for the sample value per se but, instead, for its *ordinal ranking* in the sample. So, in the example given before, 35 time units was estimated to cover 100% of the population not because the value was 35 but because that was the n^{th} order statistic of a sample with n values (i.e. the maximum)[1]. What we want, then, is to make such inferences but taking into account the sampling error, and for a generic order statistic. For that we can use the Beta distribution.

The k^{th} order statistic of the Uniform distribution can be described by a Beta(α, β) distribution with parameters $\alpha = k$ and $\beta = n + 1 - k$. The mean of the Beta distribution is given by:

$$\text{mean} = \frac{\alpha}{\alpha + \beta} \tag{1}$$

Therefore, the mean for the k^{th} order statistic is given by:

$$\text{mean} = \frac{k}{n + 1} \tag{2}$$

Consider then a sample for a Uniform(0, 1) distribution, with $n = 20$. In that case the order statistic $k_{(20)}$ would take, on average, the value of $\frac{20}{21}$, or approximately 0.95. If we used the order statistic as an upper bound we could then infer to be covering, on average, 95% of the population, instead of the 100% we would expect while not taking into account the sampling error.

We could take such a direct conclusion because for a Uniform(0, 1) distribution the rank of a quantile is equal to the value of the quantile itself. For instance, the quantile $q_{0.25}$ represents the value x such that the probability of a random variable being less than or equal to x is 25%. For a Uniform(0, 1) distribution the value x is also 0.25, the same as the rank.

For other distributions the values are not uniformly apportioned between 0 and 1. Nevertheless, the quantiles *are*, by definition, uniformly distributed between 0 and 1. As such, it is still correct to use a Beta distribution to infer what is the average rank of the quantile to which an order statistic corresponds. The next section explains how to generically apply this to a monitoring solution.

B. Monitoring Method

The overall process of adaptation is done by sampling the environment, inferring the state of the system from the sample values and adapting to that environment. The adaptation itself can start as soon as enough values are collected. The necessary amount of sample values depends on the desired (or target) average coverage.

On the one hand, it depends on the *minimum* desired coverage. The n^{th} order statistic has an average value of $n/(n + 1)$. As such, with 1 sample value it is possible to start the adaptation for minimum coverages of up to $1/(1 + 1) = 50\%$, with 2 values for minimums of up to $2/(2 + 1) = 66.7\%$, with 3 values for up to 75%, and so on.

[1]The i^{th} order statistic of S is the i^{th} smallest element of S. Such an element is said to have *rank* i.

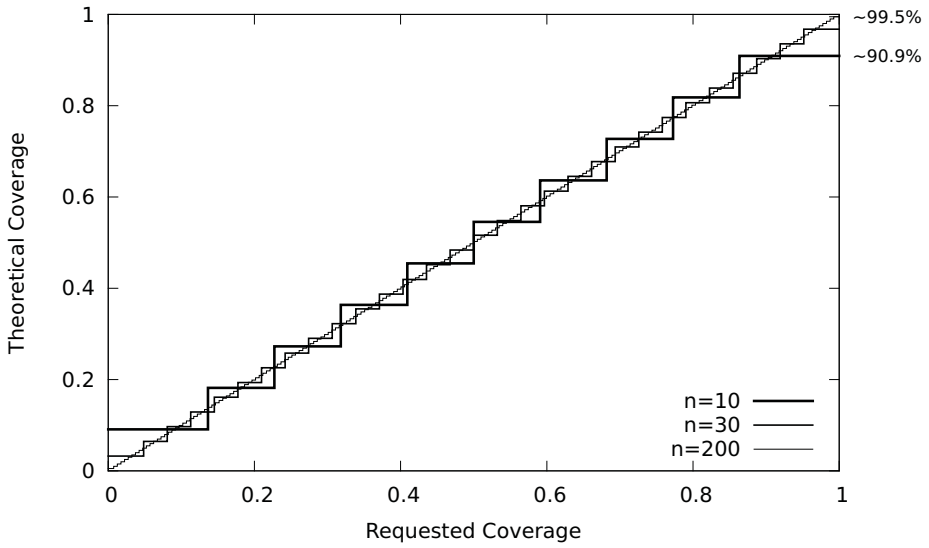


Figure 2: Non-parametric average coverages for different sample sizes

On the other hand, the number of required sample values also depends on how *accurately* the coverage must be matched. Figure 2 illustrates (with lines of different widths) the impact of choosing different sample sizes on the average coverage obtained by applications.

Ideally, the process of adaptation would result in applications obtaining a coverage equal to the one asked by the application. What we see is that with a sample of only 10 values there are large gaps between various coverages the application can ask for and the coverage that is obtained. Also, even by choosing the largest value from the sample the application will only obtain (on average) a coverage of 90.9%, which is not very high. On the other hand, for a sample with 30 values the gaps between the requested coverages and the average ones become more reasonable. Finally, for a large sample of 200 values we observe that the adaptation process starts to approach an optimal diagonal line, and can reach a high coverage of about 99.5%.

Having a sufficient number of sample values, we then order the sample and chose the value whose rank best matches a target average coverage. If we take equation 2, substitute the mean by a target coverage C and solve for the rank k we get:

$$k = C * (n + 1) \tag{3}$$

This should give us the rank of the sample value which matches the target coverage C. Unfortunately, the equation does not consider two problems. One is that the derived rank might be non-integral, if no rank exactly matches the requested average coverage. Another is that the computed rank may be out of bounds, if the sample size does not achieve a low enough or high enough average coverage.

Algorithm 1 takes the base calculation of equation 3, performs the necessary adjustments to deal with those two problems, and returns the sample value whose rank best matches the requested coverage. It is also adjusted for a zero-based indexing of the sample array, for additional clarity of implementation.

While coverages of exactly 0% and 100% would never be valid for samples of finite size, Algorithm 1 accepts target coverages with values of 0 and 1. Since floating point numbers have

Algorithm 1 Bound estimation algorithm

Input: ordered sample array $sample[n]$
Input: desired average coverage C (floating point)
Output: time bound (with average coverage C)
1: assert ($C \geq 0.0$ **and** $C \leq 1.0$)
2: $index \leftarrow \lfloor (C * (n+1)) - 0.5 \rfloor$
3: **if** $index < 0$ **then**
4: $\quad index \leftarrow 0$
5: **else if** $index \geq n$ **then**
6: $\quad index \leftarrow n - 1$
7: **end if**
8: **return** $sample[index]$

finite dynamic range, similar values may happen to be rounded to those limit values. With the addition of the guard conditions on lines 3–7 such cases are gracefully handled anyway.

After the minimum sample size is attained we can continue to append new values to the sample. The maximum sample size should be chosen according to the limitations of the hardware and the dynamics of the environment. For rapidly changing environments smaller sample sizes must be chosen, to ensure that the adaptation process does not reflect stale views of that environment. Section IV-C evaluates the empirical impact of different sample sizes. Once the sample is filled to its maximum the oldest values should be discarded before adding new values. This results in a sliding window.

IV. Evaluation

A. Complexity Analysis

Non-parametric order statistics entail selecting a k^{th} smallest value from an ordered sample. Our sample is a constantly changing (unordered) sliding window, updated from the latencies of received packets, up to a size n. This sliding window can be implemented as circular buffer, with an array for the sample values, a pointer to the beginning of the window and a fill counter. This way, adding new values to the sample takes $O(1)$ time, with a very low constant.

An obvious way to select the k^{th} smallest values from the unordered sample is to create a copy of the sample, sort it, and select the k^{th} value. Such solution has a worst-case time complexity of $O(n \log n)$ and a space complexity of $O(n)$. For small sample sizes this cost can be reasonable, even on a limited device such as a sensor node. A sorting function can generally be reused from the sensor node's software library, not incurring an extra code space penalty.

For larger sample sizes a better alternative is to use the selection algorithm first described in [5] and more clearly explained in [6], which has a $O(n)$ worst-case time complexity and $O(1)$ space complexity. This algorithm is, on average, less efficient than Hoare's Selection Algorithm, but is safer for real-time sensors, since the later algorithm has a nonlinear worst-case time complexity, making it less predictable.

If sublinear time complexity is required then a red-black tree can be used, at the expense of additional memory and an increase in insertion time. Updating the tree-based ordered sample takes time with $O(\log n)$ complexity, instead of $O(1)$, but the order statistic can be found in $O(\log n)$ time.

B. Performance Magnitude

Today's sensor nodes are equipped with very rudimentary CPUs. While that is expected to eventually change, for now that is a reality that must be dealt with.

Besides being slow, the CPUs of sensor nodes also typically lack a floating point unit. While it is possible to add floating point software emulation libraries, such libraries would make computations even slower. They would also take up a lot of code space, which is very limited on sensor nodes, generally on the order of a few kilobytes. Since the general bound selection algorithm provided in Algorithm 1 relies on floating point, for practical implementations it must be optimized.

One possible alternative is to use fixed-point arithmetic. For instance, instead of specifying a coverage of 85% as the floating point number 0.85 it can instead be specified by the integer 85. Such alternative algorithm can be implemented using an integer division, plus a few basic operations.

Still, even an integer division is a complex operation. In fact, the CPU of the popular MICA2 mote (an Atmel AVR ATmega128) does not even implement a native integer divide instruction. One possible solution is to use a software division routine. For that CPU, a 16 / 16 bit division, with 16 + 16 bit signed result can be implemented in 39 code words, taking 255 execution cycles to compute (16 bits allows for samples with more than 256 values).

Another possibility, which avoids division instructions, is to receive the input coverage as an integer in the range [0, 127], instead of [0, 100%]. Algorithm 2 exemplifies how such an optimized bound estimation algorithm can be implemented.

Algorithm 2 Fixed-point bound estimation algorithm
(coverage range [0, 1] mapped to $\{0, 1, ..., 127\}$)

Input: ordered sample array $sample[n]$
Input: desired average coverage C (integer)
Output: time bound (with average coverage C)

1: assert ($C \geq 0$ **and** $C \leq 127$)
2: $index \leftarrow ((C * (n + 1)) - 128)$
3: $index \leftarrow index/128$
4: **if** $index < 0$ **then**
5: $\quad index \leftarrow 0$
6: **else if** $index \geq n$ **then**
7: $\quad index \leftarrow n - 1$
8: **end if**
9: **output** $sample[index]$

While Algorithm 2 shows a division by 128 at line 3, that division can be implemented as a right shift instruction, and thus be computed very quickly.

With the presented simplifications, the algorithm requires on the order of less than 100 simple instructions. To this we must add the additional cost of sorting the sample, or of using a linear time selection algorithm. Assuming the worst-case cost of $5.4305*n$ comparisons for the selection algorithm reported in [5], also assuming about 5 cycles per comparison, and a sample size of 30 values, we would be adding 815 cycles. That is still well under 1000 cycles total. We can

therefore estimate a performance magnitude of more than 8,000 bound estimations per second, for a simple 8 MHz CPU.

As such, we conclude that the proposed adaptation algorithm can be made lightweight enough for Wireless Sensor Networks. It is not a bottleneck and should not have a significant impact on energy consumption.

C. Adaptation Benchmark

The statistical properties in which we rely for monitoring the environment and driving the adaptation process are theoretically guaranteed. In practice, they only hold if the assumptions in which they are based can be relied upon. Particularly, we assume that the behavior of the environment is stochastic, with limited dynamics.

In this section we benchmark an adaptation process, which uses the proposed bound estimation algorithm, to determine the empirical effectiveness of the non-parametric method. The benchmark verifies if, given a stream of real network latencies, the estimated bounds are able to maintain the desired average coverage. We selected the following collection of network latency traces for the benchmark:

- **Inmotion**: FTP file transfers between cars traveling at various speeds and an 802.11b access point;
- **Umass**: Wireless traces from University of Puerto Rico, using laptops over various distances;
- **Dartmouth**: Wireless traces from Dartmouth College;
- **LBNL/Datcat**: Traffic of an enterprise network from Lawrence Berkeley National Laboratory (LBNL);
- **RON**: Latency samples from the RON (Resilient Overlay Network) testbed;

These traces were previously used to benchmark Adaptare, an adaptation framework based on parametric methods, and are referenced in [1]. Using these specific traces has some advantages. Because we are reusing the same traces that were used to benchmark Adaptare, it allows us to accurately compare our results, with regards to the obtained coverage. The traces also exhibit a great variety of behaviors. Since the field of WSNs is still evolving rapidly (with no de facto standard for MAC and network protocols), and sensor networks can be applied in very different scenarios, it is important to test if the proposed method is effective for a variety of latency patterns.

In Figure 3 we plotted the chosen traces, showing the evolution of latencies throughout time. No scale was included for simplicity, but the figure allows us to visually confirm that the traces *do* exhibit a wide variety of patterns, and therefore are suitable to validate the proposed method.

There is necessarily a trade-off (all things being the same) between higher bounds, which can guarantee a higher coverage but will result in lower performance, and lower bounds, which provide lower coverages but also allow higher performances. As such, this benchmark tries to evaluate the capability of the proposed algorithm to estimate bounds which best match a requested coverage, with its implied trade-off.

One trial of the benchmark was performed as follows. We evaluated both the proposed non-parametric method and the Adaptare framework, for each trace. We selected the most favorable sample size to Adaptare, 30 values [1], to assure a meaningful comparison, despite that sample size limiting the maximum coverage of the non-parametric method to about 96.77%. For each method, we examined a range of target coverages, from 1% to 99%, in 1% increments. For each trace, the first 30 values were used to fill the sample, leaving the other trace values to test the bound estimation. For each of the remaining trace values we repeated these steps: 1)

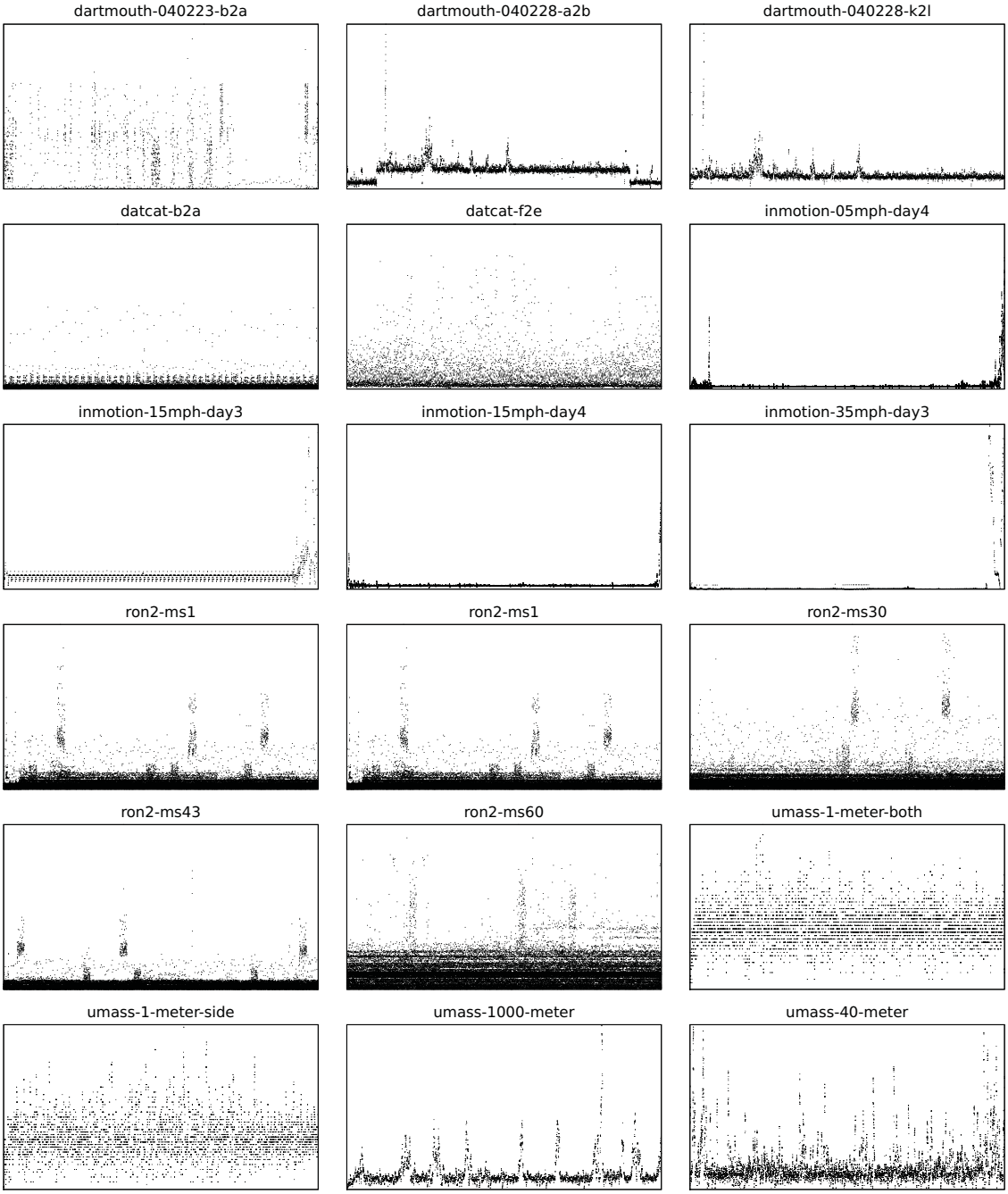


Figure 3: Plots of recorded latencies, exhibiting different dynamics

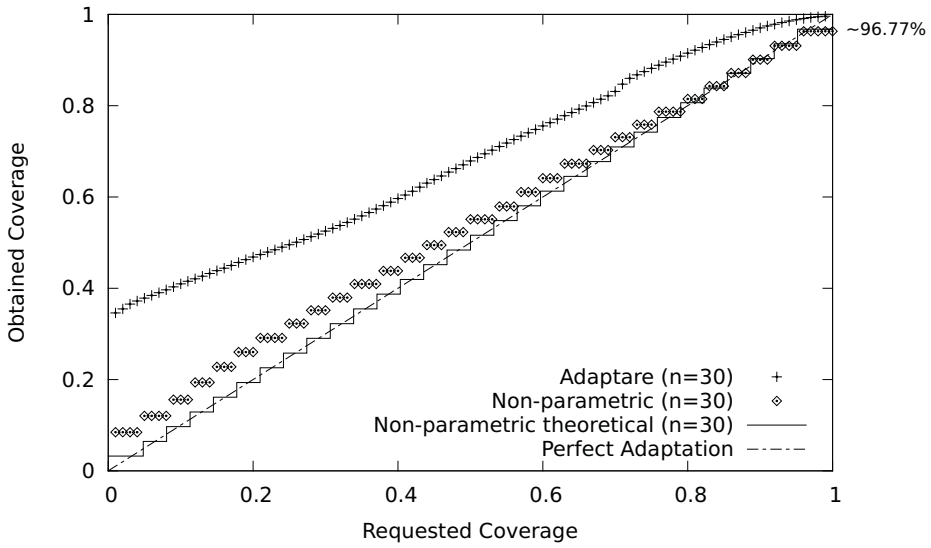


Figure 4: Empirical average coverages ($n = 30$)

estimated latency bounds using both the non-parametric method and Adaptare; 2) increased timeout counters for each method, if the trace value exceeded the estimated bound; 3) removed the oldest value from the sample; 4) added the new trace value to the sample. At the end of the process we computed the obtained average coverage for each method, using the formula:

$$\text{coverage} = 1 - (\text{timeouts}/\text{tested\_samples}). \tag{4}$$

Figure 4 plots an average of the coverages that were obtained for all the tested traces. Looking at this figure we can take the following conclusions:

- Both the proposed non-parametric method and the Adaptare framework are able to adapt to different target coverages, increasing the empirical coverage for higher target coverages;
- Despite its simplicity, the non-parametric method comes much closer to matching the target coverage (the "perfect adaptation" line), especially for higher coverages;
- Due to the small sample size the non-parametric method exhibits a significant staircase effect. These is the result of choosing the same order statistic for different target coverages, and is to be expected;
- Also due to the chosen sample size, the non-parametric method is unable to achieve the highest target coverages. Contrarily, the Adaptare framework can deduce statistical properties of the environment not directly present in the sample, to successfully estimate bounds for the highest coverages.

Figure 4 summarizes the results for the tested traces, by presenting an average of those results. It is legitimate to question what variation there is in the individual results. Figure 5 and Figure 6 present the best and the worst individual results, respectively.

We see in Figure 5 that the non-parametric method achieves nearly perfect adaptation, while the Adaptare framework retains a behavior similar to the average. In Figure 6 we observe that both for the non-parametric method as for the Adaptare framework there is some disruption in

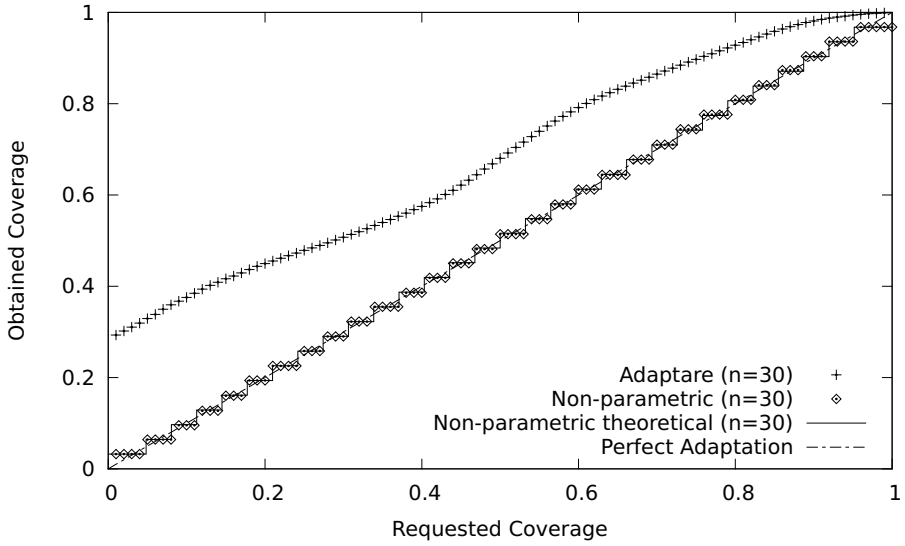


Figure 5: Empirical coverages: 'ron2-ms60-30'

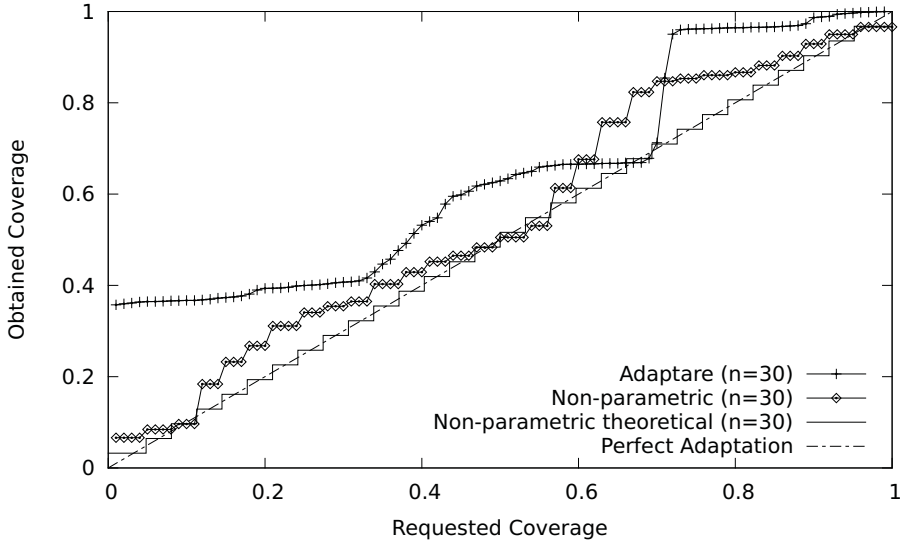


Figure 6: Empirical coverages: 'inmotion-15mph-day3-30'

the adaptation, compared to the averages. Still, the non-parametric method is on average closer to the target coverage.

Observing Figure 3 we note that the latency plots for the RON traces do exhibit a seemingly stochastic behavior, while the traces for Inmotion have repeating patterns (e.g. inmotion-15mph-day3) and other forms of low randomness. Also, while the RON traces reflect the prolonged interaction of many different network nodes, the Inmotion traces capture brief file transfers between a single mobile node and a base station, distant from other active network nodes. Therefore, we believe that the differences in the obtained results reflect how well the different scenarios meet our assumptions.

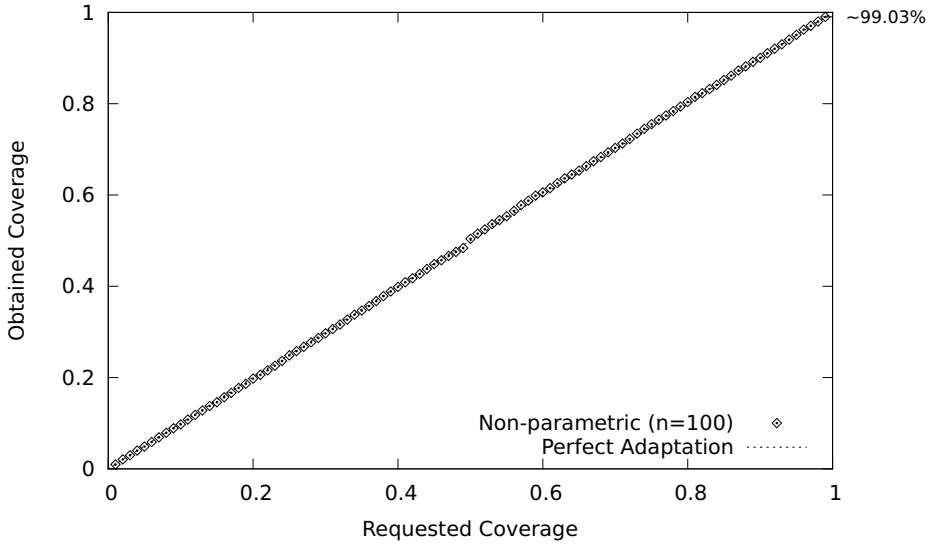


Figure 7: Empirical average coverages ($n = 100$)

Since we aim to apply the presented adaptation methods to WSNs, which typically have many nodes and operate in open environments, we believe that these results validate the adaptation capabilities of the non-parametric method. In scenarios which closely match our assumptions the results show a nearly perfect adaptation, while in adverse scenarios the method still achieves coverages generally close to the target. For the favorable scenarios the only significant limitations that were found were due to the limited sample size. For that reason, another trial of the benchmark evaluated a larger sample size.

Figure 7 presents the coverages of the non-parametric method, averaged from all traces, using a large sample: 100 values. We observe that, as expected, a larger sample size raised the maximum achievable coverage, to more than 99%. The larger sample size also removed the staircase effect and resulted in empirical coverages much closer to the target. For instance, the maximum coverage obtained was 99.07%, close to the theoretical average of 99.01%. These results also validate our assumption of limited environment dynamics, since with a larger sample (i.e. with one which includes older values) we still achieved a very good adaptation — in fact, better than with a smaller sample.

V. Related Work

There are two important lines of work related to this paper. One concerns protocols and other architectural mechanisms which aim to achieve real-time or QoS-driven behavior in WSNs. Another one deals with reliable wireless communication, including the physical mediums, radio propagation, link quality, interferences and similar issues. In this section we review related work on these subjects.

A. QoS and Real-Time

Two surveys of WSNs which summarize various QoS and real-time related issues can be found in [7] and [8]. These issues include the network architecture, platform and sensor hardware,

collaboration and coordination protocols and middleware, as well as MAC, network and transport layer protocols.

The work presented in [9] focuses on issues related to traffic QoS, and on various routing and MAC protocols that were proposed to address QoS requirements.

For exploiting the redundancy of WSNs, multi-path protocols have been proposed, for instance in [10]. Also relevant for QoS provisioning, the *Multi-Path and Multi-Speed Routing Protocol* (MMSPEED) is described in [11]. It tries to achieve QoS differentiation in two independent domains: *timeliness* and *reliability*. To achieve the necessary reliability, MMSPEED also attempts to exploit the redundancy of dense sensor networks and uses multipath forwarding. Differently from our approach, MMSPEED uses rudimentary mathematical calculations that are not supported by a statistical method. The end-to-end communication success probability is extrapolated at each node by keeping track of just a packet loss rate and adjusting for error probability for the path's number of hops.

Similarly, the RAP architecture [12] tries to provide real-time communication in large-scale WSNs. By giving higher priority to packets with longer routing distances the architecture's scheduling reduces deadline misses for packets far-away from their destination.

The paper in [13] presents a middleware mechanism that allows users of the network to request different QoS requirements. It also considers that stricter QoS requirements can be achieved by exploiting the redundancy of WSNs.

Various QoS aware routing protocols exist, which try to achieve the necessary quality while still conserving energy. Examples include the protocols published in [14] and [15].

In this paper we implicitly assumed that a sensor network implementing the proposed mechanism would be able to measure the relevant QoS metrics, such as packet latencies. One common way to achieve those capabilities is to have a notion of global time. The work in [16] proposes a way to achieve efficient global clock synchronization for WSNs.

B. Reliable Wireless Communication

The study of reliable wireless communication in the context of WSNs has investigated several issues. One problem that has been identified is that, as the popularity of WSNs and other networks grows, the existing electromagnetic spectrum will get crowded [17].

To some extent the problem is already being observed in existing environments. Experiments referred in the work of [4] identify a packet loss between 3% and 58% for a multi-hop 802.15.4 sensor network, sharing the 2.4 GHz frequency with a WiFi network, depending on the sending rate of the competing WiFi network and the length of the WSN routing path.

The problem can be dealt from different perspectives. One, particularly relevant to this paper, is to allow the networks to cooperatively allocate resources to control QoS among themselves [17]. Another solution, also relevant for reliability, is to increase the redundancy of communication, allowing nodes to operate over various communication channels and frequencies [18], [19] or even providing them with multiple radios [20].

Another issue that has received attention is understanding and modeling the propagation of radio waves. Higher accuracy models have been devised [21], [22], which are relevant for highly tuned QoS-driven architectures and for simulations to validate their results [23]. Correct models are necessary to understand the causes of packet delivery failure [24], the probability of failure [25] and consequences on multi-path routing [26].

Also relevant to dependable adaptation is how to assess the wireless link quality [27], [28], [29] and how to detect the occurrence of radio interferences [30], which is necessary to trigger

adaptation.

VI. Concluding Remarks

We presented a lightweight solution for the dependable characterization of network QoS metrics, based on non-parametric statistics, which allows WSNs and their applications to dependably adapt to changing conditions.

We evaluated the complexity and performance of the proposed solution, concluding it to be lightweight enough for WSNs. We also benchmarked its adaptation effectiveness. The benchmark validated the capability of the proposed technique to successfully adapt in a variety of real-world conditions.

Acknowledgment

The authors would like to thank Teresa Alpuim for her support.

References

[1] Dixit, M., Casimiro, A., Lollini, P., Bondavalli, A., Verissimo, P.: Adaptare: Supporting automatic and dependable adaptation in dynamic environments. ACM Transactions on Autonomous and Adaptive Systems **(to appear)** (2011) also as Technical Report DI/FCUL TR-09-19.

[2] Zhou, G., He, T., Stankovic, J.A., Abdelzaher, T.: Rid: Radio interference detection in wireless sensor networks. In: in INFOCOM. (2005)

[3] Xu, W., Trappe, W., Zhang, Y.: Channel surfing: defending wireless sensor networks from interference. In: Information Processing in Sensor Networks. (2007) 499–508

[4] Musaloiu-elefteri, R., Terzis, A.: Minimising the effect of wifi interference in 802.15.4 wireless sensor networks. International Journal of Sensor Networks **3** (2008) 43–54

[5] Blum, M., Floyd, R.W., Pratt, V.R., Rivest, R.L., Tarjan, R.E.: Time bounds for selection. Technical report, Stanford, CA, USA (1973)

[6] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. The MIT Press, New York (2001)

[7] Martínez, J.F., Garcí, A.B., Corredor, I., López, L., Hernández, V., Dasilva, A.: Qos in wireless sensor networks: survey and approach. In: Proceedings of the 2007 Euro American conference on Telematics and information systems. EATIS '07, New York, NY, USA, ACM (2007) 20:1–20:8

[8] Akyildiz, I.F., Melodia, T., Chowdhury, K.R.: A survey on wireless multimedia sensor networks. Computer Networks **51** (2007) 921–960

[9] Younis, M., Akkaya, K., Eltoweissy, M., Wadaa, A.: On handling qos traffic in wireless sensor networks. Hawaii International Conference on System Sciences **9** (2004) 90292a

[10] Li, S., Neelisetti, R., Liu, C.: Efficient multi-path protocol for wireless sensor networks. International Journal of Wireless and Mobile Networks (IJWMN) (Jan 2010)

[11] Felemban, E., gun Lee, C., Ekici, E., Boder, R., Vural, S.: Probabilistic qos guarantee in reliability and timeliness domains in wireless sensor networks. In: Proc. of the IEEE Infocom. (2005) 2646–2657

[12] 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)

[13] Sharifi, M., Taleghan, M.A., Taherkordi, A.: A middleware layer mechanism for qos support in wireless sensor networks. Mobile Communications and Learning Technologies, Conference on Networking, Conference on Systems, International Conference on **0** (2006) 118

[14] Mahapatra, A., Anand, K., Agrawal, D.: Qos and energy aware routing for real-time traffic in wireless sensor networks. Computer Communications (Jan 2006)

[15] Akkaya, K., Younis, M.: Energy and qos aware routing in wireless sensor networks. Cluster Computing (Jan 2005)

[16] Li, Q., Rus, D.: Global clock synchronization in sensor networks. IEEE Transactions on Computers **55**(2) (2006)

[17] Zhou, G., Stankovic, J.A., Son, S.H.: Crowded spectrum in wireless sensor networks. In: Proceedings of Third Workshop on Embedded Networked Sensors (EmNets. (2006)

[18] Xu, W., Trappe, W., Zhang, Y.: Channel surfing: defending wireless sensor networks from interference. In: Information Processing in Sensor Networks. (2007) 499–508

[19] Kim, Y., Shin, H., Cha, H.: Y-mac: An energy-efficient multi-channel mac protocol for dense wireless sensor networks. In: Information Processing in Sensor Networks. (2008) 53–63

[20] Ansari, J., Zhang, X., Mähönen, P.: Multi-radio medium access control protocol for wireless sensor networks. In: Conference On Embedded Networked Sensor Systems. (2007)

[21] Scott, T., Wu, K., Hoffman, D.: Radio propagation patterns in wireless sensor networks: new experimental results. In: Proceedings of the 2006 international conference on Wireless communications and mobile computing. 857–862

[22] Zhou, G., He, T., Krishnamurthy, S., Stankovic, J.A.: Models and solutions for radio irregularity in wireless sensor networks. ACM Transactions on Sensor Networks **2** (2006) 221–262

[23] Martınez-Sala, A., Molina-Garcıa-Pardo..., J.: An accurate radio channel model for wireless sensor networks simulation. Journal of Communications and Networks (Jan 2005)

[24] Srinivasan, K., Dutta, P., Tavakoli, A., Levis, P.: Understanding the causes of packet delivery success and failure in dense wireless sensor networks. Technical report, In Technical report SING-06-00 (Jan 2006)

[25] Cerpa, A., Wong, J.L., Kuang, L., Potkonjak, M., Estrin, D.: Statistical model of lossy links in wireless sensor networks. In: Information Processing in Sensor Networks. (2005) 81–88

[26] Cerpa, A., Wong, J.L., Potkonjak, M., Estrin, D.: Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In: Mobile Ad Hoc Networking and Computing. (2005) 414–425

[27] 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

[28] Baccour, N., Koubaa, A., Jam Atextcenta, M.B., Youssef, H., Zuniga, M., Alves, M.: A comparative simulation study of link quality estimators in wireless sensor networks. (2009) 1–10

[29] Xu, Y., chien Lee, W.: Exploring spatial correlation for link quality estimation in wireless sensor networks. In: in Proc. IEEE PerCom. (2006) 200–211

[30] Zhou, G., He, T., Stankovic, J.A., Abdelzaher, T.: Rid: Radio interference detection in wireless sensor networks. In: in INFOCOM. (2005)

A.2.2 Programming abstractions and middleware for building control systems as networks of smart sensors and actuators

"Programming abstractions and middleware for building control systems as networks of smart sensors and actuators". Sebastian Zug, Michael Schulze, Andre Dietrich, Joerg Kaiser, September 2010, ETFA 2010 - 15th IEEE International Conference on Emerging Technologies and Factory Automation, Bilbao, Spain.

This page is intentionally left blank.

Programming abstractions and middleware for building control systems as networks of smart sensors and actuators

Sebastian Zug, Michael Schulze, André Dietrich, Jörg Kaiser
Universität Magdeburg
Department for Distributed Systems
Universitätsplatz 2, 39106 Magdeburg, Germany
{zug, schulze, dietrich, kaiser}@ivs.cs.uni-magdeburg.de

Abstract

Developing complex sensor/actuator systems, like robot applications, is challenged by a multitude of different hardware platforms, networks, programming languages, data formats, etc. In this paper, we present our architecture that copes with this heterogeneity and allows for a flexible composition of smart sensors and actuators. The main focus lies on a two layered approach combining the communication middleware FAMOUSO and the programming abstraction MOSAIC. FAMOUSO enables the information exchange between networked systems, hides the high degree of heterogeneity on hardware and network level, and is usable from different programming environments. MOSAIC uses FAMOUSO and provides a generic access to the exchanged information. Furthermore, it offers a way to abstract from different sensor and actuator hardware and provides a framework for application development with predefined components, enabling comprehensive fault detection. The paper concludes with a case study that shows how the middleware and programming abstractions are used to build a distributed modular system for a robot manipulator.

1 Introduction

Networked components like smart sensors, actuators and special computational devices emerge as the hardware building blocks for large scale automation systems. They offer the potential to build such control systems in a modular and incremental way or even allow dynamic extension of the system when mobile units connect to fixed environmental sensors spontaneously. Unfortunately, many problems have to be solved on the way to a reliable, seamlessly integrated system that is easy to program, easy to extend and easy to maintain.

One problem originates in the high degree of heterogeneity on all system levels that adversely affects integration. The diversity appears not only on the lower system levels, i.e. different controllers, field-busses, protocols and operating systems, but also on the level of programming languages and domain specific tools to program, monitor and maintain these systems. Changes of the system configuration (e.g. replacement of a component) or low level changes in one component (e.g., a network address) may involve a chain of dependent changes in other components which may require specific tools and tedious reprogramming of flash memory. This kind of heterogeneity problem can be tackled by an adequate communication model and middleware that hides the network addressing schemes and offers a content- or subject-based routing scheme. We assume that the individual smart components have substantial resource constraints. Many smart components in the automotive and also automation industry are based on 8 or 16-Bit CPUs and the communication networks have limited bandwidth. Thus, the middleware must be able to work on these systems.

A second problem is the diversity of sensors with many different modalities and data formats. This can be solved by a standard that must cover a wide range of possible sensors from simple temperature sensors to cameras. The usual way to achieve this is defining a standard description of a sensor in the form of electronic data sheets, like in [11, 17], but modifications and extensions are needed, as we describe later. Moreover, defining a structured programming model and a general interface simplifies the implementation of sensor components, greatly.

The third problem is related to the structure of a sensor and addresses reliability issues. Assuming a decentralized system of sensors and actuators connected by wired and wireless networks, an indication of correctness of information is important. Firstly, the compatibility of information like physical units and the temporal settings during the system integration and configuration phases must ensured. We tackle this problem with expressive descriptions of components and endowing the compiler and integration tools with the respective checking capabilities [28]. The second aspect is related to the assessment of the quality of individual sensor measurements during run-time. This requires self-checking capabilities of the components. The proposed sensor structure exploits

model-based analytic redundancy and comprises building blocks for the detection of outliers and other typical sensor faults [9].

This paper gives an overview of our architecture, describes the middleware and the sensor structure, and presents a case study of a robot application that includes many sensors, actuators, networks, and monitoring and visualization components. The contribution of the paper is to show the problems when integrating such a system and how the developed concepts and mechanisms ease programming of such systems and system integration. In the next section we survey related work in the relevant areas. Then we sketch the main properties of the middleware FAMOUSO and briefly present the sensor model MOSAIC. Section 4 introduces the case study and highlights the benefits of the proposed architecture. A summary and outlook on ongoing work concludes the paper.

2 Related Work

The related work section examines two main fields that are challenging when integrating networks of smart components. Firstly, we need appropriate middleware that enables communication across different kinds of networks, covering the field-bus level and energy-efficient wireless protocols. This middleware must run on different hardware platforms down to small micro-controllers. Secondly, we require a programming abstraction that offers a uniform access to the exchanged information and a predefined processing structure, utilizing this common interface. In the following section we examine existing approaches with respect to our general requirements.

2.1 Communication Approaches

Internet Scale Middleware SIENA [6], HERMES [25] or READY [12] are publish/subscribe systems, based on a static broker overlay network with reliable TCP/IP connections. Thus, these systems support information exchange for distributed application, but they demand a standard operating system like Linux or Windows, requiring powerful devices.

The Network Data Distribution Service NDDS [24] or the ACE ORB (TAO [13]) are publish/subscribe systems, which support many platforms and also offer soft real-time features. However, the communication is based on UDP/IP, having also requiring powerful devices.

In general, traditional middleware systems such as DCOM (Distributed Component Object Model [10]), JMS (Java Messaging Service [31]) or CORBA-DDS (Common Object Request Broker Architecture-Data Distribution Service [23]) are normally heavyweight in terms of memory and computation and therefore not suitable for the use on resource-constrained embedded systems.

Robotic Middleware In the robotic field, middleware systems have been developed that try to ease the development by composing the robot's control system at the software level with components or services. Systems like OROCOS (Open RObot Control Software `www.orocos.org`), OCERA (Open Components for Embedded Real-time Applications `www.ocera.org`) or Microsoft RoboticStudio [21] fall into this category. OROCOS and OCERA are component systems, using a CORBA-DDS implementation for the information exchange of distributed applications. However, integrating low-level components connected via an industrial field-bus the communication model is different, and applications have to know where information are located and realize the access to such a bus by itself to get the required information, which makes the development uncomfortable.

In contrast to the component approach, the Microsoft RoboticStudio uses a service-oriented architecture, which supports the simulation of robot behavior in a virtual environment, based on realistic physical models that reproduce the mechanical behavior and offers the simulation of most common sensors and actuators. Furthermore, it allows applying the same control schemes to real hardware. However, the disadvantage is the high resource requirements, because it uses TCP/IP for the communication and needs a whole .NET framework to be worked.

Sensor Network Middleware For Wireless Sensor Networks (WSNs) only few middleware systems are available, like MIRES [30] supporting publish/subscribe communication or TinyLIME [7] providing a tuple space. Using these middleware systems means to be tied to the component model of TinyOS [16], because both are developed on top of this. Furthermore, applications have to be programmed in NesC, the programming language of TinyOS, which means on the one hand the developer can not use its preferred tools and on the other hand it needs to learn a new language. However, the main drawback of using the mentioned systems is the lack of support of TinyOS for different platforms, because at the moment of this writing only three different CPUs are supported.

2.2 Programming Abstraction Approaches

Instrumented Logical Sensor Henderson et al. proposed hierarchically applicable fusion/filter units, called Logical Sensors or Instrumented Logical Sensors [14] and developed a complete toolchain with a sensor description, configuration, and code generation. Henderson's approach focuses on an adaptability of each Logical Sensor to a varying number of incoming data. A sensor selection mechanism manages the data acquisition for this purpose and tries to compensate missing individual sensor measurements or network inputs. The Logical Sensor integrates network interfaces only. Real transducers and their drivers are executed separately in a special gateway instance of a Logical Sensor. The characteristic output vector, defined for each Logical Sensor, does not consider several aspects of sensor applications like perception uncertainties, units of measurement, etc. necessary for a tailored processing or fusion.

Fusion Channels The fundamental abstraction of the architecture described by Agarwalla et al. in [26] is called a Fusion Channel (FC). A FC abstracts a set of inputs and encapsulates a programmer defined fusion function. The inputs are obtained from a distinct address space or from a remote host. The behavior of the fusion process is controllable by requests to the FC or triggered in case of new input data. Applications access the FC result in two ways: as a single value with a timestamps or the whole FC output buffer. Requests specify a minimum number of inputs and a timeout to get a result. Further, FC may be organized in hierarchical structures. The FC approach does not consider an abstract description of the exchanged information. The developer, who prepares a fusion application running inside a FC, has to have an explicit knowledge of the memory usage. An implementation of the FC concept in the Dfuse framework [19] requires a complex predefined infrastructure and uses Ethernet based protocols only.

Virtual Sensors The traditional Virtual Sensor merges several measurements into a joint estimation, quite often based on a physical model as presented in [2] . In contrast, Bose et al. [4] describe a programming abstraction for distributed applications and defines a number of subclasses for different purposes of hierarchical ordering. The first level, the Singleton Virtual Sensor (SVS), accepts only individual measurements and assigns sensor position, sensor ID, etc. A Basic Virtual Sensor (BVS) combines multiple SVSs of the same type and provides a better reliability. A Derived Virtual Sensor integrates different BVSs and provides abstract SQL queries to raw and joint data. In case of crashed Virtual Sensors the network structure is reconfigured automatically. The reconfiguration mechanism limits the Virtual Sensors to simple sensor assumption about sensor specifications i. e., equal measurement noise, equal range, etc. for all sensors. Additionally, Virtual Sensor applications are limited to a hierarchical depth of three nodes, according to the definition of the three subclasses.

Smart Transducer Interface (STI) The OMG Smart Transducer Interface Specification [22] provides an access to sensor measurements via the CORBA real-time service interface. The standardization of the different interfaces is mapped on an interface file system (IFS) typically in the memory of each Smart Transducer. For an interpretation of the outputs an additional metadata for each IFS are stored on a central node with higher performance. The integration of CORBA limits the implementation of the STI approach to powerful CPUs. The authors of [11] enhanced the STI concept and developed an XML description of the functionality for simple fusion tasks, also offering a TTP/A network support.

IEEE 1451 The Smart Transducer Interface represents a family of standards for connecting smart devices [17]. IEEE 1451.2 defines an electronic data sheet and a digital sensor interface to access sensor measurements, set actuators, control maintenance functions, or to obtain the data sheet of the sensor/actuator system. Hence, the standard establishes the communication between a Network Capable Application Processor (NCAP) and an actual sensor node called Smart Transducer Interface Modules (STIM). The combination enables a flexible access to different networks via special NCAP gateways. The standards 1451.3 to 1451.5 enhance the interaction between STIMs and NCAPs to various protocols and interfaces. The description of the sensors, stored at each node, contains a detailed specification of the sensor's vendor, firmware, and physics in a compressed Transducer Electronic Data Sheets (TEDs). TEDs do not support complex sensor information like characteristic curves for linearization or probability functions of sensor's noise. The use in realistic sensor scenarios without this information is quite limited. Compared to other mentioned approaches IEEE 1451 represents an abstract description of the sensors interfaces only, which have to be mapped to a predefined number of sensor types. In [18] the authors use some concepts of the standard to develop a more common programming abstraction with middleware interactions but restricted to simple sensor models.

2.3 Conclusion

The described middleware implementations are usable in their specific contexts. They support special types of networks and protocols and are either limited in scope and functionality when supporting small devices or they require very powerful nodes. None of the enumerated abstractions, standards, etc. fully meets our expectations for a unified programming abstraction for sensors and actuators that considers varying configurations with a common interface access and are executable on performance limited devices. Moreover, the related schemes do not provide integrated fault-tolerance mechanisms.

3 Architecture

We propose an architecture, providing the flexibility to integrate and segregate components during run-time, dynamically. For this purpose we combine a communication middleware and a programming abstraction for distributed applications. The middleware organizes the transmission of all necessary information, while the programming abstraction is responsible for an adequate filtering, selection, fusion, and validation.

3.1 Communication – FAMOUSO

Our middleware FAMOUSO (Family of Adaptive Middleware for autonomOUs Sentient Objects [15, 27, 29]) provides an event-based communication over different network types, according to the publisher/subscriber paradigm. In contrast to an address-based communication, an anonymous content-based communication is used,

where events are exchanged between communication endpoints. Publishers as well as subscribers are roles that applications have during the communication. Related to its characteristic as publisher, subscriber, or both, applications specify the kind of events they produce or consume. On that simple scheme, FAMOUSO provides spontaneous and dynamic many-to-many communication without any assumptions about synchrony of events. The communication is always asynchronous, avoiding control flow dependencies and enabling the autonomy of communication participants.

The exchanged information – FAMOUSO events – consists of three parts: a subject, optional attributes, and content. Optional attributes could be context attributes, which deliver additional information about the origin of the event like location or timestamp. Subjects are defined by the applications, and they build a global address space, spanning across all networks. This feature is exploited by gateways that enable and manage communication between different networks. The uniqueness of subjects is used firstly to filter the information flow on network borders if a subject is only required within a specific subnet, and secondly to perform forwarding if the subject is subscribed outside.

From the perspective of most applications, the definition of events and its use should be sufficient. However, in the embedded field, applications have often quality of service (QoS) demands regarding real-time or dependability issues. These demands have repercussions to the underlying support system, because the system has to ensure and enforce given guarantees. To tackle that challenge, FAMOUSO has the notion of event channel, which is used firstly as an abstraction for event disseminations, secondly for the specification of dissemination requirements like deadline, jitter, omission degree, etc., and thirdly for reserving the needed local as well as network resources to enforce the given guarantees if possible. Further, FAMOUSO supports with its Multi-Level Composability Check Architecture (MLCCA [28]) an integrated component that detects a misconfiguration or an unrealizable application demand as early as possible. If an event channel is correctly setup, events can be transferred through this event channel to its destinations, if the subject of the event corresponds to that from the channel.

FAMOUSO is realized as a layered architecture. The number of layers depends on the selected middleware configuration, but FAMOUSO has usually three layers, as depicted schematically in Figure 1. The number of layers grows in case of complex configurations for e. g., gateways due to the need to integrate the respective functionality into the infrastructure. On each layer certain functionality is implemented, and the abstraction level increases from the concrete network layer up to the event layer, which provides the publisher/subscriber interface. Applications get and use, independently of the actual configuration of the middleware, only this interface.

The different layers offer special functionalities. On

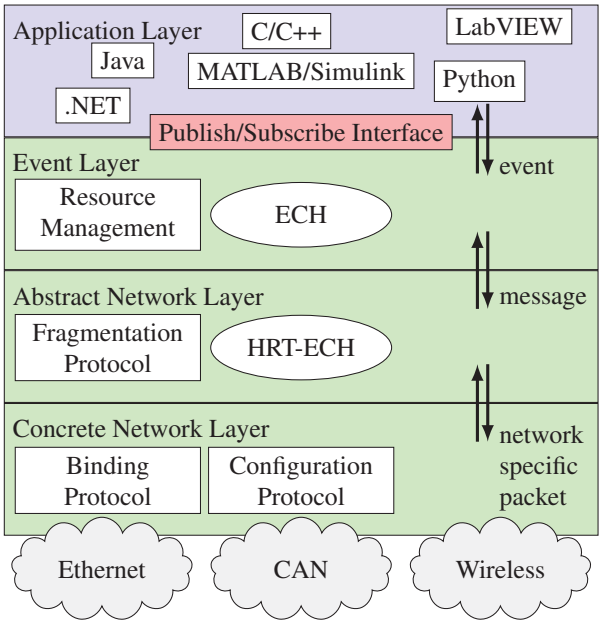


Figure 1. Schematic view on the layered architecture of FAMOUSO

the Event Layer level, the event channels are managed by the Event Channel Handler (ECH). The ECH takes care about necessary resources, and observes the guaranteed QoS parameters at run-time. For example, if an application specifies a period of $50ms$ for incoming events on an event channel, and within the last $50ms$ no event arises, leading to a specification violation, the middleware calls an error-handler callback for this event channel. The callback mechanism of the middleware permits indicating specification violations, enabling applications to be aware of violations, and thus reacting accordingly for example with a fail-safe state or with adaptation.

The following layer, which is the Abstract Network Layer, is responsible for functionality that can be realized independently of specific network characteristics and thus made available for several networks. One example is the adaptive fragmentation protocol, which enables transferring large events over networks that are not able to send such events as a whole. Furthermore, the layer provides the real-time communication mechanisms for events that are termed as messages here.

The Concrete Network Layer (CNL), the lowest layer, encapsulates all this functionality that is absolutely specific for the respective network, because networks differ in a lot of ways (e. g., address scheme and message format) and a generalization of all functionality is not possible. Firstly, the CNL contains the binding protocol, which is responsible for binding the subject to a specific network representation, because this totally differs between CAN or Ethernet, and secondly, the configuration protocol configures the node to give it a unique network name. To ensure the compliance of the network protocol character-

istics, the protocols exist in specialized versions for each supported network and they are part of the respective network layer realization, which supports the upper layers with functionality for the communication.

FAMOUSO supports a broad variety of different hardware platforms ranging from low-end 8-Bit microcontrollers up to high-end 64-Bit server systems and enables interaction over different communication media like the CANfield-bus, Wireless Sensor Networks (WSN) like IEEE 802.15.4,Wireless Mesh Networks like AWDS [3], and Ethernet like UDP broad- and multicast (Figure 1). FAMOUSO can be used from different programming languages (C/C++, Python, Java, .NET) as well as from engineering tools (LabVIEW, MATLAB/Simulink) simultaneously. Thus, the middleware enables the developers to individually choose their preferred combination of tools and languages. Objectives of FAMOUSO are configurability, adaptability, portability, and efficient resource usage to allow also the deployment on small resource-constrained embedded devices.

3.2 MOSAIC

A programming abstraction for distributed applications should offer three core elements: firstly, it needs a generic access to the exchanged information and an abstract interface to sensors/actuators. Secondly, it should provide a modular structure for applications, because such a predefined modular structure allows a flexible replaceability of inner components and enables a comprehensive fault detection and classification as the third important property.

Based on these requirements we developed our fraMework for fault-tOlerant Sensor dAta processIng in dynamiC environments (MOSAIC) that defines an appropriate programming abstraction – the "Smart Abstract Entity". This approach extends and combines the concept of preprocessed and self describing measurements done by Smart Sensors in combination with the Abstract Sensor concept of Marzullo [20].

Due to the flexible composability and in relation to the different purposes of Smart Abstract Entities, we distinguish between three variants that can be combined in distributed applications: the first one, the Smart Abstract Sensor, visible in Figure 2, uses one or more real transducers to perceive the environment and communicates the measurements after filtering and validation tasks. In contrast to this variant, the second one is the Smart Abstract Actuator, which controls a mechatronic device based on the information obtained form the communication interface. The Smart Abstract Fusion Node does neither include sensors nor actuators and uses the communication interface only. Such entities are used for measurement fusion and processing, simulated sensors, etc.

In Figure 2 we depict the basic building blocks of a Smart Abstract Sensor. Except for the actuator output interface, Smart Abstract Actuators are structured very similarly to Smart Abstract Sensors. From the application point of view we have to cope with two interface

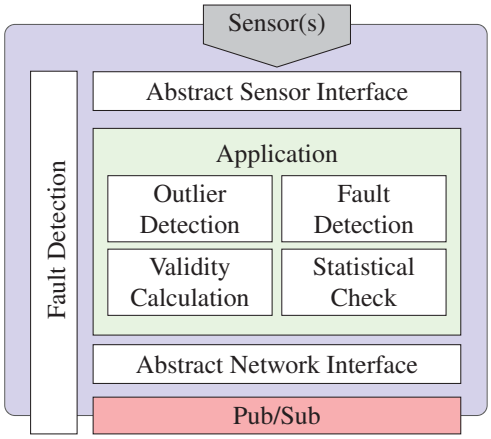


Figure 2. Smart Abstract Sensor Structure

types in general. The first one organizes the access to data contained in FAMOUSO events, arranges and buffers the data according to application requirements and collects fault notifications from FAMOUSO (for instance due to the absence of periodic events). The knowledge about the data format of events is located within an XML electronic data sheet for each FAMOUSO event channel. This XML-document contains all information that are necessary to interpret an incoming event correctly (data types, attributes, units, uncertainties, etc.) and to supervise the channel (deadlines, periods, and omissions).

The second interface of a Smart Abstract Entity depicted in the upper part of Figure 2 is responsible for the communication with sensor and actuator hardware, and furthermore linearizes and transforms sensor measurements (e. g., voltage from ADC into degree Celsius for temperature sensors). We used the same approach again and developed an XML description that contains sensor and hardware specific properties and embed them into a toolchain for Abstract Entities based on MATLAB/Simulink [5]. In combination with the FAMOUSO event channel descriptions, the developer obtains an application framework that includes both abstract interfaces. The specific configuration of sensor interfaces is encapsulated in the model generation process. Thus, the engineer focuses on the application development, uses domain specific development tools, and does not need not to cope with communication mechanisms nor hardware implementation.

A predefined application structure is important to enable fault detection mechanisms in all components, because it is a cross cutting concern. We enhanced the existing idea of Smart Sensors with the fault-tolerance aspect. Each component of the application framework calculates a fault probability that is merged in a fault detection component and assigned to each generated event at the end. Therefore, we analyzed the faults of distributed sensor and actuator applications and discussed possible detection methods and derived a modular structure for Smart Abstract Sensors in [9]. Besides, the flexible communi-

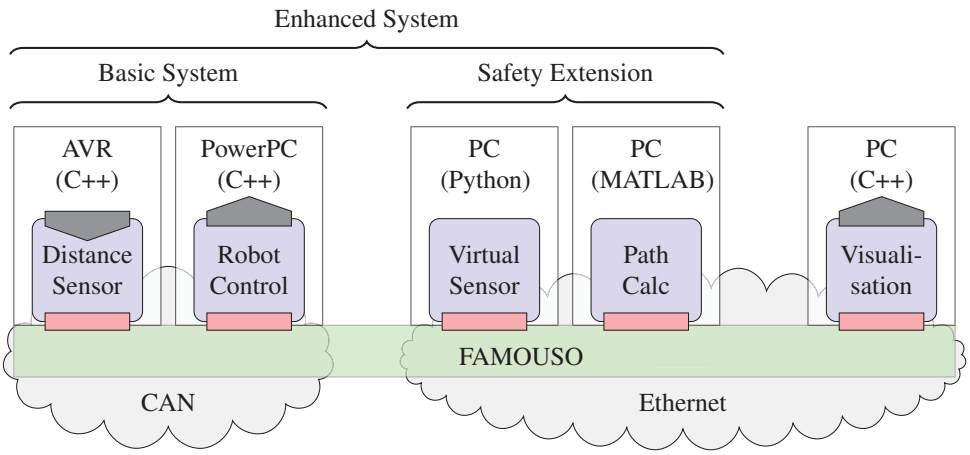


Figure 3. Scenario structure combining FAMOUSO and MOSAIC

cation infrastructure enables new fault-tolerance methods for Smart Abstract Sensors. Each Smart Abstract Sensor is subscribed to the fusion result that is potentially available. This feedback offers redundant information and enables an efficient validation of the current measurement.

As shown in this section, we continued the layered architecture of FAMOUSO in our programming abstraction, reduced the integration effort for sensors and communication interfaces, and presented a comprehensive programming abstraction for distributed applications.

4 Scenario

Using our layered FAMOUSO/MOSAIC architecture throughout the system allows to setup distributed applications easily. Furthermore, applications may be enhanced by adding components dynamically and without any need to change application code of other running system components. In industrial environments for example, a high level plant asset management system subscribes to information of different production lines for monitoring or even control purposes. On a lower system level, additional sensors can be integrated to extend the sensor-based perception area of the environment.

We present a scenario that serves as a typical example to emphasize the benefits of our approach by showing aspects like modularity and dynamic composability. As a physical actuator, we use a robot manipulator that is equipped with a limited number of real distance sensors, observing the near environment. The manipulator follows a pre-calculated trajectory and stops in case of a detected obstacle. Next, we integrate a safety extension, which allows for adding e. g. virtual walls dynamically, in order to restrict the manipulator's working area for safety reasons. Furthermore, we use this example setup in the development phase of reliable and maintainable robot applications as well as when exploring extended human-robot interaction schemes.

Figure 3 presents the schematic structure of the scenario and illustrates the diversity of components, programming languages, and underlying communication networks. Due to the FAMOUSO middleware and MOSAIC all components can be easily integrated or segregated without much effort. Implementing the scenario without the support of FAMOUSO and MOSAIC is possible, however, it means implementing the low-level access to a CAN network "Basic System" and accessing a UDP network from three different programming languages. The data exchange between the CAN and UDP network is also in the responsibility of the developer, but using FAMOUSO gateways (not depicted in Figure 3) are an integral part of the infrastructure. In the same way an implementation may be done without fault-tolerance mechanisms and individual data formats instead of generalized definitions in electronic data sheets. However, the effort to establish a dynamic configurable and maintainable application increases significantly and error-proneness grows.

4.1 Basic System

The basic robot application consists of two elements that are connected via CAN. These are a PowerPC, in the role of a Smart Abstract Actuator, and a Smart Abstract Distance Sensor that is controlled by an AVR AT90CAN128, both programmed in C++. The AVR publishes events, periodically. Events contain a distance measurement, the related validity value, and respective sensor failure/error modes. The PowerPC is responsible for controlling the Katanta robot, a five degrees of freedom manipulator. This *Robot Control* publishes status data (e. g., angles of robot's axes, different modes, and present current) and subscribes to control commands for movement, speed, and emergency stop.

4.2 Safety System

The basic system may be enhanced by a safety system without changing anything at the base system. As shown in Figure 3, we integrate a safety system consisting of two additional PCs that are connected to the Ethernet. A

FAMOUSO gateway connects both networks and ensures that events are routed to the interested participants. One of the additional components, a *Virtual Sensor*, is implemented in Python, and it publishes distance values. These distance values are calculated from the robot's distance to some virtual walls. In this way it is possible to define virtual safety areas the robot is not allowed to leave. The only data that is required by the *Virtual Sensor* here are the angles of robot's axes.

The second component, the *Path Calculation* is realized on a separate PC and is implemented in MATLAB. The manipulator's trajectory is calculated depending on all available sensor distance data – real or virtual – as well as to the values and states of the robot's axes. A path is composed by multiple stages that are published sequentially in form of axes values. A new stage is transmitted when the robot reaches a target position. If the robot is unable to reach its target position due to a detected obstacle, another path will be calculated and published.

4.3 Visualization

As a third part of our scenario, we add a *Visualization*, again without any adaptation of other components of the original system at all. This application is for supporting factory workers, developers, and maintenance workers by using the technique of Augmented Reality, which presents an enhanced visual representation of a work space. The realization of the *Visualization* uses ARToolKit [1] and OpenGL to overlay real world camera images with additional information. A detailed description of the benefits of using Augmented Reality to support different kinds of users in industrial application is beyond the scope of this paper and can be found in [8].

The *Visualization* component subscribes to different information (e. g., axes positions, sensor measurements, stages of path calculation) according to user requirements and presents the information in an appropriate manner. For visualizing the information perspectively correct, Augmented Reality needs to match the real and virtual images. The ARToolKit uses marker-based object identification, and information is relatively drawn to the detected markers.

The screenshot taken from an ordinary monitor in Figure 4 shows such an Augmented Reality in operation. It depicts for example a view for a developer, which differs completely from the view that a factory worker would need to interact with the robot. For these scenarios and applications head-mounted displays will be more appropriate than monitors, however the application can be adapted very easily to these more advanced devices.

A factory worker requires a proper graphical representation of safety areas (subscription to *Virtual Sensor*), robot's future trajectories (subscription to *Path Calculation*), or a graphical representation of robot's states (subscription to *Robot Control*).

The visualization for developers like in our scenario can be more complex to enable playing around with the experimental system to acquire experience. This includes for example sensor data (subscription to *Virtual Sensor* and *Distance Sensor*), robot's states and position of axes (subscription to *Robot Control*). While in this case it is also appropriate to use colors, color transitions or transparency to visualize additional information like uncertainties, ranges, or the age of sensor data. Additional information like diagrams or robot's contour can be placed onto display as well.

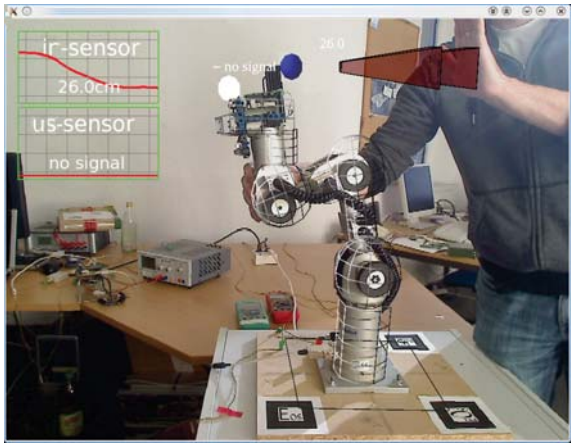


Figure 4. Visualization for engineers, presenting the current sensor output

5 Conclusions and Outlook

MOSAIC and FAMOUSO ease the development of extensible, distributed and modular applications across different platforms and networks. We demonstrated the benefits in a robotic scenario. The communication middleware enables a dynamic composition during development process, configuration and even on run-time for system upgrades. FAMOUSO supports domain specific languages and flexible and seamless integration of distributed hardware and software modules. This opens an easy way for Hardware-/Software-in-the-Loop test scenarios. To address the specific problems of correct sensor information in a decentralized and dynamic scenario, MOSAIC establishes a generic smart component structure and interface.

Next steps will include improvement of the code generation process for constrained platforms, integration of multilevel compliance checks based on our XML descriptions and extended detection mechanisms for faults, typical for smart distributed sensors.

Acknowledgement

This work is partly founded by the Ministry of Education and Science (BMBF) within the project "Virtual and Augmented Reality for Highly Safety and Reliable Embedded Systems" (ViERforES - no. 01IM08003C).

References

[1] Artoolkit. http://www.hitl.washington.edu/artoolkit/, 2007. [(online), as at: 25.02. 2010].

[2] P. Albertos and G. Goodwin. Virtual sensors for control applications. *Annual Reviews in Control*, 26(1):101–112, 2002.

[3] AWDS project. http://awds.berlios.de, 2009.

[4] R. Bose, A. Helal, V. Sivakumar, and S. Lim. Virtual sensors for service oriented intelligent environments. In *Proceedings of the third conference on IASTED International Conference: Advances in Computer Science and Technology*, pages 165–170, Phuket, Thailand, 2007. ACTA Press.

[5] T. Brade, M. Schulze, S. Zug, and J. Kaiser. Model-Driven development of embedded systems. In *12th Brazilian Workshop on Real-Time and Embedded Systems (WTR)*, Gramado, Brazil, 24 May 2010. Brazilian Computer Society.

[6] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Trans. Comput. Syst.*, 19(3):332–383, 2001.

[7] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. Murphy, and G. Picco. TinyLIME: Bridging Mobile and Sensor Networks through Middleware. In *Third IEEE International Conference on Pervasive Computing and Communications*, pages 61–72, Kauai Island, HI, USA, March 2005.

[8] A. Dietrich, M. Schulze, S. Zug, and J. Kaiser. Visualization of Robot's Awareness and Perception. In *First International Workshop on Digital Engineering (IWDE)*, Magdeburg, Germany, 14 June 2010.

[9] A. Dietrich, S. Zug, and J. Kaiser. Detecting external measurement disturbances based on statistical analysis for smart sensors. In *Procedings of the IEEE International Symposium on Industrial Electronics (ISIE)*, 2010.

[10] G. Eddon and H. Eddon. *Inside Distributed COM*. Microsoft Press, 1998. ISBN 1-57231-849-x.

[11] W. Elmenreich, S. Pitzek, and M. Schlager. Modeling Distributed Embedded Applications on an Interface File System. In *Proceedings of the Seventh IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'04)*, pages 175–182, Vienna, Austria, 2004.

[12] R. Gruber, B. Krishnamurthy, and E. Panagos. The Architecture of the READY Event Notification Service. In *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems Middleware Workshop*, pages 01–08, 1999.

[13] T. H. Harrison, D. L. Levine, and D. C. Schmidt. The Design and Performance of a Real-time CORBA Event Service. *ACM SIGPLAN Notices*, 32(10):184–200, October 1997.

[14] T. C. Henderson and M. Dekhil. Instrumented Sensor System Architecture. *The International Journal of Robotics Research*, 17(4):402–417, 1998.

[15] A. Herms, M. Schulze, J. Kaiser, and E. Nett. Exploiting Publish/Subscribe Communication in Wireless Mesh Networks for Industrial Scenarios. In *Proceedings of Emerging Technologies in Factory Automation (ETFA '08)*, pages 648–655, Hamburg, Germany, September 2008.

[16] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Networked Sensors. *ACM SIGPLAN Notices*, 35(11):93–104, November 2000.

[17] IEEE Standards Association. *IEEE Standard for a Smart Transducer Interface for Sensors and Actuators (IEEE 1451.2)*, 1997.

[18] J. Kaiser and H. Piontek. CODES: Supporting the development process in a publish/subscribe system. In *Proceedings of the fourth Workshop on Intelligent Solutions in Embedded Systems WISES 06*, pages 1–12, Vienna, 30. June 2006. ISBN: 3-902463-06-6.

[19] R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, and U. Ramachandran. Dfuse: a framework for distributed data fusion. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 114–125, Los Angeles, California, USA, 2003. ACM.

[20] K. Marzullo. Tolerating Failures of Continuous-Valued Sensors. *ACM Transactions on Computer Systems (TOCS)*, 8(4):284–304, November 1990.

[21] Microsoft Corporation . Microsoft robotics studio. online, http://msdn.microsoft.com/en-gb/library/bb881626.aspx.

[22] Object Managment Group (OMG). *Smart Transducer INterface Specification*, 2003.

[23] OMG. *Data Distribution Service for Real-time Systems Version 1.2*. Object Managment Group, 1. January 2007.

[24] G. Pardo-Castellote and S. A. Schneider. The Network Data Delivery Service: Real-Time Data Connectivity for Distributed Control Applications. In *Proceedings of the ICRA*, volume 4, pages 2870–2876, San Diego, CA, USA, May 1994. IEEE Computer Society Press.

[25] P. R. Pietzuch and J. Bacon. Hermes: A Distributed Event-Based Middleware Architecture. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 611–618, Washington, DC, USA, 2002. IEEE Computer Society.

[26] U. Ramachandran, R. Kumar, M. Wolenetz, B. Cooper, B. Agarwalla, J. Shin, P. Hutto, and A. Paul. Dynamic Data Fusion for Future Sensor Networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(3):404–443, 2006.

[27] M. Schulze. FAMOUSO – Eine adaptierbare Publish/ Subscribe Middleware für ressourcenbeschränkte Systeme. *Electronic Communications of the EASST (ISSN: 1863-2122)*, 17, 2009.

[28] M. Schulze and G. Lukas. MLCCA – Multi-Level Composability Check Architecture for Dependable Communication over Heterogeneous Networks. In *Procedings of 14th International Conference on Emerging Technologies and Factory Automation*, Mallorca, Spain, 22-26 Septmeber 2009. IEEE.

[29] M. Schulze and S. Zug. Exploiting the FAMOUSO Middleware in Multi-Robot Application Development with Matlab/Simulink. In *Proceedings of the ACM/IFIP/USENIX Middleware '08 Conference Companion*, pages 74–77, Leuven, Belgium, 1-5 December 2008. ACM.

[30] E. Souto, a. Germano Guimar G. Vasconcelos, M. Vieira, N. Rosa, and C. Ferraz. A Message-Oriented Middleware for Sensor Networks. In *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, volume 77 of *ACM International Conference Proceeding Series*, pages 127–134, Toronto, Ontario, Canada, 2004. ACM.

[31] Sun Microsystems, Inc. Java Message Service (JMS) Specification 1.0.2, 1999.

A.2.3 A fault-aware sensor architecture for cooperative mobile applications

"A fault-aware sensor architecture for cooperative mobile applications". Joerg Kaiser, Sebastian Zug, May 2012, 26th IEEE International Parallel and Distributed Processing Symposium, Shanghai, China.

This page is intentionally left blank.

A fault-aware sensor architecture for cooperative mobile applications

Jörg Kaiser, Sebastian Zug
Universität Magdeburg
Department for Distributed Systems
Universitätsplatz 2, 39106 Magdeburg, Germany
{kaiser,zug}@ivs.cs.uni-magdeburg.de

***Abstract*—One of the striking characteristics of mobile embedded systems is the interaction with the physical world. Prerequisite for this interaction is the reliable perception of the environment by sensors. Exploiting remote sensors of an instrumented environment and other mobile systems will extend the range and modalities of sensing and, in principle, will contribute to a better environment perception. However, such as distributed sensor system also puts new substantial challenges on the assessment and dynamic use of sensor data. This paper focus on dependability issues and presents a fault abstraction and fault-handling concept, that encapsulates individual sensor faults and encourages multi-level fault detection. Based on an analysis of sensor faults, our approach generates a validation for each sensor measurement. This allows assessing remote sensor data quality in a uniform way and provides the basis for a distributed mobile application in which new sensor sources can be discovered and exploited dynamically.**

***Index Terms*—Fault detection, Fault abstraction, Fault model, Distributed application, Smart sensors,**

I. Introduction

It is a commonplace today that most computers work as an embedded system. The list of emerging applications is long and well-known examples comprise navigation systems, intelligent cars, electrical machines in the household or, at the industrial end, smart payloads and handling systems in logistics or autonomous transportation and processing in industrial automation. There is a key property that all these systems have in common. They all perceive aspects of their physical environment and react on it proactively and autonomously. Networking capabilities of the devices open the door to share a rich set of environmental information with other systems, extending the range of perception and the modalities of information sources. In industrial automation, autonomous vehicles may exploit navigational information or remote obstacle information from sensors of roadside instrumentation or from other vehicles and adjust trajectory planning accordingly. In logistics, payload and transportation system may cooperate to negotiate the handling and the destination. Systems of ambient intelligence or ubiquitous computing, seen as a network of things, include co-operating cars, things in the household and also mobile robots for assisted living or in search and rescue missions. The crucial point of all these applications is the need for reliable and trustworthy environmental information that is sampled by a large number of sensors, some of them may be local in the system, some of them may be networked sensors supplying remote data.

Because these systems interact with the physical environment and particularly with humans there is the danger of damage and injury. Therefore the environment perception requires an increased effort to achieve a high level of dependability, particularly under the aspect of functional safety. While in conventional systems, the number of sensors is fixed, the sampling periods are thoroughly adjusted according to the time-value relation, the communication is scheduled deterministically, and consequently the quality of sensing information and the jitter of arrival is known, all these parameters are subject of substantial uncertainty in a scenario described above. This puts substantial new challenges to the field of control.

It points out that firstly distribution and secondly mobility adds new levels of variability. Distribution leads to the situation that readings from a fixed set of sensors encapsulated in event messages may not be available when a control decision has to be taken. Thus the control algorithm has to deal with a varying number of sensors readings and the age of sensor information that is affected by latency characteristics of general-purpose wireless connections. To put that further, mobile systems may use information of sensors that are discovered dynamically. Thus the set, modality and quality of sensors will be subjects of change and the quality of information may be uncertain.

Using spontaneously distributed sensor information in a control loop requires a substantial advance over conventional techniques. Firstly, it would be desirable to obtain an estimation of sensor data quality together with the actual measurement data. This refers to a self-assessment of a sensor and would allow an application selecting the most appropriate sensors form the available sources[1]. Next, there must be a notion of age of sensor data and also a model that relates this error in the temporal domain to the error in the value domain. This is particularly important because of varying network

[1]Clearly, self-assessment always bears the danger of being faulty if the computational components of the networked sensor are affected by a fault. However in this paper we deal with failures of the sensing component in the first line and therefore do not address for the moment, common mode failures (the analogue and the digital parts are affected simultaneously) and faults in the digital components.

latencies in a distributed wireless system. Finally, the multiple sensor data obtained from the environment have to be prepared for comparison and/or fusion that requires scaling, adjustment of physical units etc.

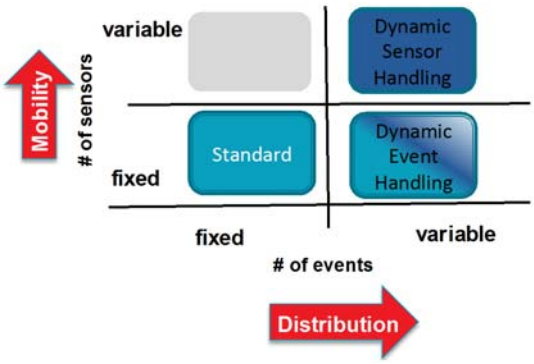


Fig. 1. Impact of distribution and mobility on control

In this paper we present a sensor model that is suited for the mobile scenarios above. Particularly, we introduce the notion of a smart sensor that encapsulates the complex failure modes of a physical sensor and provides a uniform interface that can be exploited by the subsequent control algorithms. This interface offers a well-defined fault semantics and a standard access to sensor data. Using this interface the designer of the distributed control does not have to care about the low-level sensor specific issues and the design of the control algorithm is decoupled from specific sensor parameters. This is a prerequisite for discovering and using sensor information dynamically when a mobile entity is moving around. The paper is organized as follows. The subsequent section introduces related work and further motivates the architecture of MOSAIC that is presented in Sec. IV. Here we focus on the fault tolerance mechanisms and illustrate the benefits of the approach for distributed sensing applications in Sec. V.

II. Related Work

So far, two fairly isolated communities are involved in the issue of dependable sensor perception. The fault-tolerance community in the first line deals with failures of the computing and the communication system, i.e. the digital components. The respective redundancy techniques rely on replication, coding and repetition, i.e. space, information and time redundancy, respectively. The main point is that the information, once in the system, is considered to be correct and should be maintained immutable or transformed correctly in the presence of faults. One of the well-studied problems is what was called *interactive consistency* (also called *source congruence* or *Byzantine agreement*), meaning that all non-faulty processes are guaranteed to have identical sets of sensor readings [1], [2]. Thus, the sensors were working correctly but the processes may fail. The control engineering community on the other side assumes that all system components of the controller are working correctly, but that the faults affect the controlled object (usually called the plant). This includes sensor and actuator faults. A certain degree of uncertainty in sensor readings, actuator insufficiencies and environmental disturbances are handled by robust design of the control system. However, if these impairments go beyond a certain threshold, they have to be recognized as faults and the control laws have to be changed or the faulty components have to be detected and removed from the system. Respective techniques like Fault Detection and Isolation (FDI) rely on knowledge about the overall system captured in a model that is checked against the actual system behaviour [3], [4]. Thus, this model is the prevalent form of redundancy in these systems. Some authors therefore distinguish between physical redundancy which subsumes e.g. time, space and code redundancy and analytic redundancy which is based on a model of the system behaviour that is checked against the actual behaviour. It should be noted that these forms of redundancy originate from distinct information and fault models. The approach handles failures from a fixed and known set of sensors. The control algorithm is adapted in an anticipated and pre-calculated way in response of a failure. Thus, the faults and the respective tolerance measures are tightly related and measures are precisely adjusted to the anticipated faults.

In control systems as mentioned above, all these individual failure modes are identified and implicitly considered in the fault-tolerant control algorithm. However, when striving for a distributed, collaborative sensor system in which the set of sensors is not known a priori and the number of sensors may vary dynamically, this approach is not feasible. What is needed, is a more unified behaviour of a sensor in case of a fault. The crucial point is to derive a failure semantics that describes the observable behaviour of a component in case of an internal failure [5] at the components interface. The failure semantics provides a higher-level fault model for a component that hides a more complex behaviour inside, thus it strongly supports the development of fault-tolerant applications. As an example, a hierarchy of fault abstractions in the temporal and in the value domain has been established in the distributed computing community [6], [5], [7], [8]. System designers developing distributed algorithms base their programs on the assumed fault model and only have to deal with the respective defined behaviour. It is by far easier to base e.g. distributed agreement on a fail silent crash semantics than on arbitrary timing and value faults. The problem is to make realistic assumptions about the component failure modes and the mechanisms of the underlying hardware/software to ensure the fault semantics at the interface of a component whatever failure happens inside. Enforcing a well-defined component behaviour in case of a fault can be seen as a fault-containment strategy that supports the separation of concerns.

The most important difference between distributed computing in general and sensor-driven computations when including a real-world interface is the nature of failures and the required redundancy. A sensor delivers continuous valued data and the sensor reading is inherently affected by a measurement error. This can best be compared to a clock that never exactly complies with the real time (whatever this is!). Therefore,

a clock value is considered correct if it is within specified bounds rather than complying with a single value. This has been early recognized by Marzullo [9], who introduced a fault-tolerant sensor scheme inspired by previous work on distributed clock synchronization. Marzullo distinguishes between a physical sensor (or concrete sensor his terminology) and an abstract sensor that includes a physical sensor and represents a programming model with a more convenient behaviour compared to the raw sensor output, e.g. it delivers an application related unit, it can be accessed whenever needed, it may deliver a continuum of values, etc. The fault-tolerance mechanism is based on replicating the abstract sensor and performing fault-tolerant averaging. Although our approach was inspired by Marzullo's work, we identified a number of differences and necessary extensions that reflect the many uncertainties of a distributed multi-modal sensor system that requires a higher level of adaptivity and failure awareness than provided in previous concepts.

Fault handling for fixed, static systems identifies a number of fault abstractions that encapsulate a known set of fault types. For instance, the fail silence *crash fault* semantics maps all faults on a single state that indicates the current fault status of the node. The designer anticipates all possible fault modes and ensures a continuous supervision. In case of a detected fault, the monitor unit stops any output or communication. Precondition of a *fail-silent* system is the detection of the assumed fault types [10]. Clearly, the spectrum of fault types that are captured by the detection mechanism determine the quality of the *fail silence* assumption. On small size sensor nodes this is usually limited by the processor performance that does not provide the power for complex analytic detection schemes. Additionally, the binary decision of the *fail-silent approach* is not adequate for continuous valued data that rather needs a more fine grained classification of the sensor data. To realize a more adaptive processing some authors categorize individual faults in meta models. E.g. Sharma et al. [11] define *short* that indicates a time-limited discontinuous change, *noise* that contains a stochastic component and *constant* that represents a time-dependent bias. In [12], [13], [14] the authors do not classify the faults but derive a validity value for each measurement. The validity value represents the probability of a fault occurrence during the measurement phase. In contrast to the *fail-silent* approach this additional information shifts the decision about an integration or rejection of a measurement to a higher level, e.g. provided by a more powerful node that executes a fusion algorithm. The authors of [15] describe a validity value, that distinguishes 16 levels. Kaiser and Piontek [12] introduce a continuous scale to define the validity of a measurement.

In our work we strive for providing a uniform interface to a sensing component that includes an estimation of the validity of data. This validity value represents the outcome of various test exploiting a combination of analytical models on various levels and a flexible way of comparing multiple related sensors.

III. Models for smart sensors

We assume a system of network-connected nodes that include sensor nodes, purely computational nodes, and nodes with actuators. Nodes with sensors and actuators may be deployed statically in the environment or they may be part of a mobile vehicle like an autonomous robot, a smart car or alike. Rather than raw signals, the sensor now communicates higher-level information, requiring some transformation process and protocol execution. For such components, the notion of a *smart sensor* was introduced. This term usually refers to components that integrate physical sensors, a computational component and a network interface. The "'smart"' indicates that these components are information sensors and actuators in the sense that they generate and consume meaningful application related information rather than raw data representing some physical unit directly measured by the raw sensor. The model of a smart sensor is of great importance when building distributed sensor systems because it frees the designer from dealing with low level details of the transformation process and provides a standardized message interface rather than a proprietary, complex and rather specific individual instrumentation interface to a sensor. There are many terms that describe a continuum of "'smartness"' reaching from just conditioning raw sensor data to components providing high quality application-level environment information. The spectrum reaches from early work of [16] who coined the term "'intelligent sensor"' and focus on integration on a single VLSI-chip, over smart sensors [17] that stress application relevance of sensor information, abstract sensors [9] that strive for an easy to use programming model and fault-tolerance, smart transducers [18] that highlight network and configuration issues, to cogent sensors that emphasize the fact that these sensors deliver high quality trustworthy information [19].This includes means to assess this information and handle sensor faults.

Although some of the presented schemes are close to our approach, we particularly strive for a programming model that supports fault-awareness and a hierarchical construction method for reliable sensors. The first part is achieved by a careful analysis and classification of sensor faults. From this we derive an assessment of sensor information quality. Secondly, we provide a uniform scheme to build very reliable sensors in a hierarchical way. We follow an approach where on the communication level, all components are modeled as sentient objects [20], [21]. Sentient objects constitute a general programming model and can be seen a general abstraction of an information sensor [22]. Sentient objects seamlessly communicate via a publish-subscribe middleware using typed communication objects called events. Sentient objects receive regular events (in contrast to raw, physical events), carrying sensor information via the communication system and generate regular events as output. Only at the system periphery to the physical environment, real sensors will sample physical events that are transformed into regular events. This transformation process however, is encapsulated in the smart sensor component. The notion of sentient objects supports building

a hierarchy of more sophisticated sensing components from multiple simple sensors. E.g. a distance sensor that exploits multiple modalities and fuses the regular events received from a smart infrared and an ultrasound distance sensor and a laser scanner. A sentient object that particularly represents a sensor will be subsequently called an abstract sensor.

IV. Requirements for a distributed dependable sensor system

To meet the level of generality and flexibility required by large scale distributed and dynamic sensor-actuator systems we have to consider a number of additional requirements that are not included in any of the schemes presented above. Firstly, we cannot assume a fixed set of sensors with known modalities because a mobile robot may discover and use sensor information spontaneously. Secondly, because environment information is disseminated via a wireless network, we have to cope with larger variations in delay, jitter and omissions compared to the dedicated links e.g. of the on-board sensors. Finally, we need to know the quality and trustworthiness of remote sensor information, because we will have to select the most appropriate one from the available set. Including bad sensor information will decrease the overall perception quality. Again, distributed clock synchronization is a good example because a pre-selection process usually sorts out outliers before a fault-tolerant average or midpoint is calculated [23]. However, the assessment of sensor quality in general is harder compared to the identification of a bad clock reading because of the continuous and linear model of (Newtonian) time.

When considering a sensor/actuator network with mobile and remote components, fault-tolerance has to cope with highly dynamic environments. We have to deal with the uncertainty about the available resources, which may change because of mobility or temporary unavailability. Secondly, sensor values coming from remote sensors over a network may be taken at different times or suffer from varying network latencies. Because sensor data represent time/value entities, the network latencies have to be bounded or have to be assessable at least. In an open wireless network, it is very hard to provide reliable bounds on latencies and jitter dynamically. Therefore we explore the assessment of these uncertainties than the enforcement of bounds. Thirdly, replicated sensors may suffer from the same external disturbance, i.e. they do not fail independently. Therefore multiple sensor modalities are desirable, however, this excludes simple averaging and requires more sophisticated fusion of sensor values. Fourth, replication of sensors is not feasible in general. For active sensors like infrared, ultrasound range sensors or laser scanners replication is not feasible, difficult and at least costly. Therefore, redundancy schemes beyond replication and comparison have to be considered. Finally, there is the uncertainty of the sensor information itself. How trustworthy and accurate is the information coming from a remote sensor? In this paper, we will present an architecture for such a system of smart sensors that combines multiple redundancy schemes and exploits the dynamic availability of sensors in proximity. We may summarize and condense the above discussion in the following requirements for a distributed sensor system. It must have the ability to cope:

1) with multiple modalities of sensors,
2) with different numbers of sensors,
3) with uncertainties coming from latencies and omissions of the network,
4) with uncertainties coming from bad sensor readings.

V. MOSAIC

MOSAIC constitutes an architecture for fault-aware abstract sensors[2] coping with heterogeneity of sensors, self assessment, distribution and mobility.

A. Architecture

A MOSAIC abstract sensors comprises components for sensor data acquisition, signal conditioning and processing, error detection and data validation and result transmission [24].The abstract sensor input layer that may receive sensor data from a physical sensors or via the network interface streams incoming data sets to a uniform processing chain. In case of dynamically discovered sensors, the respective event structures are extracted from an electronic data sheet which is an extension of the IEEE 1451 TEDS [25]. This enables the preparation of data sets compatible to the formats of the subsequent processing chain.

Each processing step generates an individual estimation of the validity. The results are combined by the fault management module providing the general measurement validity. This value is useful for assessing the measurements of multiple sensors of the same type. However, it is not sufficient to compare the quality of data from heterogeneous sensors. We are not able to interpret a validity value without a general reference scale that reflects characteristics of the tests as coverage and quality. E.g. a simple test for easy to detect outliers cannot generate the same level of confidence as a complex check based on a sophisticated model of the process. To handle this problem, we propose a fault effect validation that considers the type of fault and the effort of fault detection and maps the validity on a uniform confidence scale.

In a first step we analyze the relevant fault types and their characteristics in sensor networks. A detailed description can be found in [26]. The fault types like stuck-at-faults, different stochastic faults, time correlated faults, etc. are ordered in a fault classification scheme. This scheme provides a systematic investigation of a sensor, sensor node or processing chain to identify the possible faults classes. A data set is added to each fault class covering the relevant parameters. Beside specific information like noise characteristics or delay distributions it contains parameters needed for the fault effect analysis. The fault classification represents the first level fault abstraction. Individual fault specifications are mapped on a common scheme to cope with sensor and fault heterogeneity

[2]In fact MOSAIC was also developed to model abstract actuators. This aspect however is beyond the scope of this paper.

in distributed applications. As a result of this step the user has a list of the faults in a uniform representation.

The second step quantifies the effect of the node specific faults collected in the step before. Two parameters of a fault determine its relevance: the occurrence probability and the maximum fault amplitude. E.g. the fault category *constant noise* continuously affects a measurement and thus results in a high occurrence probability. The maximum fault amplitude depends on the noise characteristics but may be low in general. In contrast outliers generate large deviations, thus usually they exhibit a large amplitude but they have a rather low occurrence probability. Additionally, the correct weighting and processing of a measurement depends on the assigned detection methods. We map the capability of a method to realize a certain fault type by a detection probability.

For an overall validity value we have to consider three parameters – occurrence probability, maximum fault amplitude and detection probability – for n fault types. To combine these values in a single index we propose an adapted Failure Modes and Effects Analysis (FMEA) method. FMEA is a scheme for identification, validation and classification of faults in product development and operations management. FMEA methods are frequently used in industrial development processes for a system decomposition, an analysis of the potential faults and risk assessment.

FMEA constitutes a classification scheme that provides a systematic approach to compare systems. It is illustrated in its adapted form in Tab. I. The three parameters are mapped independently on an index value from 1 to 10. Following the FMEA notation A_n denotes the fault amplitude, O_n the fault occurrence probability and D_n the fault detection probability. The smaller the index value for O_n, A_n or D_n the more limited is the effect of a certain fault. For instance, the occurrence probability bandwidth reaches from 100.000 ppm to 1 ppm. For a very unreliable sensor which generates a fault every tenth measurement or more, we define $O_n = 10$. The amplitude of a fault is normalized by the measurement range of a sensor. If the fault level reaches the range of the sensor (1), we set the fault amplitude index value to 10. The detection probability measures the effectiveness of the testing mechanisms and assigns a appropriate index value to D_n. The index system follows the FMEA-classification standard of the automobile industry union standard (VDA 4.2) [27] but was adapted related to the higher fault level of distributed sensor actuator applications.

When the index entries for A_n, O_n and D_n are determined, the three indices must merged to a single number. FMEA recommends multiplying the index numbers. Hence, for a perfect system we will reach a result of 1. For a system with the lowest reliability without any detection possibility FMEA generates 1000. This number is known as the Risk Priority Number (RPN). In case of multiple fault types n the maximum fault specific RPN_n defines the system RPN, $RPN = max(RPN_i) = max(A_i \cdot O_i \cdot D_i)$ with $1 \leq i \leq n$.

In our fault-handling concept and in relation to the *event validity* we call it *system validity*. System validity combines and encapsulates the spectrum of anticipated sensor faults and the associated fault detection methods.

TABLE I
QUANTIFICATION OF FAULT'S EFFECT BASED ON ITS AMPLITUDE, OCCURRENCE AND DETECTION PROBABILITY

Index	Fault amplitude (A) to range	Occurrence (O) ppm	Detection (D) probability
10	≥1	100.000	≤ 50 %
9		50.000	>50
8		20.000	75 %
7	≥0.1	10.000	
6		5.000	90 %
5		2.000	
4	≥0.01	1.000	
3		100	99 %
2		50	
1	≥0.001	1	99.99 %

A MOSAIC node provides an electronic data sheet containing an description of sensor parameters, message formats, etc. Here we extend existing description approaches [25], [12]. The relevant fault types n and its parameters A_n, O_n and D_n are made available for other nodes. This normalization of the validity value allows a node to assess a remote sensor measurement and include or reject it in the local control algorithm.

VI. EXAMPLE

We implemented the MOSAIC approach in distributed robotic applications. External and internal sensors information is flexibly merged in this application for localization, navigation, environment monitoring and collision avoidance. To illustrate the benefits of the described fault abstraction, we present the processing chain of an infrared distance sensor.

TABLE II
FAULT CLASSIFICATION FOR INFRARED SENSORS GP2D12

	Fault type	Fault ampli. (A)	Occur. prob. (O)	System validity without detection ($A \cdot O \cdot 10$)
Outlier	②	10	6	600
Spikes	⑦	8	8	640
Offset	③ + ④	8	9	720
Offset	③	10	4	400

Tab. II structures the fault types of the sensor type [28] and their parameters for fault amplitude (F_n)and for occurrence probability (O_n). Based on extensive experiments with this sensor type [29] we identified the following four fault types:

- Outliers are single samples with a significant displacement related to the correct value. Mostly the maximum amplitude of the sensor is reached ($F = 10$). Outliers occur for $\approx 0.5\%$ of all measurements. The corresponding value for O derived from Tab. I is 6.
- Spikes are more frequently than outliers $\approx 2\%$ but generate only a limited deviation ($F = 8, O = 8$).
- Two variants of offsets were observed. Both have different effects. The first one is caused by external strong light disturbing the physical measurement process. The

deviations of the faulty measurements range from $8\,\mathrm{cm}$ to $-18\,\mathrm{cm}$. Additionally, the noise level and spectrum is change. Hence, we have to consider a combination of fault types in this case ($F = 8, O = 9$).

- The fourth fault occurs due to varying battery power of the sensor nodes. If voltage level of the battery level decreases a certain level, the Analog-to-Digital Converter (ADC) does not work correctly. The level of this disturbance varies up to $5\,\mathrm{cm}$ ($F = 10, O = 4$).

An indication for the low validity of the raw sensor system is the derived maximum RPN of 720. Without any further mechanism to detect and handle faults, the confidence is only very low and of limited value in a critical application. Tab. III assigns a number of fault detection methods to each fault type and adapts the RPN value accordingly. Outliers can be detected with several methods. In the example we implemented an interval and a gradient check. Both detection mechanisms provide a high probability of discarding a faulty measurement. The interval (or range) check verifies whether the sensor output is within its bounds specified by the physical sensor characteristics. The gradient check compares the derivation of the system with a maximum slope. Due to signal noise, the gradient method shows a lower detection probability than the interval check ($D = 3$). This is not too surprising because the interval check only detects a rather significant violation resulting from an outlier, while for the gradient method it is more difficult to decide whether the slope of a signal is out of range.

Due to the smaller gradient, spikes are more difficult to detect than outliers. Consequently, the gradient check does not reach the same detection probability ($RPN = 384$). The RPN value depends on the number of available sensor readings.

For a detecting an offset caused by strong external light a statistical test was presented in [29]. It requires a series of measurements and a comparison with a reference sample. The correct fault classification is delayed depending on the number of consecutive measurements. Due to this limitation the detection probability index D is set to 6 as specified in Tab. I

Based on Tab. III an analysis of different variants of the system is possible. If a single sensor application integrates only interval and gradient check, the system cannot correctly detect any offset fault. The common system validity is given by the highest value $max(min(60, 120), 384, 720, 400) = 720$. With an implementation of the statistical test the RPN goes down to $max(min(60, 120), 384, 504, 400) = 504$. The confidence increases by one third!

By applying the complete fault detection chain, a GP2D12 produces a distance result in which we can put high confidence. Each of the relevant fault type is covered by at least one detection method.

If multiple redundant sensors are available it is generally very appealing to move parts of the error detection to a fusion node that processes the multiple values and, according to the hierarchical MOSAIC concept, constitutes a (higher level) reliable abstract sensor. Some faults, e.g. offsets caused by an incorrect reference voltage can only be recognized in such a fusion node. For our example scenario the detection quality depends on the number of redundant measurements, as visible in Tab. III, for all fault types. It reaches the highest level for five available data sets. The maximum system validity can be defined by $max(min(60, 120, 60), min(384, 128), min(504, 216), 120) = 144$ in this case. For four or three available measurements we obtain an increased RMP of $max(min(60, 120, 120), min(384, 192), min(504, 216), 120) = 216$.

It should be noted that the flexible distribution of error detection is substantially supported by the MOSAIC concept. Every measurement is encapsulated in a well-defined event disseminated by a sensor. This event carries attributes, described in an electronic data sheet, that allow precisely assessing the confidence in the data provided. An application can even decide whether to use a more uncertain value that is available very fast and very frequently or the more reliable result disseminated by a fusion node. The subject-based publish/subscribe communication system supports such flexible choices conveniently.

A second advantage from this flexible distribution of detection capabilities comes in when considering implementation issues. In many applications, e.g. wireless sensor networks or also inside a car, the processing capabilities of networked sensor components are rather limited. E.g. performing a statistical test on offset fault detection in a reasonable time frame requires more computational power than it is available from the embedded controller. MOSAIC enables the distribution of test in a generic and easy to use way.

TABLE III
SYSTEM VALIDATION FOR GP2D12 SENSOR AND ITS PROCESSING CHAIN (DARK GRAY ROWS MARKS DETECTION METHODS EXECUTED ON A EMBEDDED SENSOR NODE, LIGHT GRAY POWERFUL PROCESSING NODES AND WHITE LINES INDICATES THE FUSION NODE.)

	System validity without detection ($A \cdot O \cdot 10$)	Assigned method	Detect. prob. (D)	System validity ($A \cdot O \cdot D$)
Outlier	600	Interval Check	1	60
	600	Gradient check	3	120
	600	Comparator (3)	2	120
	600	Comparator (4)	2	120
	600	Comparator (5)	1	60
Spikes	640	Gradient check	6	384
	640	Comparator (3)	3	192
	640	Comparator (4)	3	192
	640	Comparator (5)	2	128
Offset	720	Statistical test	7	504
	720	Comparator (3)	3	216
	720	Comparator (4)	3	216
	720	Comparator (5)	2	144
Offset	400	Comparator (3)	3	120
	400	Comparator (4)	3	120
	400	Comparator (5)	2	80

Fig. 2 depicts the implementation of the distributed infrared sensor processing chain. The diagrams illustrate the effect of

the fault detection and handling in a distributed application integrating a sensor node, a processing node, and fusion node. The figures depict the raw sensor measurement, the validity of an individual measurement event and the overall system validity. The system validity, represented by the RPN, is a confirmation about the trustworthiness of the event validity. E.g. an error may have been detected in a measurement by a local mechanism. Then the event validity would go down to a low level, however, the system validity would remain at the same level. The node receiving this event can infer from these two values that the low event validity was obtained with respectively high confidence.

The raw distance measurement signal is affected by the four fault types listed in Tab. II. The sensor node locally applies the gradient and interval check. The second diagram shows the corresponding event validity based on the detection results. The gradient and interval check generate a binary decision depicted in the event validity graph. As expected, outlier and spikes are recognized very well. Offset faults are only detected if the gradient is sufficiently high. But they are associated with the wrong fault type. The system validity is constantly on a low level ($RPN = 720$).

For the more sophisticated statistical test for offsets, a more powerful processing node is used. This node receives the distance events that have been preprocessed, filters incoming measurements with an event validity below a defined threshold and calculates the probability of an external light disturbance. The resulting event validity shows a delayed but correct detection of the error caused by an environmental condition violating the sensor specification. The system validity is improved and reaches the lower numerical RPN value of 504.

The last stage of deriving a highly valid distance value is a node that fuses events from multiple distance sensors. Here, the system validity is an important criterion for acceptance the fusion node. It only accepts measurements with a RPN below a defined threshold. We assume that all five sensors meet this requirement and pass the first stage of the acceptance test. In a second step the individual event validities are checked. All events with a validity below 0.6 are rejected. The other measurements are considered for comparison. Now the last filter works on the value itself according to the techniques of fault-tolerant mean or midpoint, i.e. measurements with the largest deviations are not considered. As can be seen in the graph, the system validity changes. This is because some events are discarded leading to a smaller set of measurements used in the mean calculation. According to Tab. III the RPN now reaches a level of 120. Again, the example shows the multiple stages of event processing which are supported by the MOSAIC system eventually reaching a high level of confidence.

VII. Conclusions

Dealing with the complex failure and error modes of continuous valued sensor data is a substantial challenge in distributed mobile applications. Exploiting remote sensors installed in an instrumented environment and provided by the on-board

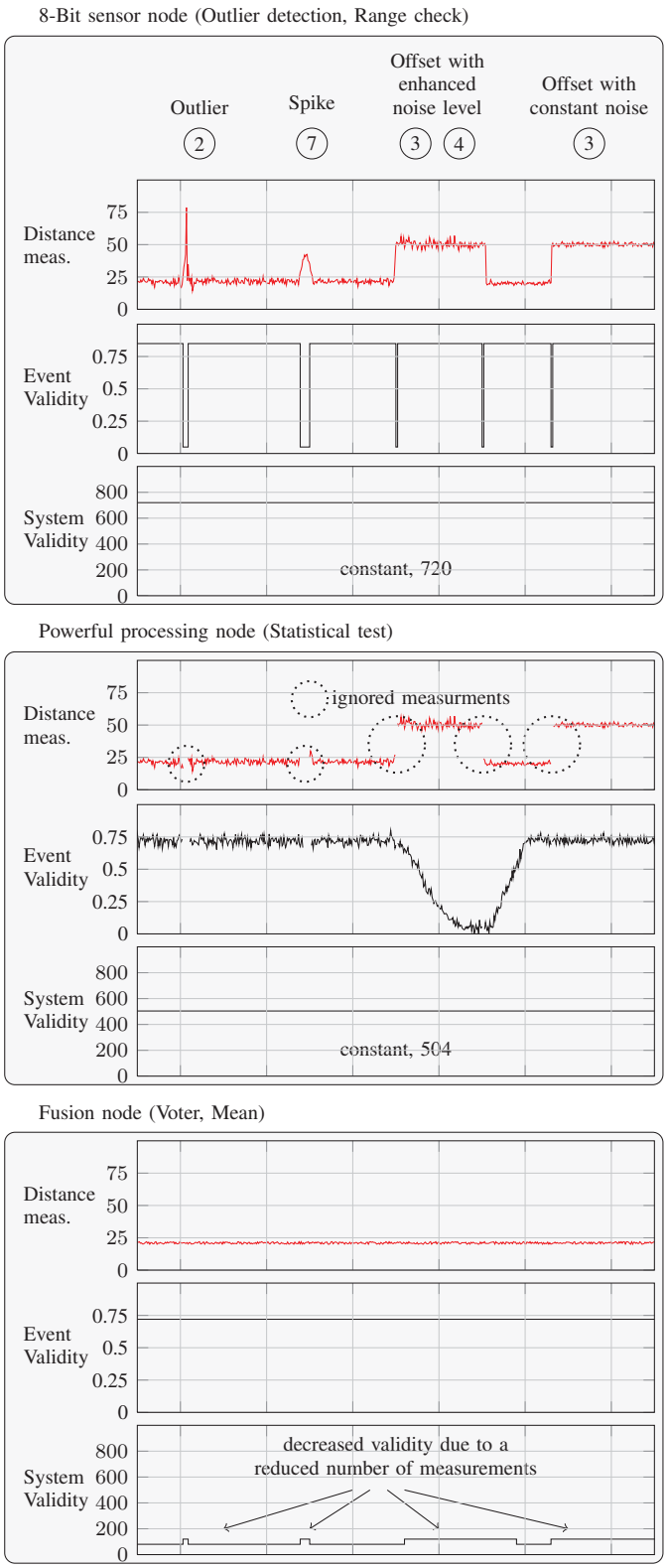


Fig. 2. Implementation of the multi-level fault detection for an IR distance sensor using MOSAIC - Diagram (adapted from [26])

sensors of mobile vehicles is very desirable. However, the designer of a local control algorithm, e.g. for trajectory planning or obstacle avoidance can use very little a priori knowledge about the sensor system. Prerequisit for using remote sensor data is that it has to be accompanied by attributes describing its contents, context and the quality of data. This paper focuses on the on-line assessment of the sensor data quality and validity for building dependable distributed sensor applications. We propose MOSAIC that provides a uniform programming model and an interface for the designer of such systems. The internal structure of a MOSAIC smart sensor node comprises a number of checking components each tailored for a specific sensor fault class. This allows analysing and mapping the complex sensor malfunctions to a well-defined validity value. The uniform model encourages a multi-level checking structure. It also provides an easy to use assessment scheme for the application designer that can select the most appropriate data from a set of available sensor sources. This work is part of larger projects that deals with a dependable infrastructure for mobile cooperative scenarios. So far, we work with mobile robots that navigate in an instrumented environment. Main research directions are machine exploitable descriptions of sensor and actuator characteristics, the dynamic discovery and use of remote sensor data, and on dependability with an emphasis on functional safety. Future work will extend the possibilities of dynamically discovered sensors in an unknown environment. Research will include geometric environment models that will further enable to estimate the usefulness of a sensor in a 3-D space. We work on the concept of a situated, directed sensor. This additional information will lead to a truly spontaneous use of remote sensing information with a minimal amount of a priori knowledge.

ACKNOWLEDGMENT

This work has partially supported by the EU under the FP7-ICT programme, through project 288195 “Kernel-based ARchitecture for safetY-critical cONtrol” (KARYON) and by the Ministry of Education and Science (BMBF) within the project “Virtual and Augmented Reality for Highly Safety and Reliable Embedded Systems” (ViERforES - no. 01IM08003C).

REFERENCES

[1] J. Rushby, “Reconfiguration and transient recovery in state-machine architectures,” in *Fault Tolerant Computing Symposium 26*. Sendai, Japan: IEEE Computer Society, 6 1996, pp. 6–15.

[2] C. B. Weinstock and J. Goldberg, “Sift : Software implemented fault-tolerance,” in *Fault Tolerant Computing Symposium 26*. Madison, Wisconsin, USA: IEEE Computer Society, 6 1979.

[3] Frank, P.M., “Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy,” *Automatica*, vol. 26, no. 3, pp. 459–474, 1990.

[4] G. Heredia, A. Ollero, A. Bejar, and R. Mahtani, “Sensor and actuator fault detection in small autonomous helicopters,” *Mechatronics*, vol. 18, no. 2, pp. 90–99, 2008.

[5] F. Cristian, “Understanding fault-tolerant distributed systems,” *Commun. ACM*, vol. 34, pp. 56–78, 2 1991.

[6] ——, “A rigorous approach to fault-tolerant programming,” *Software Engineering, IEEE Transactions on*, no. 1, pp. 23–31, 1985.

[7] B. E. H. Otto Wittner, Carsten J.E. Hoelper, “Failure semantics of mobile agent systems involved in network fault management,” in *Proceedings of Norsk Informatikk-Konferanse (NIK’99)*, Trondheim, Norway, 11 1999.

[8] V. Hadzilacos and S. Toueg, “A modular approach to the specification and implementation of fault-tolerant broadcasts,” Department of Computer Science, Cornell University, Ithaca NY., Tech. Rep. Technical Report TR94-1425, 5 1994.

[9] K. Marzullo, “Tolerating Failures of Continuous-Valued Sensors,” *ACM Transactions on Computer Systems (TOCS)*, vol. 8, no. 4, pp. 284–304, 11 1990.

[10] A. Avizienis, J. Laprie, and B. Randell, “Fundamental concepts of dependability,” University of Newcastle upon Tyne, Tech. Rep., 2001.

[11] A. Sharma, L. Golubchik, and R. Govindan, “On the prevalence of sensor faults in real-world deployments,” in *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON’07. 4th Annual IEEE Communications Society Conference on*. IEEE, 2007, pp. 213–222.

[12] J. Kaiser and H. Piontek, “CODES: Supporting the development process in a publish/subscribe system,” in *Proceedings of the fourth Workshop on Intelligent Solutions in Embedded Systems WISES 06*, Vienna, 6 2006, pp. 1–12, iSBN: 3-902463-06-6.

[13] S. Sukumar, H. Bozdogan, D. Page, A. Koschan, and M. Abidi, “Sensor selection using information complexity for multi-sensor mobile robot localization,” in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, pp. 4158–4163.

[14] W. Elmenreich, S. Pitzek, and M. Schlager, “Modeling Distributed Embedded Applications on an Interface File System,” in *Proceedings of the Seventh IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC’04)*, Vienna, Austria, 2004, pp. 175–182.

[15] H. Kopetz, M. Holzmann, and W. Elmenreich, “A universal smart transducer interface: TTP/A,” in *Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*. Newport Beach, California: Published by the IEEE Computer Society, 3 2000, p. 16.

[16] W. Ko and C. Fung, “Vlsi and intelligent transducers,” *Sensors and actuators*, vol. 2, pp. 239–250, 1982.

[17] A. Moini, “Vision chips or seeing silicon,” Report of Department of Electronic Engineering, University of Adelaide, Australia, Tech. Rep., 3 1997.

[18] H. Kopetz, M. Holzmann, and W. Elmenreich, “A universal smart transducer interface: Ttp/a,” *International Journal of Computer System Science & Engineering*, no. 2, pp. 71–77, 3 2001.

[19] R. M. Newman and E. I. Gaura, “System issues in arrays of autonomous intelligent sensors,” *Technical Proceedings of the NSTI Nanotechnology Conference and Trade Show*, vol. 1, 2004.

[20] A. Fitzpatrick, G. Biegel, S. Clarke, and V. Cahill, “Towards a sentient object model,” in *Workshop on Engineering Context-Aware Object Oriented Systems and Environments (ECOOSE)*. Citeseer, 2002.

[21] A. Casimiro, J. Kaiser, and P. Verissimo, “Generic-events architecture: Integrating real-world aspects in event-based systems,” *Lecture Notes in Computer Science (Architecting Dependable Systems IV)*, vol. Volume 4615, pp. 287–315, 2007.

[22] J. Kaiser, M. Schulze, S. Zug, C. Cardeira, and F. Carreira, “Sentient objects for designing and controlling service robots,” in *Proccedings of IFAC’08*, vol. 17th International Federation of Automatic Control World Congress, Seoul, Korea, 7 2008, pp. 8315–8320.

[23] D. Mills, “Network time protocol (ntp),” *Network*, 1985.

[24] S. Zug, M. Schulze, A. Dietrich, and J. Kaiser, “Programming abstractions and middleware for building control systems as networks of smart sensors and actuators,” in *Proceedings of Emerging Technologies in Factory Automation (ETFA ’10)*, Bilbao, Spain, 9 2010.

[25] IEEE Standards Association, *IEEE Standard for a Smart Transducer Interface for Sensors and Actuators (IEEE 1451.2)*, 1997. [Online]. Available: http://ieeexplore.ieee.org/xpl/standards.jsp?findtitle=1451&letter=1451

[26] S. Zug, A. Dietrich, and J. Kaiser, *Fault-Handling in Networked Sensor Systems*. St. Franklin, AUS: Concept Press Ltd., 2012.

[27] Verband der Automobilindustrie e.V., *Qualitätsmanagement in der Automobilindustrie - Sicherung der Qualität vor Serieneinsatz*, 1996.

[28] Sharp Cooperation, *GP2D120 Data Sheet*, online, url = http://sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a21yk_e.pdf, 2007.

[29] A. Dietrich, S. Zug, and J. Kaiser, “Detecting External Measurement Disturbances Based on Statistical Analysis for Smart Sensors,” in *Procedings of the IEEE International Symposium on Industrial Electronics (ISIE)*, 7 2010, pp. 2067–2072.

A.3 Reliable Assessment of Global State

A.3.1 Self-Stabilizing Byzantine Resilient Topology Discovery and Message Delivery

"Self-Stabilizing Byzantine Resilient Topology Discovery and Message Delivery". S. Dolev, O. Liba, E. M. Schiller, CoRR abs/1208.5620, August 2012. http://arxiv.org/abs/1208.5620

This page is intentionally left blank.

Self-Stabilizing Byzantine Resilient Topology Discovery and Message Delivery

(Technical Report)

Shlomi Dolev * Omri Liba * Elad M. Schiller †

Abstract

Traditional Byzantine resilient algorithms use $2f+1$ vertex disjoint paths to ensure message delivery in the presence of up to f Byzantine nodes. The question of how these paths are identified is related to the fundamental problem of topology discovery.

Distributed algorithms for topology discovery cope with a never ending task, dealing with frequent changes in the network topology and unpredictable transient faults. Therefore, algorithms for topology discovery should be self-stabilizing to ensure convergence of the topology information following any such unpredictable sequence of events. We present the first such algorithm that can cope with Byzantine nodes. Starting in an arbitrary global state, and in the presence of f Byzantine nodes, each node is eventually aware of all the other non-Byzantine nodes and their connecting communication links.

Using the topology information, nodes can, for example, route messages across the network and deliver messages from one end user to another. We present the first deterministic, cryptographic-assumptions-free, self-stabilizing, Byzantine-resilient algorithms for network topology discovery and end-to-end message delivery. We also consider the task of r-neighborhood discovery for the case in which r and the degree of nodes are bounded by constants. The use of r-neighborhood discovery facilitates polynomial time, communication and space solutions for the above tasks.

The obtained algorithms can be used to authenticate parties, in particular during the establishment of private secrets, thus forming public key schemes that are resistant to man-in-the-middle attacks of the compromised Byzantine nodes. A polynomial and efficient end-to-end algorithm that is based on the established private secrets can be employed in between periodical re-establishments of the secrets.

1 Introduction

Self-stabilizing Byzantine resilient topology discovery is a fundamental distributed task that enables communication among parties in the network even if some of the components are compromised by an adversary. Such topology discovery is becoming extremely important nowadays where countries main infrastructures, such as the electrical smart-grid, water supply networks and intelligent transportation systems are subject

*Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel. Email: {dolev, liba}@cs.bgu.ac.il. Partially supported by Deutsche Telekom, Rita Altura Trust Chair in Computer Sciences, Lynne and William Frankel Center for Computer Sciences, Israel Science Foundation (grant number 428/11) and Cabarnit Cyber Security MAGNET Consortium.

†Department of Computer Science and Engineering, Chalmers University of Technology, Goeteborg, Sweden. Email: elad@chalmers.se. Partially supported by the EC, through project FP7-STREP-288195, KARYON (Kernel-based ARchitecture for safetY-critical cONtrol) and the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 257007.

to cyber-attacks. Self-stabilizing Byzantine resilient algorithms naturally cope with mobile attacks [e.g., 16]. Whenever the set of compromised components is fixed (or dynamic, but small) during a period that suffice for convergence of the algorithm the system starts demonstrating useful behavior following the convergence. For example, consider the case in which nodes of the smart-grid are constantly compromised by an adversary while local recovery techniques, such as local node reset and/or refresh, ensure the recovery of a compromised node after a bounded time. Once the current compromised set does not imply a partition of the communication graph, the distributed control of the smart grid automatically recovers. Self-stabilizing Byzantine resilient algorithms for topology discovery and message delivery are important for systems that have to cope with unanticipated transient violations of the assumptions that the algorithms are based upon, such as unanticipated violation of the upper number of compromised nodes and unanticipated transmission interferences that is beyond the error correction code capabilities.

The dynamic and difficult-to-predict nature of electrical smart-grid and intelligent transportation systems give rise to many fault-tolerance issues and require efficient solutions. Such networks are subject to transient faults due to hardware/software temporal malfunctions or short-lived violations of the assumed settings for the location and state of their nodes. Fault-tolerant systems that are *self-stabilizing* [5] can recover after the occurrence of transient faults, which can drive the system to an arbitrary system state. The system designers consider *all* configurations as possible configurations from which the system is started. The self-stabilization design criteria liberate the system designer from dealing with specific fault scenarios, risking neglecting some scenarios, and having to address each fault scenario separately.

We also consider Byzantine faults that address the possibility of a node to be compromised by an adversary and/or to run a corrupted program, rather than merely assuming that they start in an arbitrary local state. Byzantine components may behave arbitrarily (selfishly, or even maliciously) as message senders and/or as relaying nodes. For example, Byzantine nodes may block messages, selective omit messages, redirect the route of messages, playback messages, or modify messages. Any system behavior is possible, when all (or one third or more of) the nodes are Byzantine nodes. Thus, the number of Byzantine nodes, f, is usually restricted to be less than one third of the nodes [5, 13].

The task of *r-neighborhood network discovery* allows each node to know the set of nodes that are at most r hops away from it in the communication network. Moreover, the task provides information about the communication links attached to these nodes. The task *topology discovery* considers knowledge regarding the node's entire connected component. The r-neighborhood network discovery and network topology discovery tasks are identical when r is the diameter of the communication graph.

This work presents the first deterministic self-stabilizing algorithms for r-neighborhood discovery in the presence of Byzantine nodes. We assume that every r-neighborhood cannot be partitioned by the Byzantine nodes. In particular, we assume the existence of at least $2f + 1$ vertex disjoint paths in the r-neighborhood, between any two non-Byzantine nodes, where at most f Byzantine nodes are present in the r-neighborhood, rather than in the entire network. [1] Note that by the self-stabilizing nature of our algorithms, recovery is guaranteed after a temporal violation of the above assumption. When r is defined to be the diameter of the communication graph, our assumptions are equivalent to the standard assumption for Byzantine agreement in general (rather than only complete) communication graphs. In particular the standard assumption is that $2f + 1$ vertex disjoint paths exist and *are known* (see e.g., [13]) while we present distributed algorithms to find these paths starting in an arbitrary state.

Related work. Self-stabilizing algorithms for finding vertex disjoint paths for at most two paths between

[1] Section 4 considers cases in which r and the node degree, Δ, are constants. For these case, we have $\mathcal{O}(n)$ disjoint r-neighborhoods. Each of these (disjoint) r-neighborhoods may have up to f Byzantine nodes, and yet the above assumptions, about at least $2f + 1$ vertex disjoint paths in the r-neighborhood, hold.

any pair of nodes, and for all vertex disjoint paths in anonymous mesh networks appear in [1] and in [11], respectively. We propose self-stabilizing Byzantine resilient procedures for finding $f + 1$ vertex disjoint paths in $2f + 1$-connected graphs. In [9], the authors study the problem of spanning tree construction in the presence of Byzantine nodes. Nesterenko and Tixeuil [15] presented a deterministic *non-stabilizing* algorithm for topology discovery in the presence of Byzantine nodes. The authors do not consider the automatic recovery implied by the self-stabilization property. [[Awerbuch and Sipser [3] consider algorithms that were designed for synchronous static network and give topology update as an example. They show how to use such algorithms in asynchronous dynamic networks. Unfortunately, their scheme starts from a consistent state and cannot cope with transient faults or Byzantine.]]

Byzantine gossip [2, 4, 6, 10, 12, 14] and *Byzantine Broadcast* [8, 17] consider the dissemination of information in the presence of Byzantine nodes rather than self-stabilizing topology discovery. Non-self-stabilizing Byzantine resilient gossip in the presence of one selfish node is considered in [2, 12]. In [6], the authors study oblivious deterministic gossip algorithms for multi-channel radio networks with a malicious adversary. They assume that the adversary can disrupt one channel per round, preventing communication on that channel. In [4], the authors consider probabilistic gossip mechanisms for reducing the redundant transmissions of flooding algorithms. They present several protocols that exploit local connectivity to adaptively correct propagation failures and protect against Byzantine attacks. Probabilistic gossip mechanisms in the context of recommendations and social networks are considered in [10]. In [14] the authors consider rules for avoiding a combinatorial explosion in (non-self-stabilizing) gossip protocol. Note that deterministic and self-stabilizing solutions are not presented in [2, 4, 6, 10, 12, 14].

Drabkin et al. [8] consider non-self-stabilizing broadcast protocols that overcome Byzantine failures by using digital signatures, message signature gossiping, and failure detectors. Our deterministic self-stabilizing algorithm merely use the topological properties of the communication graph to ensure that messages dropped or modified by Byzantine nodes will be detected, and retransmitted in a way that guarantees correct delivery to the application layer. A non-self-stabilizing broadcasting algorithm is considered in [17]. The authors assume the restricted case in which links and nodes of a communication network are subject to Byzantine failures, and that faults are distributed randomly and independently.

Our contribution. We present two cryptographic-assumptions-free yet secure algorithms that are deterministic, self-stabilizing and Byzantine resilient.

We start by showing the existence of deterministic, self-stabilizing, Byzantine resilient algorithms for network topology discovery and end-to-end message delivery. [[The algorithms convergence time is in $\mathcal{O}(n)$. They take in to account every possible path and requiring bounded (yet exponential) memory and bounded (yet exponential) communication costs.]] Therefore, we also consider the task of r-neighborhood discovery, where r is a constant. We assume that if the r-neighborhood of a node has f Byzantine nodes, there are $2f + 1$ vertex independent paths between the node and any non-Byzantine node in its r-neighborhood. The obtained r-neighborhood discovery requires polynomial memory and communication costs and supports deterministic, self-stabilizing, Byzantine resilient algorithm for end-to-end message delivery across the network. [[Unlike topology update, the proposed end-to-end message delivery algorithm establishes message exchange synchronization between end-users that is based on message reception acknowledgments.]]

Document structure. Settings and requirements appear in Section 2. The self-stabilizing Byzantine resilient distributed algorithm for topology discovery is presented in Section 3. The end-to-end communication algorithm appears in Section 4. Extensions and concluding remarks appear in Section 5. Detailed proofs appear in the Appendix and in [7].

2 Preliminaries

We consider settings of a standard asynchronous system [cf. 5]. The system consists of a set, $N = \{p_i\}$ of communicating entities, chosen from a set P, which we call *nodes*. The upper bound on the number of nodes in the system is $n = |P|$. Each node has a unique identifier. Sometime we refer to a set, $P \setminus N$, of nonexisting nodes that a false indication on their existence can be recorded in the system. A node p_i can directly communicate with its *neighbors*, $N_i \subseteq N$. The system can be represented by a network of directly communicating nodes, $G = (N, E)$, named the *communication graph*, where $E = \{(p_i, p_j) \in N \times N : p_j \in N_i\}$. We denote N_k's set of indices by $indices(N_k) = \{m : p_m \in N_k\}$ and N_k's set of edges by $edges(N_j) = \{p_j\} \times N_j$.

The r-neighborhood of a node $p_i \in N$ is the connected component that includes p_i and all nodes that can be reached from p_i by a path of length r or less. The r-neighborhood version of the algorithm for network topology discovery considers communication graphs in which the number of neighbors of a node p_i is bounded by a constant Δ. Hence, when both the neighborhood radius, r, and the node degree Δ are constants the number of nodes in the r-neighborhood is also bounded by a constant, namely by [[$\mathcal{O}(\Delta^{r+1})$.]]

We model the communication channel, $queue_{i,j}$, from node p_i to node $p_j \in N_i$ as a FIFO queuing list of the messages that p_i has sent to p_j and p_j is about to receive. When p_i sends message m, the operation send inserts a copy of m to every $queue_{i,j}$, such that $p_j \in N_i$. We assume that the number of messages in transit, i.e., stored in $queue_{i,j}$, is at most $capacity$. Once m arrives, p_j executes receive and m is dequeued.

We assume that p_i is completely aware of N_i, as in [15]. In particular, we assume that the identity of the sending node is known to the receiving one. In the context of the studied problem, we say that node $p_i \in N$ is *correct* if it reports on its genuine neighborhood, N_i. A *Byzantine* node, $p_b \in N$, is a node that can send arbitrarily corrupted messages. Byzantine nodes can introduce new messages and modify or omit messages that pass through them. This way they can, e.g., disinform correct nodes about their neighborhoods, or about the neighborhood of other correct nodes, or the path through which messages travel, to name a very few specific misleading actions that Byzantine nodes may exhibit. We denote by C and B the set of correct, and respectively, Byzantine nodes. We assume that $|B| = f$, the identity of the nodes in B is unknown to the nodes in C. Nevertheless, B is fixed throughout the considered execution segment. These execution segments are long enough for convergence and then for obtaining sufficient useful work. We assume that between any pair of correct nodes there are at least $2f + 1$ vertex disjoints paths. We denote by $G_c = (C, E \cap C \times C)$ the *correct graph* induced by the set of correct nodes.

Self-stabilizing algorithms never terminate (see [5]). The non-termination property can be easily identified in the code of a self-stabilizing algorithm: the code is usually a do forever loop that contains communication operations with the neighbors. An iteration is said to be complete if it starts in the loop's first line and ends at the last (regardless of whether it enters branches).

Every node, p_i, executes a program that is a sequence of *(atomic) steps*. For ease of description, we assume the interleaving model where steps are executed atomically, a single step at any given time. An input event can either be the receipt of a message or a periodic timer going off triggering p_i to send. Note that the system is totally asynchronous and the (non-fixed) node processing rates are irrelevant to the correctness proof.

The *state* s_i of a node p_i consists of the value of all the variables of the node (including the set of all incoming communication channels, $\{queue_{j,i} | p_j \in N_i\}$). The execution of a step in the algorithm can change the state of a node. The term *(system) configuration* is used for a tuple of the form $(s_1, s_2, \cdots, s_n)$, where each s_i is the state of node p_i (including messages in transit for p_i). We define an *execution* $E = c[0], a[0], c[1], a[1], \ldots$ as an alternating sequence of system configurations $c[x]$ and steps $a[x]$, such that each

configuration $c[x+1]$ (except the initial configuration $c[0]$) is obtained from the preceding configuration $c[x]$ by the execution of the step $a[x]$. We often associate the notation of a step with its executing node p_i using a subscript, e.g., a_i. An execution R (run) is *fair* if every correct node, $p_i \in C$, executes a step infinitely often in R. Time (e.g. needed for convergence) is measured by the number of *asynchronous rounds*, where the first asynchronous round is the minimal prefix of the execution in which every node takes at least one step. The second asynchronous round is the first asynchronous round in the suffix of the run that follows the first asynchronous round, and so on. The message complexity (e.g. needed for convergence) is the number of messages measured in the specific case of synchronous execution.

We define the system's task by a set of executions called *legal executions* (LE) in which the task's requirements hold. A configuration c is a *safe configuration* for an algorithm and the task of LE provided that any execution that starts in c is a legal execution (belongs to LE). An algorithm is *self-stabilizing* with relation to the task LE when every infinite execution of the algorithm reaches a safe configuration with relation to the algorithm and the task.

3 Topology Discovery

The topology discovery is based on accumulating messages from vertex disjoint paths. Each message contains an ordered list of nodes it passed so far, starting in a source node, and a neighborhood, which is the set of nodes, claimed to be directly connected to the source.

Each node p_i periodically sends a message to each neighbor. The message sent contains the local topology, a source i and an empty path. The arrival of a message m to p_i triggers an insert of m to $informedTopology_i$ and a consistency test of the content of $informedTopology_i$. The consistency test results in storing local topologies for which there are enough independent evidence in a result array. The result array is initialized just prior to the consistency test. The consistency test of p_i iterates over each node p_k such that, p_k appears in at least one of the messages stored in $informedTopology_i$. For each such node p_k, node p_i checks whether there are at least $f+1$ messages from the same source node that have mutually vertex disjoint paths and report on the same neighborhood. The neighborhood of each such p_k, that has at least $f+1$ vertex disjoint paths with identical neighborhood, is accumulated in $Result[k]$. Moreover, the total number of paths [[that]] relayed this neighborhood is kept in $Count[k]$.

We note that there may still be nodes $p_{fake} \in P \setminus (C \cup B)$, for which there is an entry $Result[fake]$. For example, $informedTopology$ may contain f messages, all originated from different Byzantine nodes, and a message m' that appears in the initial configuration and supports the (false) neighborhood the Byzantine messages refer to. These $f+1$ messages can contain mutually vertex disjoint paths, and thus during the consistency test, a result will be found for $Result[fake]$. We show that during the next computations, the message m' will be identified and ignored.

The $Result$ set should include two reports for each (undirected) edge; the two nodes that are attached to the edge, each send a report. Hence, $Result$ includes a set of directed (report) edges. The term *contradicting edge* is needed when examining the $Result$ set consistency.

Definition 1 (Contradicting edges) *Given two nodes, $p_i, p_j \in P$, we say that the edge (p_i, p_j) is* contradicting with the set $Neighborhood_j \subseteq edges(N_j)$, *if* $(p_i, p_j) \notin Neighborhood_j$.

Following the consistency test, p_i examines the $Result$ array for contradictions. Node p_i checks the path of each message $m \in informedTopology_i$ with source p_r, neighborhood $neighborhood_r$ and $Path_r$. If every edge (p_s, p_j) on the path appears in $Result[s]$ and $Result[j]$, then we move to the next message. Otherwise, we found a fake supporter, and therefore we reduce $Count[r]$ by one. In case the resulting $Count[r]$ is smaller than $f+1$, we nullify the r'th entry of the $Result$ array. Once all messages were

processed, the $Result$ array consisting of the (confirmed) local topologies is the output. At the end p_i forwards the arriving message m to each neighbor that does not appear in the path of m. The message sent by p_i includes the node from which m arrived as part of the path m.

The pseudocode appears in Algorithm 1. In every iteration of the infinite loop, p_i starts to compute its preliminary topology view by calling $ComputeResults$ in line 2. Then, every node p_k in the queue $InformedTopology$, node p_i goes over the messages in the queue from head to bottom. While iterating the queue, for every message m with source p_k, neighborhood N_k and visited path $Path_k$, p_i inserts $Path_k$ to $opinion[N_k]$, see line 18. After inserting, p_i checks if there is a neighborhood $Neig_k$ for which $opinion[Neig_k]$ contains at least [[$f + 1$]] vertex disjoint paths, see line 19. When such a neighborhood is found, it is stored in the $Result$ array (line 19). In line 20, p_i stores the number of vertex disjoint paths relayed messages that contained the selected neighborhood for p_k. After computing an initial view of the topology, in line 3, p_i removes non-existing nodes from the computed topology. For every message m in $InformedTopology$, node p_i aims at validating its visited path. In line 24, p_i checks if there exists a node p_k whose neighborhood contradicts the visited path of m. If such a node exists, p_i decreases the associated entry in the $Count$ array (line 25). This decrease may cause $Count[r]$ to be smaller than $f + 1$, in this case p_i considers p_k to be fake and deletes the local topology of p_k from $Result[r]$ (line 26).

Upon receiving a message m, node p_i inserts the message to the queue, in case it does not already exist, and just moves it to the top of the queue in case it does. The node p_i now needs to relay the message p_i got to all neighbors that are not on the message visited path (line 9). When sending, p_i also attaches the identifier of the node, from which the message was received, to the visited path of the message.

Algorithm's correctness proof. We now prove that within a linear amount of asynchronous rounds, the system stabilizes and every output is legal. The proof considers an arbitrary starting configuration with arbitrary messages in transit that could be actually in the communication channel or already stored in p_j's message queue and will be forwarded in the next steps of p_j. Each message in transit that traverse correct nodes can be forwarded within less than $\mathcal{O}(|C|)$ asynchronous rounds. Note that any message that traverses Byzantine nodes and arrives to a correct node that has at least one Byzantine node in its paths. The reason is that the correct neighbor to the last Byzantine in the path lists the Byzantine node when forwarding the message. Thus, f is at most the number of messages that encode vertex disjoint paths from a certain source that are initiated or corrupted by a Byzantine node. Since there are at least $f + 1$ vertex disjoint paths with no Byzantine nodes from any source p_k to any node p_i and since p_k repeatedly sends messages to all nodes on all possible paths, p_i receives at least $f + 1$ messages from p_k with vertex disjoint paths.

The usage of the FIFO queue and the repeated send operations of p_k ensure that the most recent $f + 1$ messages with vertex disjoint paths in $InformedTopology$ queue are uncorrupted messages. Namely, misleading messages that were present in the initial configuration will be pushed to appear below the new $f + 1$ uncorrupted messages. Thus, each node p_i eventually has the local topology of each correct node (stored in the $Result_i$ array). The opposite is however not correct as local topologies of non-existing nodes may still appear in the result array. For example, $InformedTopology_i$ may include in the first configuration $f + 1$ messages with vertex disjoint paths for a non-existing node.

Since after $ComputeResults$ we know the correct neighborhood of each correct node p_k, we may try to ensure the validity of all messages. For every message that encodes a non-existing source node, there must be a node p_ℓ on the message path, such that p_ℓ is correct and p_ℓ's neighbor is non-existing, this is true since p_i itself is correct. Thus, we may identify these messages and ignore them. Furthermore, no valid messages are ignored because of this validity check.

We also note that, since we assume that the nodes of the system are a subset of P. The size of the queue $InformedTopology$ is bounded. Next, we bound the amount of memory of a node. The details of the correctness and convergence proofs appear in the Appendix and in [7].

Algorithm 1: Topology discovery, code for node p_i

Input: $Neighborhood_i$: The ids of the nodes with which node p_i can communicate directly;
Output: $ConfirmedTopology \subset P \times P$: Discovered topology, which is represent by a directed edge set;
Variable $InformedTopology : Queue$, see Figure 1: topological messages, $\langle node, neighborhood, path\rangle$;
Function: $NodeDisjointPaths(S)$: Test $S = \{\langle node, neighborhood, path\rangle\}$ to encode at least $f + 1$ vertex disjoint paths;
Function: $PathContradictsNeighborhood(k, Neighborhood_k, path)$: Test that there is no node $p_j \in N$ for which there is an edge (p_k, p_j) in the message's visited path, $path \subseteq P \times N$, such that (p_k, p_j) is contradicting with $Neighborhood_k$;

1 **while true do**
2 $\quad Result \leftarrow ComputeResults()$
3 $\quad$ **let** $Result \leftarrow RemoveContradictions(Result)$
4 $\quad RemoveGarbage(Result)$
5 $\quad ConfirmedTopology \leftarrow ConfirmedTopology \cup (\bigcup_{p_k \in P} Result[k])$
6 $\quad$ **foreach** $p_k \in N_i$ **do send**$(i, Neighborhood_i, \emptyset)$ **to** p_k

7 **Upon Receive** $(\langle \ell, Neighborhood_\ell, VisitedPath_\ell\rangle)$ **from** p_j;
begin
8 $\quad Insert(p_\ell, Neighborhood_\ell, VisitedPath_\ell \cup \{j\})$
9 $\quad$ **foreach** $p_k \in N_i$ **do if** $k \notin VisitedPath_\ell$ **then send**$(p_\ell, Neighborhood_\ell, VisitedPath_\ell \cup \{j\})$ **to** p_k

10 **Procedure:** $Insert(k, Neighborhood_k, VisitedPath_k)$;
begin
11 $\quad$ **if** $\exists m = \langle \ell, Neighborhood_\ell, VisitedPath_\ell\rangle \in InformedTopology : (\ell, Neighborhood_\ell, VisitedPath_\ell) = (k, Neighborhood_k, VisitedPath_k)$ **then** $InformedTopology.MoveToHead(m)$
12 $\quad$ **else if** $p_k \in N \wedge Neighborhood_k \subseteq indices(N) \wedge VisitedPath_k \subseteq indices(N)$ **then** $InformedTopology.Insert(\langle k, Neighborhood_k, VisitedPath_k\rangle)$

13 **Function:** $ComputeResults()$;
begin
14 $\quad$ **foreach** $p_k \in P : \langle k, Neighborhood_k, VisitedPath_k\rangle \in InformedTopology$ **do**
15 $\quad\quad$ **let** $(FirstDisjointPathsFound, Message, opinion[]) \leftarrow (false, InformedTopology.Iterator(), [\emptyset])$
16 $\quad\quad$ **while** $Message.hasNext()$ **do**
17 $\quad\quad\quad \langle \ell, Neighborhood_\ell, VisitedPath_\ell\rangle \leftarrow Message.Next()$
18 $\quad\quad\quad$ **if** $\ell = k$ **then** $opinion[Neighborhood_\ell].Insert(\langle \ell, Neighborhood_\ell, VisitedPath_\ell\rangle)$
19 $\quad\quad\quad$ **if** $FirstDisjointPathsFound = false \wedge NodeDisjointPaths(opinion[Neighborhood_\ell])$ **then**
$\quad\quad\quad\quad (Result[k], FirstDisjointPathsFound) \leftarrow (Neighborhood_\ell, \mathbf{true})$
20 $\quad\quad Count[k] \leftarrow opinion[Result[k.SizeOf()]]$
21 $\quad$ **return** $Result$

22 **Function:** $RemoveContradictions(Result)$;
begin
23 $\quad$ **foreach** $\langle r, Neighborhood_r, VisitedPath_r\rangle \in InformedTopology$ **do**
24 $\quad\quad$ **if** $\exists p_k \in P : PathContradictsNeighborhood(p_k, Result[k], VisitedPath_r) = \mathbf{true}$ **then**
25 $\quad\quad\quad$ **if** $Neighborhood_r = Result[r]$ **then** $Count[r] \leftarrow Count[r] - 1$
26 $\quad\quad\quad$ **if** $Count[r] \leq f$ **then** $Result[r] \leftarrow \emptyset$
27 $\quad$ **return** $Result$

28 **Procedure:** $RemoveGarbage(Result)$;
begin
29 $\quad$ **foreach** $p_k \in N$ **do**
30 $\quad\quad$ **foreach** $m = \langle k, Neighborhood_k, VisitedPath_k\rangle \in InformedTopology :$
$\quad\quad \{k\} \cup Neighborhood_k \cup VisitedPath_k \not\subseteq P \vee InformedTopology.IsAfter(m, opinion[k][Result[k]])$ **do**
$\quad\quad InformedTopology.Remove(m)$

Lemma 1 (Bounded memory) *Let $p_i \in C$ be a correct node. At any time, there are at most $n \cdot 2^{2^n}$ messages in $InformedTopologyany_i$, where $n = |P|$ and $\mathcal{O}(|P| \log(|P|))$ is the message size.*

r-neighborhood discovery. Algorithm 1 demonstrates the existence of a deterministic self-stabilizing Byzantine resilient algorithm for topology discovery. Lemma 1 shows that the memory costs are high when the entire system topology is to be discovered. We note that one may consider the task of r-neighborhood

- $Insert(m)$: Insert item m to the head of the queue.
- $Remove(Messagem)$: Remove item m from the queue.
- $Iterator()$: Returns an pointer for iterating over the queue's elements by the order in which they reside in the queue.
- $HasNext()$: Tests whether the Iterator is at the end of the queue.
- $Next()$ Returns the next element to iterate over.
- $SizeOf()$ Returns the number of elements in the calling set.
- $MoveToHead(m)$: Move item m to the head of the queue.
- $IsAfter(m, S)$: Test that item m is after the items $m' \in S$, where S is a set of items in the queue.

Figure 1: $Queue$: general purpose data structure for queuing items, and its operation list.

discovery. Recall that in the r-neighborhood discovery task, it is assumed that every r-neighborhood cannot be partitioned by Byzantine nodes. Therefore, it is sufficient to constrain the maximal path length in line 9. The correctness proof of the algorithm for the r-neighborhood discovery follows similar arguments to the correctness proof of Algorithm 1.

4 End-to-End Delivery

We use the discovered network topology to design a self-stabilizing Byzantine resilient algorithm for the transport layer protocol. Namely, using the repeatedly collected topology information for implementing end-to-end communication between (not necessarily neighboring) nodes. In this context, we face the challenge of finding $f + 1$ correct vertex disjoint paths and the need to propose efficient solutions for different system settings.

The value of $ConfirmedTopology$ is a set of directed edges (p_i, p_j). An undirected edge is approved if both (p_i, p_j) and (p_j, p_i) appear in $ConfirmedTopology$. An edge is said to be suspected, whenever only one edge (in one direction) appears in $ConfirmedTopology$. The sender has to choose $2f + 1$ vertex independent paths to the receiver. If there exists at least one such set of paths then the sender can safely use them to communicate with the receiver (similar to Algorithm 1). However, the collected topology may not include even one such set of $2f + 1$ vertex independent paths. The reason is that f of the paths that should appear in the collected topology may be controlled by Byzantine nodes. Namely, the information about at least one edge in each such path may not arrive to the sender.

We propose three procedures for overcoming this difficulty in different system setting. The first procedure assumes f is a constant. Thus, the sender may apply the following procedure for selecting a set of vertex disjoint paths $Paths$, that contains $f + 1$ correct paths. For each possible choice of f nodes $p_1, p_2, \ldots p_f$ in the system, the sender computes a new graph G' which is the result of removing $p_1, p_2, \ldots p_f$, from G_{out}, the graph defined by the collected topology. The sender now computes a set $\mathcal{P}$ of vertex disjoint paths, where $|\mathcal{P}| = f + 1$, if such a set exists. For each such set $\mathcal{P}$, the sender relays the current message on all paths in $\mathcal{P}$. First we show that this procedure only sends message through a polynomial number of paths. There are $\mathcal{O}(n^f)$ possibilities for choosing f nodes from the system. Thus, $\mathcal{O}(n^f)$ sets of paths are computed, and since f is a constant, this number is polynomial. Moreover, each such set contains at most $f + 1$ paths, because p_i only computes a set $\mathcal{P}$ of size $f + 1$. Thus, in total, the sender sends the message on at most a polynomial number of paths. We now show that this procedure ensures that the message is sent on a sufficient amount of correct paths i.e., $f + 1$. Consider the permutation in which the set of f chosen nodes actually contains the set of Byzantine nodes in the system. Thus G' contains only correct nodes. Furthermore, at least $f+1$ paths that were present in G_{out} are still present in G', since we removed f nodes. Hence, in G', there are at least $f+1$ correct vertex disjoint paths. As stated, the sender chooses a set of paths of size $f + 1$. Each of these paths is correct, and therefore the sender sends the message on at least $f + 1$ correct

vertex disjoint paths as needed.

The second procedure assumes that r and Δ are both constants. The sender sends the message over all possible paths to the receiver. This is feasible only when considering r-neighborhoods, rather than the entire connected component, where the neighborhood radius, r, and the node degree Δ are constants. Next, we present a polynomial solution for the case in which f, r and Δ are not constants, assuming that Byzantine nodes are not directly connected.

The third procedure assumes that Byzantine nodes cannot be immediate neighbors and that all neighbors of a given Byzantine node refer to the Byzantine with the same identifier. Our polynomial cost solution considers the (extended) graph, G_{ext}, that includes all the edges in $confirmedTopology$ and *suspicious edges*, see Definition 2.

Definition 2 (Suspicious edges) *Given three nodes, $p_i, p_j, p_k \in P$, we say that node p_i considers the undirected edge (p_k, p_j) suspicious, if the edge appears as a directed edge in $ConfirmedTopology_i$ for only one direction, e.g., (p_j, p_k).*

The extended graph, G_{ext}, may contain fake edges that Byzantine nodes reports on their existence. Nevertheless, G_{ext} includes all the correct paths of the communication graph, G. Therefore, the $2f + 1$ vertex disjoint paths that exists in G also exists in G_{ext}. These $2f + 1$ paths facilitate our polynomial cost solution.

The sender uses the chosen paths to repeatedly forward the message m that should arrive to the receiver. The sender uses a label to identify the messages. Roughly speaking, the receiver deliver a message received at least $c \cdot n + 1$ consecutive times from $f + 1$ vertex independent paths (according to the path carried in the message). Once the receiver delivers a message to the network layer, the receiver starts to repeatedly send acknowledgments with the label of the delivered message over $2f + 1$ vertex disjoint paths. In addition, the receiver also restarts its counters and the log of received messages upon a message delivery to the network layer. Similarly the sender count acknowledgments to the current label used, when the sender receives at least $c \cdot n + 1$ acknowledgments on a set of $f + 1$ vertex disjoint paths, the sender fetches the next message from the network layer, changes the label and starts to send the new message. We note that starting from an arbitrary configuration, the sender eventually fetches a message from the network layer. This is obvious since if the sender is sending the same message forever, then the receiver counters on $f + 1$ paths must exceed $c \cdot n + 1$. From this point the receiver sends acknowledgments with the correct label forever ensuring that the sender fetches the next message.

The pseudocode of the algorithm appears in Algorithm 2. In every iteration of the infinite loop, p_i fetches a message (line 3). Following the fetch, p_i prepares the label for the next message (line 4). Once the label is ready, p_i starts sending the message over $2f + 1$ vertex disjoint messages which p_i calculates in the procedure $ByzantineFaultToleranceSend(Message)$. When p_i gets enough acknowledgments regarding the current message (see line 5), p_i stops sending the current message and fetches another message.

Upon receiving a message m, node p_i checks in line 7 whether p_i is the destination of the message. If not, p_i forwards the message to the next node on the intended path of the message, not forgetting to update the visited path. If however p_i is the destination of the message, p_i checks the type of the message in line 10. If the type of the message is *Data* then (in line 11) p_i inserts the message payload and label to the part of the data structure associated with the message source, i.e., the sender, and the message visited path. In line 27, node p_i checks whether $2f + 1$ vertex disjoint paths relayed the message at least $capacity \cdot n + 1$ times, where $capacity$ is an upper bound on the number of messages in transit over a communication link. If so, p_i delivers the message to the above layer (line 20), clears the entire data structure and finally sends acknowledgments on $2f + 1$ vertex disjoint paths until a new message is confirmed. Moreover, in line 21 we signal that we are ready to receive a new message. If the type of the message is ACK, we act almost as when the message is of type $Data$. When the condition in line 18 holds, we signal that the message was

confirmed at the receiver by setting $Approved$ to be $true$, in line 18.

Correctness proof. Let us consider three labels, 0, 1, and 2 that are used by the sender in a round robin fashion. Whenever at least $c \cdot n + 1$ identical messages arrive at the receiver with the same label on each of $f + 1$ vertex independent paths, the receiver delivers them, nullify the counters, empty queues and send acknowledges with the label of the delivered message over $2f + 1$ vertex-disjoint paths (cf. line 13). The sender clears counters and queues whenever the sender changes label.

First we prove that the sender fetches infinitely often, by assuming it is not and proving that eventually the receiver sends acknowledgments with the label used by the sender. Hence, the sender must fetch (see Lemma 13). Then in between the second and the fourth fetch of any four successive fetches, where without the loss of generality, the first fetch is with label 0, the second with 1, the third with label 2 and the fourth with 0 the receiver clears its counter and the last fetched message in this sequence that is with label 0 is later delivered.

Following the fetch of each of the first three messages and before the next one, the sender must count $c \cdot n + 1$ acknowledgments with the current label that the sender uses to send, namely with 0, 1 and 2. Since the sender reset the counters when changing the sending label to 1, the receiver must send at least one acknowledgment with label 1 and then with label 2, following the corresponding fetches. Thus, the receiver must clear its counters at least once following the second fetch and before the fourth fetch and then start sending acknowledgments with label 2. After clearing the counters by the receiver and starting sending acknowledgments with label 2 a message with label 0 that is next to be sent, must be delivered and no other message can be counted as arriving at least $c \cdot n + 1$ times through $f + 1$ vertex-disjoint paths. Detailed proof appears in the Appendix and in [7].

Note that the code of Algorithm 2 considers only one possible pair of source and destination. A many-source to many-destination version of this algorithm can simply use a separate instantiation of this algorithm for each pair of source and destination.

5 Extensions and Conclusions

As extension, we suggest to combine the algorithms for r-neighborhood network discovery and the end-to-end capabilities in order to allow the use of end-to-end message delivery within the r-neighborhoods. These two algorithms can be used by the nodes, under reasonable node density assumptions, for discovering their r-neighborhoods and then extending the scope of their end-to-end capabilities beyond their r-neighborhood, as we sketch next. We instruct further remote nodes to relay topology information, and in this way collect information on remote neighborhoods. One can consider an algorithm for studying specific remote neighborhood that are defined, for example, by their geographic region, assuming the usage of GPS inputs; a specific direction and distance from the topology exploring node defines the exploration goal. The algorithm nominates $2f + 1$ nodes in the specific direction to return further information towards the desired direction. The sender uses end-to-end communication to the current $2f + 1$ nodes in the *front* of the current exploration, asking them for their r-neighborhood, chooses a new set of $2f + 1$ nodes for forming a new front. It then instructs each of the current nodes in the current front to communicate with each node in the chosen new front, to nominate the new front nodes to form the exploration front.

To ensure stabilization, this interactive process of remote information collection should never stop. Whenever the current collection process investigates beyond the closest r-neighborhood, we concurrently start a new collection process in a pipeline fashion. The output is the result of the last finalized collection process. Thus, having a correct output after the first time a complete topology investigation is finalized.

In this work we presented two deterministic, self-stabilizing Byzantine-resilience algorithms for topology discovery and end-to-end message delivery. We have also considered an algorithm for discovering r-neighborhood in polynomial time, communication and space. Lastly, we mentioned a possible extension for exploring and communicating with remote r-neighborhoods using polynomial resources as well.

The obtained end-to-end capabilities can be used for communicating the public keys of parties and establish private keys, in spite of f corrupted nodes that may try to conduct man-in-the-middle attacks, an attack that the classical Public key infrastructure (PKI) does not cope with. Once private keys are established encrypted messages can be forwarded over any specific $f + 1$ node independent paths, one of which must be Byzantine free. The Byzantine free path will forward the encrypted message to the receiver while all corrupted messages will be discarded. Since our system should be self-stabilizing, the common private secret should be re-established periodically.

References

[1] F. M. Al-Azemi and M. H. Karaata. Brief announcement: A stabilizing algorithm for finding two edge-disjoint paths in arbitrary graphs. In X. Défago, F. Petit, and V. Villain, editors, *SSS*, volume 6976 of *Lecture Notes in Computer Science*, pages 433–434. Springer, 2011.

[2] L. Alvisi, J. Doumen, R. Guerraoui, B. Koldehofe, H. C. Li, R. van Renesse, and G. Trédan. How robust are gossip-based communication protocols? *Operating Systems Review*, 41(5):14–18, 2007.

[3] B. Awerbuch and M. Sipser. Dynamic networks are as fast as static networks (preliminary version). In *FOCS*, pages 206–220. IEEE Computer Society, 1988.

[4] M. Burmester, T. V. Le, and A. Yasinsac. Adaptive gossip protocols: Managing security and redundancy in dense ad hoc networks. *Ad Hoc Networks*, 5(3):313–323, 2007.

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

[6] S. Dolev, S. Gilbert, R. Guerraoui, and C. C. Newport. Gossiping in a multi-channel radio network. In A. Pelc, editor, *DISC*, volume 4731 of *Lecture Notes in Computer Science*, pages 208–222. Springer, 2007.

[7] S. Dolev, O. Liba, and E. M. Schiller. Self-stabilizing byzantine resilient topology discovery and message delivery. Technical Report 2012:01, Chalmers University of Technology, 2012. ISSN 1652-926X.

[8] V. Drabkin, R. Friedman, and M. Segal. Efficient byzantine broadcast in wireless ad-hoc networks. In *DSN*, pages 160–169. IEEE Computer Society, 2005.

[9] S. Dubois, T. Masuzawa, and S. Tixeuil. Maximum metric spanning tree made byzantine tolerant. In D. Peleg, editor, *DISC*, volume 6950 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2011.

[10] Y. Fernandess and D. Malkhi. On spreading recommendations via social gossip. In F. Meyer auf der Heide and N. Shavit, editors, *SPAA*, pages 91–97. ACM, 2008.

[11] R. Hadid and M. H. Karaata. An adaptive stabilizing algorithm for finding all disjoint paths in anonymous mesh networks. *Computer Communications*, 32(5):858–866, 2009.

[12] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. Bar gossip. In *OSDI*, pages 191–204. USENIX Association, 2006.

[13] N. Lynch. *Distributed Computing*. Morgan Kaufmann Publishers, 1996.

[14] Y. Minsky and F. B. Schneider. Tolerating malicious gossip. *Distributed Computing*, 16(1):49–68, 2003.

[15] M. Nesterenko and S. Tixeuil. Discovering network topology in the presence of byzantine faults. *IEEE Trans. Parallel Distrib. Syst.*, 20(12):1777–1789, 2009.

[16] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks (extended abstract). In L. Logrippo, editor, *PODC*, pages 51–59. ACM, 1991.

[17] M. Paquette and A. Pelc. Fast broadcasting with byzantine faults. *Int. J. Found. Comput. Sci.*, 17(6):1423–1440, 2006.

Algorithm 2: Self-stabilizing Byzantine resilient end-to-end delivery, code for node p_i.

Interface: $FetchMessage()$: Get a new message from the upper layer. We denote by $InputMessageQueue$ the unbounded queue of all messages that are to be delivered to the destination;
Interface: $DeliverMessage(Source, Message)$: Deliver an arriving message to the higher layer. We denote by $OutputMessageQueue$ the unbounded queue of all messages that are to be delivered to the higher layer. We assume that it always contains at least the last message inserted to it;
Input: $ConfirmedTopology$: The discovered topology, which is represent by a set of directed edges included in $P \times P$, see Algorithm 1;
Data Structure: Transport layer messages: $\langle Source, Destination, VisitedPath, IntentedPath, ARQLabel, Type, Payload\rangle$, where $Source$ is the sending node, $Destination$ is the target node, $VisitedPath$ is the actual relay path, $IntentedPath$ is the planned relay path, $ARQLabel$ is the sequence number of the stop-and-wait ARQ protocol, and $Type \in \{Data, ACK\}$ message type, where DATA and ACK are constant;
Field: $Payload$: the message data;
Variable $Message$: the current message being sent;
Variable $ReceivedMessages[j][Path]$: queue of p_j's messages that were relayed over path $Path$ (see Figure 1);
Variable $Confirmations[j][Path]$: queue of p_j's message acknowledgments that were relayed over path $Path$ (see Figure 1);
Variable $label$: the current sequence number of the stop-and-wait ARQ protocol;
Variable $Approved$: A Boolean variable indicating whether $Message$ was accepted at the destination;
Function: $NodeDisjointPaths(S)$: Test S, a set of paths, to encode at least $f+1$ vertex disjoint paths;
Function: $FloodedPath(MessageQueue, m)$: Test whether m is encoded by the first $capacity \cdot n + 1$ messages in $MessageQueue$, where $capacity$ is an upper bound on the number of messages in transit over a communication link.;
Function: $SuspiciousEdges()$: Get the set of suspicious edges;
Function: $getDisjointPaths(Topology, Source, Destination)$: Get a set of $f+1$ vertex disjoint paths between source and destination in the graph induced by Topology.;
Function: $ClearQueue(Source)$: Delete all data in $ReceivedMessages[Source][*]$;
Function: $ClearAckQueue(Destination)$: Delete all data in $Confirmations[Destination][*]$;

```
1  while true do
2      ClearAckQueue(Message.Destination)
3      Message ← FetchMessage()
4      label ← label + 1 modulo 3
5      while Approved = false do ByzantineFaultToleranceSend(Message)
6  Upon Receive (msg) From p_j;
   begin
7      if msg.Destination ≠ i then
8          msg.VisitedPath ← msg.VisitedPath ∪ {j}
9          send(msg)
10     else if msg.Type = Data then
11         ReceivedMessages[msg.Source][msg.VisitedPath].insert(⟨msg.Payload, msg.ARQLabel⟩)
12         if ∃m ∈ ReceivedMessages[msg.Source][*] : Paths = {Path :
           FloodedPath(ReceivedMessages[msg.Source][Path], m)} ∧ NodeDisjointPaths(Paths) ∧
           msg.source = m.source then
13             Confirm(msg.Source, m.ARQLabel, m.Payload)
14             NewMesssage = true
15     else if msg.Type = ACK then
16         if label = msg.ARQLabel then
               Confirmations[msg.Source][msg.VisitedPath].insert(⟨msg.Payload, msg.ARQLabel⟩)
17         let Paths ← {Path : FloodedPath(Confirmations[msg.Source][Path], ⟨msg.Payload, msg.ARQLabel⟩)}
18         if NodeDisjointPaths(Paths) then Approved = true
19 Function: Confirm(Source, ARQLabel, Payload);
   begin
20     if CurrentLabel ≠ ARQLabel then DeliverMessage(Source, Payload)
21     (CurrLbl, NewMessage) ← (ARQLbl, false)
22     ClearQueue(Source)
23     while NewMessage = false do ByzantineFaultToleranceSend(⟨Source, ARQLabel, ACK, Payload⟩)
24 Function: ByzantineFaultToleranceSend(Destination, ARQLabel, Type, Payload);
   begin
25     let Paths ← getDisjointPaths(ConfirmedTopology ∪ SuspiciousEdges(), i, Destination)
26     foreach Path ∈ Paths do send(⟨i, Destination, ∅, Path, ARQLabel, Type, Payload⟩) to first(Path)
```

A Correctness of Algorithm 1

Lemma 1 (Bounded memory) *Let $p_i \in C$ be a correct node. At any time, there are at most $n \cdot 2^{2n}$ messages in $InformedTopologyany_i$, where $n = |P|$ and $\mathcal{O}(|P|\log(|P|))$ is the message size.*

Proof. The queue $InformedTopologyany_i$, is made up of messages in the form $\langle node, neighborhood, visitedpath\rangle$. All nodes that appear in the message, i.e., in the first, second or third entry of the tuple are in N. The first entry, i.e. the node name is one of n possibilities. The second and third entries are subsets of N. Thus each of them has 2^n possibilities. In total there can be at most $2^n \cdot 2^n \cdot n$ messages in every $InformedTopologyany_i$. ■

Definition 3 specifies the requirements of the network topology discovery task. Definition 4 considers correct paths and Definition 5 considers uncorrupted graph topology messages.

Definition 3 (Legal output) *Given correct node $p_i \in C$, we say that p_i's output is* legal*, if it encodes graph $G_{output} = (V_{out}, E_{out})$: (1) $C \subseteq V_{out} \subseteq C \cup B \subseteq N$, and (2) $(E \cap (C \times C)) \subseteq E_{out} \subseteq (E \cap (C \times C)) \cup (B \times (C \cup B)) \subseteq N \times N$.*

Definition 4 (Correct path) *We say $path \subseteq N$ is a* correct *one if all its nodes are correct, i.e., $path \subseteq C$.*

Definition 5 (Valid message) *In Algorithm 1, we refer to a message $m = \langle k, Neighborhood_k, VisitedPath_k\rangle$ as a* valid *message when: (1) $p_k \in C$ and $VisitedPath_k$ encodes a correct path in the communication graph, G, that starts in p_k, and (2) $Neighborhood_k = indices(N_k)$.*

Lemma 2 shows that eventually correct paths do not relay non valid messages. Namely, invalid messages can only exist as the result of: (1) Byzantine interventions that corrupt messages, or (2) transient faults, which occur only prior to the arbitrary starting configuration considered. [2]

Lemma 2 (Eventually valid messages) *Let R be a fair execution of Algorithm 1 that starts in an arbitrary configuration. Within $\mathcal{O}(|C \cup B|)$ asynchronous rounds, the system reaches a configuration after which only valid messages are relayed on correct paths.*

Proof. Let $c \in R$ be the starting configuration. Suppose that c includes an invalid message, $m = \langle \ell, Neighborhood_\ell, VisitedPath_\ell\rangle$, in transit between correct nodes. The lemma is obviously correct for the case that m is relayed by Byzantine nodes during the first $\mathcal{O}(|C \cup B|)$ asynchronous rounds of R. Therefore, we consider only the correct paths, $path$, over which m is relayed during the first $\mathcal{O}(|C \cup B|)$ asynchronous rounds of R. We show that, within $\mathcal{O}(|C \cup B|)$ asynchronous rounds, no correct node in $path$ relays m.

Let $p_j, p_i \in path$ be correct neighbors on the correct path. Suppose that in c, message m is in transit from p_j to p_i. Upon the arrival of message m to p_i (line 7), p_i sends $m_i = \langle \ell, Neighborhood_\ell, VisitedPath_\ell \cup \{j\}\rangle$ to any neighbor $p_k \in path$ on the path for which $p_k \in N_i \wedge k \notin VisitedPath_\ell$, see line 9.

Node p_i adds p_j's identifier to m's visited path $VisitedPath_\ell$, see line 9. The same argument holds for any correct neighbors, $p'_j, p'_j \in path$ when p_j sends message m'_j to the next node in $path$, node p'_i.

[2] This is a common way to argue about self-stabilization, we consider executions that start in an arbitrary configuration that follows the last transient fault, recalling that if additional transient faults occur a new arbitrary configuration is reached from which automatic convergence starts.

Therefore, within $|path \setminus VisitedPath_\ell|$ asynchronous rounds, it holds that $N'_i \cap (path \setminus VisitedPath_\ell) = \{p'_j, p'_i\}$.

Note that p'_i makes sure that $VisitedPath'_\ell$ does not encode loops, i.e., $p_k \notin VisitedPath'_\ell$, see line 9. Therefore, node p'_i does not relay message m' to p_k. ■

Definition 6 considers queues that their recent valid messages encode at least $f+1$ vertex disjoint paths. Moreover, the invalid ones encode at most f such paths.

Definition 6 (Valid queue) *Let $p_i, p_k \in C$ be two correct nodes. We say that p_i's queue, $InformedTopology_i$, is* valid (with respect to p_k) *whenever there is a prefix, $ValidInformation_{i,k}$, of messages m_k in the queue $InformedTopology_i$, such that: (1) there is a subset, $Valid = \{m_\ell = \langle k, Neighborhood_k, VisitedPath_\ell\rangle : m_\ell \text{ is valid}\} \subseteq ValidInformation_{i,k}$, for which the set $\{VisitedPath_\ell\}$ encodes at least $f+1$ vertex disjoint paths, and (2) the set, $Invalid = \{m_\ell = \langle k, Neighborhood_k, VisitedPath_\ell\rangle : m_\ell \text{ is invalid}\} \subseteq ValidInformation_{i,k}$, for which the set $\{VisitedPath_\ell\}$ encodes at most f vertex disjoint paths.*

Claim 3 shows that, within $\mathcal{O}(|C|)$ asynchronous rounds, correct paths propagate valid messages.

Claim 3 *Let $path \subseteq C$ be a correct path from p_i to p_k. Suppose that $m_i = \langle i, N_i, \emptyset\rangle$ is a (valid) message that p_i sends, see line 6. Within $\mathcal{O}(|path|)$ asynchronous rounds, message m_i is relayed on $path$, and arrives at p_k as $m'_i = \langle i, N_i, path\rangle$. Namely, $path$ is m'_i's visited path.*

Proof. Let $c \in R$ be the first configuration that follows the start of m_i's propagation in $path$. I.e., c is the configuration that immediately follows the step in which node p_i sends m_i by executing line 6. Let $p_r, p_j \in path$ be two correct neighbors on the path. Without the loss of generality, suppose that node p_i sends message m_i directly to node p_r, i.e., in c, node p_r is just about to receive m_i. The proof arguments hold also when assuming that p_j sends message $m_j = \langle i, N_i, \{r\}\rangle$ to the next node in $path$. Thus, generality is not lost.

We show that, within one asynchronous round, p_r sends m_r to p_j. Upon the arrival of message m_i to p_r (line 7), node p_r sends the message m_r to any neighbor, such as p_j, for which $p_j \in N_r \wedge r \notin VisitedPath_i = \emptyset$, see line 9. Since the same argument holds when p_j sends m_j to the next node in $path$, we have that within $|path|$ asynchronous rounds, m'_i is delivered to node p_k.

□

Lemma 4 shows that queues get to become valid.

Lemma 4 (Eventually valid queues) *Let R be a fair execution of Algorithm 1 that starts in an arbitrary configuration and $p_i, p_k \in C$ be any pair of correct nodes. The system reaches a configuration in which the queue, $InformedTopology_i$, is valid (with respect to p_k), within $\mathcal{O}(|C \cup B|)$ asynchronous rounds.*

Proof. Let $c \in R$ be a configuration achieved in Lemma 2 within $\mathcal{O}(|C \cup B|)$ asynchronous rounds. We show that within $\mathcal{O}(|C \cup B|)$ asynchronous rounds after c, the system reaches a configuration in which $InformedTopology_i$, is *valid (with respect to p_k)*, see Definition 6.

In configuration c, all messages in transit on correct paths are valid, see Lemma 2. Thus, the only messages entering $InformedTopology_i$ are either valid or have passed through Byzantine nodes. Denote $m_{barrier}$ to be the top message the queue $InformedTopology_i$. Moreover, $ValidInformation_{i,k}$ includes all the messages in $InformedTopology_i$, that are between the queue's head and $m_{barrier}$.

We show that condition (1) of Definition 6 holds. There are $2f+1$ vertex disjoint paths between p_i and p_k. At most f nodes are Byzantine and thus, there are at least $f+1$ vertex disjoint paths between p_i and p_k that are correct. By Claim 3 within $\mathcal{O}(|C|)$ asynchronous rounds, a valid message, m_k, is received on all

$f+1$ (correct) vertex disjoint paths. Message m_k is inserted to $InformedTopology_i$ after configuration c. Therefore, m_k is in front of $m_{barrier}$. Hence, the set $Valid = \{m_\ell = \langle \ell, Neighborhood_\ell, VisitedPath_\ell \rangle : m_\ell \text{ is valid}\} \subseteq ValidInformation_{i,k}$ contains at least $f+1$ valid messages whose respective visited paths, $VisitedPath_\ell$, are vertex disjoint.

We show that condition (2) of Definition 6 holds. Any invalid messages, m_k, that is sent after configuration c, must go through a Byzantine node, see Lemma 2.

Claim 5 *Suppose that message m is relayed through a Byzantine node after configuration c, then in any following configuration, while m is still in transit, there is a Byzantine node in the visited path.*

Proof. Observe the first correct node p_k after the last Byzantine node b on m's path. p_k is correct, thus it inserts b to the visited path. b is the last on the path and so the visited path must contain it until end of transit or passing through a different Byzantine. □

Each such Byzantine node is recorded in the message path, see Claim 5. Since there are at most f Byzantine nodes, there could be at most f such messages with vertex disjoint paths. This completes the proof condition (2) and the lemma. ■

Lemma 7 shows that correct information gets confirmed, and requires Definition 7.

Definition 7 (Message confirmation) *We say that message $m_i = \langle k, Neighborhood_k, VisitedPath_{k_i} \rangle$ is* confirmed (by node p_i) *when $Neighborhood_k \subseteq ConfirmedTopology_i$.*

Lemma 6 (Eventually confirmed messages) *Let R be a fair execution of Algorithm 1 that starts in an arbitrary configuration and $p_i, p_k \in C$ be any pair of correct nodes. Within $\mathcal{O}(|C \cup B|)$ asynchronous rounds, the system reaches a configuration after which the fact that message $m_i = \langle k, Neighborhood_k, VisitedPath_{k_i} \rangle$ is confirmed, implies that $Neighborhood_k = indices(N_\ell)$.*

Proof. Let $c \in R$ be the first configuration in which $InformedTopology_i$ is a valid queue and node p_i completes a full iteration of the do forever loop that starts in line 1. By Lemma 4, the system reaches c within $\mathcal{O}(|C \cup B|)$ asynchronous rounds.

We how that in configuration c, the array $Result_i$ satisfies that $Result_i[k] = indices(N_\ell)$. We go through the computation of $Result$ in lines 2 to 4.

- $ComputeResults()$, line 2. Let $Res_i[k] = indices(N'_\ell)$ be $ComputeResults()$'s return value with respect to node p_k. We show that $Res_i[k] = indices(N_\ell)$. Moreover, we show that the neighborhood that will be found will be that which is represented in $Valid = \{m_\ell = \langle k, Neighborhood_k, VisitedPath_\ell \rangle : m_\ell \text{ is valid}\} \subseteq ValidInformation_{i,k}$.

We recall that the set $\{VisitedPath_\ell\}$ encodes at least $f+1$ disjoint paths. Also in the prefix $ValidInformation_{i,k}$ one can not find $f+1$ invalid messages with vertex disjoint messages; See Definition 6.

The function must choose the message containing the neighborhood $Neighborhood_k$. Otherwise, we have chosen a different neighborhood for k, say $Neighborhood'_k \neq Neighborhood_k = indices(N_k)$. That is, at the time of checking line 19 with neighborhood $Neighborhood_\ell = Neighborhood'_k$, there were at least $f+1$ vertex disjoint paths in $opinion[Neighborhood_\ell]$. This is in contradiction to condition (2) of Definition 6. Moreover in line 20, it holds $Count[k] > f+1$, since at least all the correct paths were counted.

- $RemoveContradictions()$, line 3. Let $Res_i = ComputeResults()$ and $ResRemoveContradictions_i = RemoveContradictions(Res_i)$ (line 3). We show that $ResRemoveContradictions_i[r] = indices(N_r)$. The function $RemoveContradictions()$ modifies

$Res_i[r]$ only in line 26 by nullifying it whenever $Count[r] \leq f$. We demonstrate that, for any correct path $VisitedPath_k$, there exists no p_ℓ for which $PathContradictsNeighborhood(p_\ell, Res_i[\ell], VisitedPath_k) = \mathbf{true}$, which is the condition in line 24.

We explain that there is no node p_ℓ and a contradicting edge (p_j, p_ℓ) with the set $Res_i[\ell]$. By the assumption that $VisitedPath_k$ is correct and that node $p_\ell \in VisitedPath_k$, we have that $p_\ell \in C$ is correct. Thus $Res_i[\ell] = indices(N_\ell)$, see previous item of this claim on $ComputeResults()$. $VisitedPath_k$ is correct, and therefore (p_j, p_ℓ) must be in $VisitedPath_k$.

- $RemoveGarbage()$, line 4. This procedure does not modify $Res_i = RemoveContradictions(ComputeResults())$. We have shown that $Result_i[k] = indices(N_k)$. Thus, only the correct neighborhood is confirmed for every correct node p_k. ■

Lemma 7 shows that eventually there are no fake nodes.

Lemma 7 (Eventually no fake nodes) *Let R be a fair execution of Algorithm 1 that starts in an arbitrary configuration, $p_j \in N$ be any node, and $p_\ell \in P \setminus (C \cup B)$ be a node that is not included in the communication graph, G. Within $\mathcal{O}(|C \cup B|)$ asynchronous rounds, the system reaches a configuration after which $(p_j, p_\ell) \notin ConfirmedTopology_i$*

Proof. Let $c \in R$ be the configuration reached within $\mathcal{O}(|C \cup B|)$ asynchronous rounds according to Lemma 6. For any correct node, $p_i \in C$, we show that in c, the execution of $RemoveContradictions()$ results in $Count_i[\ell] \leq f$ and nullifies $Result_i[\ell]$.

We start by showing that for every path p that relays a message which encodes the set $Result_i[\ell]$, and does not contain Byzantine nodes, a contradiction is found in $RemoveContradictions()$. Namely, the if conditions of line 24 holds.

Note that, p may not be a correct path even though it contains no Byzantine nodes. For example p may contain nodes p_z that are not even in the communication graph, i.e., $p_z \in P \setminus (C \cup B)$.

Let $p_r \in C \cup B$ be the first correct node on path p. Such a node exists, because p_i is correct and on the path p. Since p_r is correct, after the execution of $ComputeResults()$, we have that p_r's neighborhood, N_r, is encoded in $Result_i[r]$, see Lemma 6.

Denote the last edge in the path (p_r, p_s), where $p_s \in P \setminus (C \cup B)$. Note that node p_s is not a node in the system and since $Result_i[r]$ encodes N_r's neighborhood, we have that $p_s \notin Result_i[r]$. Thus, the edge (p_r, p_s) is contradicting with the set $Result_i[r]$. Namely, by the condition in line 24, we have that line 25 must decrease $Count[\ell]$.

We note that immediately before the function $RemoveContradictions()$ returns, the integer $Count[\ell]$ may count only incorrect paths, which contain at least one Byzantine node. Since there are at most f Byzantine nodes, $Count[\ell] \leq f$ as needed. ■

Theorem 8 demonstrates the self-stabilization properties.

Theorem 8 (Self-stabilization) *Let R be a fair execution of Algorithm 1 that starts in an arbitrary configuration and $p_i \in C$ be a correct node. Within $\mathcal{O}(|C \cup B|)$ asynchronous rounds, the system reaches a safe configuration after which p_i's output is always legal, see Definition 3.*

Proof. The systems reaches configuration $c \in R$ of Lemma 6 within $\mathcal{O}(|C \cup B|)$ asynchronous rounds. We show that c is a safe configuration by showing that the output is legal, we must show that $ConfirmedTopology_i$ encodes a graph $G_{output} = (V_{out}, E_{out})$, such that: (1) $C \subseteq V_{out}$, (2) $(E \cap (C \times C)) \subseteq E_{out}$, (3) $V_{out} \subseteq C \cup B \subseteq N$, and (4) $E_{out} \subseteq (E \cap (C \times C)) \cup (B \times (C \cup B)) \subseteq P \times N$.

For every correct node $p_k \in C$, we have that N_k is confirmed in c, see Lemma 6. Thus, $p_k \in V_{out}$ and condition (1) holds.

Let (p_j, p_k) be an edge in the communication graph between two correct nodes, we show $(p_j, p_k) \in E_{out}$. Since p_j is correct, it is inserted to $ConfirmedTopology_i$, see Lemma 6. Thus, $(p_j, p_k) \in edges(N_j) \wedge edges(N_j) \subseteq ConfirmedTopology_i$ in c, thus condition (2) holds as well.

There is no $p_\ell \in P \setminus (C \cup B)$ and node $p_j \in N$, such that $(p_\ell p_j) \in ConfirmedTopology_i$, see Lemma 7. Thus, $V_{out} \subseteq C \cup B \subseteq N$ and $E_{out} \subseteq (E \cap (C \times C)) \cup (B \times (C \cup B)) \subseteq P \times N$. I.e., conditions (3) and (4) hold in c. ■

B Correctness of Algorithm 2

Lemma 9 shows that senders and receivers can eventually find at least $2f + 1$ vertex-disjoint paths between them. Note that at least $f + 1$ of them are correct.

Lemma 9 *Let R be a fair execution of Algorithm 2 that starts in an arbitrary configuration and $p_s, p_r \in C$ a pair of correct nodes (sender and receiver). Within $\mathcal{O}(|C \cup B|)$ asynchronous rounds the system reaches a configuration in which the set $ConfirmedTopology \cup SuspiciousEdges$ encodes a set of $2f + 1$ vertex disjoint paths from p_s to p_r and at least $f + 1$ of them are correct.*

Proof. Let c be a safe configuration with respect to Algorithm 1. Let $Paths = getDisjointPaths(ConfirmedTopology \cup SuspiciousEdges(), i, Destination)$ be a set of disjoint paths in c, as in line 25, where $i \in \{s, r\}$. We first show that $\mid Paths \mid \geq 2f + 1$ before showing that at least $f + 1$ of them are correct.

We consider the graph $G' = (N, E_{G'})$, which is computed from $ConfirmedTopology$ and the suspicious edges in c. We demonstrate that G' contains the real communication graph, G. Let $e = (p_j, p_k) \in E_{G'}$. When p_j and p_k are both correct, $e \in G'$ since c is safe. When p_j is correct and p_k is Byzantine, we must consider the cases in which p_k reports, and does not report, e as part of its local neighborhood. Namely, either $e \in ConfirmedTopology$, or $e \in SuspiciousEdges()$, because p_k does not report about e, but p_i does. Since $G \subseteq G'$, G' must contain $2f + 1$ vertex disjoint paths between any p_s and p_r, because G does. Thus $\mid Paths \mid \geq 2f + 1$.

Moreover, the same arguments implies that there may be at most f incorrect paths, which contain at least one Byzantine node. Hence, there are at least $f + 1$ correct nodes in $Paths$. ■

Definitions 8, 9 and 10 are needed for lemmas 11, 12 and 13.

Definition 8 (Confirmation) *Given configuration c, we say that message m is* confirmed (by the receiver) *when $m \in OutputMessageQueue$.*

Definition 9 (Approve) *Given fair execution, R, of Algorithm 2, we say that message $m = \langle Source, Destination, VisitedPath, IntentedPath, ARQLabel, DATA, Payload \rangle$ is* being approved (by the sender p_{Source}) *during the first atomic step, a_{sender}, in which the sender executes line 18, where $Source = sender$ $ARQLabel = msg_{sender}.ARQLabel$ and $Payload = msg_{sender}.Payload$, see line 17. Denote by $c_{approved}$ the configuration that immediately follows a_{sender}. Given configuration c that appears after $c_{approved}$ in R, we say that message m* is approved (by the sender) *in configuration c.*

Definition 10 (Clear-sender-receiver) *Given configuration c, we say that the sender is* clear (with respect to the receiver)*, if the queue $Confirmations[receiver] = \emptyset$ in c. Moreover, the receiver is* clear (with respect to the sender) *, if the queue $ReceivedMessages[sender] = \emptyset$ in c.*

Claim 10 shows that a message that is relayed on a correct path is received at the destination within $\mathcal{O}(|C \cup B|)$ asynchronous rounds. Moreover, the destination receives the message with correct visiting set.

Claim 10 *Let R be a fair execution of Algorithm 2 that starts in a safe configuration, c, with respect to Algorithm 1. Let $p_{source}, p_{dest} \in C$ be pair of correct nodes. Let c_{send} be the configuration immediately following a step in which p_{source} sends message Msg on a correct path $Path = p_{source}, p_1, p_2, \ldots p_{dest}$ from source, p_{source}, to destination, p_{dest}. Within $\mathcal{O}(|C \cup B|)$ asynchronous rounds, p_{dest} receives Msg with a visiting set containing all nodes on $Path$ except p_{dest}.*

Proof. Upon the arrival of message m to p_k (line 6), node p_i asserts that he is not the destination, p_{dest}, (line 7). Immediately after, p_i sends the message m to the next neighbor, p_{i+1}, see line 9. Since the same argument holds when p_j sends m to the next node in $path$, we have that within $|Path|$ asynchronous rounds, m is delivered to node p_{dest}. ■

Claim 11 says that when the sender repeatedly sends message Msg, for a duration of at least $\mathcal{O}(|C \cup B|)$ asynchronous rounds, the receiver eventually confirms message Msg.

Claim 11 *Let R be a fair execution of Algorithm 2 that starts in a safe configuration, c, with respect to Algorithm 1. Let $p_s, p_r \in C$ be a pair of correct sending and receiving nodes. Suppose that, for a duration of at least $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds, p_s's steps include only the execution of the function $ByzantineFaultToleranceSend(Msg)$ in the loop of line 5. Within that period, the system reaches configuration $c_{receive}$ in which p_r confirms Msg.*

Proof. Denote c_{send} as the configuration immediately following the first step in which p_s sends message Msg in R, see line 28. Within $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds, the first frame containing Msg arrives at p_r, see Claim 10. Moreover, after another $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds, every correct path relays message Msg at least $\mathcal{O}(capacity \cdot |C \cup B|)$ times. This is correct since every asynchronous round, p_s sends a new frame containing Msg on each of the $2f + 1$ vertex disjoint paths. Moreover, by Claim 10, the last frame sent on all $2f + 1$ paths arrives after another $\mathcal{O}(capacity \cdot |C \cup B|)$.

Assume, in the way of proof by contradiction, that Msg is not confirmed by p_r. This implies that the queues, $ReceivedMessages[p_s][*]$, in p_r containing messages sent from p_s were not cleared at least since c_{send}, see line 22. Thus, p_r contains $capacity \cdot n + 1$ indications of Msg on $f + 1$ vertex disjoint paths. Denote c_{last} as the configuration immediately after the arrival of the $(capacity \cdot n + 1)$-th frame of the $f + 1$'th path to relay $capacity \cdot n + 1$ frames containing Msg. Immediately after c_{last}, p_s must go through line 12, because the conditions in line 12 hold. Thus, a contradiction and Msg is confirmed within $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds. ■

Claim 12 says that when the receiver is sending acknowledgments about a message, that message eventually becomes approved. We note that Claim 12 considers acknowledgments sent from the receiver to the sender, rather than messages sent from the sender to the receiver, as in Claim 11.

Claim 12 *Let R be a fair execution of Algorithm 2 that starts in a safe configuration, c, with respect to Algorithm 1. Let $p_s, p_r \in C$ be a pair of correct sending and receiving nodes. Suppose that, for a duration of at least $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds, p_r's steps include only the execution of the function $ByzantineFaultToleranceSend(Ack)$ in the loop of line 23. That is, p_r is sending acknowledgments on message Msg. Within that period, the system reaches configuration $c_{receive}$ in which p_s approves Msg, see Definition 9.*

Proof. Denote c_{send} as the configuration immediately following the first step in which p_r sends acknowledgment Ack in R, see line 23. Within $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds, the first frame containing Ack arrives at p_s, see Claim 10. Moreover, after another $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds, every correct path relays message Ack at least $\mathcal{O}(capacity \cdot |C \cup B|)$ times. This is correct since every asynchronous round, p_r sends a new frame containing Ack on each of the $2f + 1$ vertex disjoint paths. Moreover, by Claim 10, the last frame sent on all $2f + 1$ paths arrives after another $\mathcal{O}(capacity \cdot |C \cup B|)$.

The queues, $Confirmations[p_r][*]$ are cleared only when a message sent to p_r is approved, see line 2. Since, p_r is acknowledging the current message, Msg, by sending Ack, the only message that can be approved is Msg. This is true since each path, $Path$, may contain at most $capacity \cdot |C \cup B|$ acknowledgments for other messages in the path queues.

Assume, in the way of proof by contradiction, that Msg is not approved by p_s. By the arguments above, p_s's queues, $Confirmations_s[p_r][*]$, which contains p_r's acknowledgments that p_s received, were not cleared at least since c_{send}, see line 2. Thus, p_s contains $capacity \cdot n + 1$ indications of Ack on $f + 1$ vertex disjoint paths. Denote c_{last} as the configuration immediately after the arrival of the $(capacity \cdot n + 1)$-th frame of the $f + 1$'th path to relay $capacity \cdot n + 1$ frames containing Ack. Immediately after c_{last}, p_s must go through line 18, because the conditions in line 18 hold. Thus, a contradiction and Msg is approved within $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds. ■

Lemma 13 shows that the senders repeatedly fetch messages.

Lemma 13 *Let R be a fair execution of Algorithm 2 that starts in a safe configuration, c, with respect to Algorithm 1. Let $p_s, p_r \in C$ be pair of correct sending and receiving nodes. Moreover, c_ℓ is the configuration that immediately follows the ℓ-th time in R in which p_s fetches a message from the input queue. For every ℓ, the system reaches c_ℓ within $\mathcal{O}(\ell \cdot |C \cup B|)$ asynchronous rounds.*

Proof. By the code of Algorithm 2, on every iteration of the do forever loop (lines 2 to 5), a message is fetched in line 3. This do forever loop includes another loop in line 5. We prove the lemma by showing that the loop of line 5 is completed within $\mathcal{O}(|C \cup B|)$ asynchronous rounds.

The proof considers the case in which the sender, p_s, does not wait in line 5 for a long time before considering the case in which p_s does wait. We show that for the latter case, the receiver, p_r, confirms p_s's current message. After confirming the message, the receiver, p_r, begins sending acknowledgments to the sender, p_s. The proof shows that after the acknowledgments are sent, p_s approves the message and fetches a new one. We show this by considering the case in which p_r repeatedly sends acknowledgments for a sufficient amount of time, and a case in which it does not.

Suppose that p_s does not wait in line 5 more than $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds. In this case, p_s starts the infinite loop again within $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds, and fetch a new message, see line 3. Thus, for the case in which p_s does not wait in line 5 more than $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds, the lemma is correct.

Suppose that p_s is executing line 5 and waits for acknowledgments on message Msg for more than $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds. Thus, p_s floods $2f + 1$ vertex-disjoint paths with the message Msg, see Claim 9. Eventually, the receiver, p_r, receives message Msg for $\mathcal{O}(capacity \cdot |C \cup B|)$ times on $f + 1$ vertex-disjoint paths and confirms Msg, see Claim 11. After confirming it, the receiver sends acknowledgments on $2f + 1$ vertex-disjoint paths until confirming a new message Msg_{new}. This is true because the condition in line 23 holds only when a new message is confirmed, see line 14.

Let us consider the case in which, during $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds, message Msg_{new} does not arrive to the receiver. By Claim 12, eventually the sender receives the acknowledgments for

$capacity \cdot n + 1$ times on $f + 1$ vertex disjoint paths. Claim 12 also says that the sender considers the message accepted by the receiver. In line 18, the sender assigns $Approved = \mathbf{true}$. Thus, the condition in line 5 holds and the sender fetches the next message, see line 3. Hence, the system reaches configuration c_{fetch} that immediately follows a step in which the sender, p_s, fetches the next message. Thus, for the case in which, during $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds, message Msg_{new} does not arrive to the receiver, the lemma is correct.

We continue by considering the case in which, during $\mathcal{O}(capacity \cdot |C \cup B|)$ asynchronous rounds, message Msg_{new} does arrive to the receiver. Let c_{conf} be the configuration that immediately follows the step in which p_r confirms Msg. Since the receiver confirms Msg, we have that p_r is clear (with respect to the sender) in configuration c_{conf}, see Definition 10 and line 22.

If Msg_{new} was sent by the sender, it must have been fetched after c, and c_{fetch} is reached when message Msg_{new} is fetched. It may be the case however, that Msg_{new} was not sent by the sender. Message Msg_{new} was confirmed by $2f + 1$ vertex disjoint paths. Since there are at most f Byzantines, at least one of these paths, $Path$, must be correct. Moreover, in c_{conf}, the receiver is clear, thus the $capacity \cdot n + 1$ that p_r counts in $ReceivedMessages[p_s][*]$ have all been received after configuration c_{conf}. Note that the sender sends at least one of these messages, because at most $capacity \cdot n$ messages could be in the edges of $Path$ at any given configuration. Thus the sender sends Msg_{new}, which p_s fetches immediately before c_{fetch}. I.e., the system reaches c_{fetch}. ■

Theorem 8 says that, starting from that fourth (or even the third) message that the sender fetches, the receiver confirms the sender's messages. The proof of Theorem 8 is based on Lemma 14, which says that, in every sequence of four messages that the sender is fetching, the receiver confirms the fourth (or even the third) message.

Lemma 14 *Let R be a fair execution of Algorithm 2 that starts in a safe configuration, c_{start}, with respect to Algorithm 1. Let c_h be a configuration that immediately follows the h-th step in which the sender fetches the h-th input queue message, m_h. Within $\mathcal{O}(|C \cup B|)$ asynchronous rounds, the receiver confirms message m_4.*

Proof.

Claim 15 *In c_2, the sender is clear (with respect to the receiver), see Definition 10.*

Proof. By definition, c_2 immediately follows atomic step a_2, in which, after clearing the confirmation queue in line 2, the sender fetches message m_2 and sends it. □

Claim 16 *Between the configurations c_3 and c_4, there is a configuration $c_{receiver-clear}$ in which the receiver is clear (with respect to the sender).*

Proof. Suppose, without the loss of generality, that immediately after $c_{sender-clear}$, the sender is waiting for a message with label 1. By lemma 13, the sender eventually fetches the next message. The sender can only fetch a new message once $Approved$ is true, see line 5. Moreover, $Approved$ is only set to $true$ once the queue $Confirmations[receiver][*]$ contains $2f + 1$ flooded paths, see line 18. Thus, the sender counts $2f + 1$ vertex disjoint paths that relayed acknowledgments with label 1. Moreover, the sender is clear in $c_{sender-clear}$. Hence, configuration $c_{sender-clear}$ contains no message in $Confirmations[receiver][*]$. Starting from $c_{sender-clear}$, the sender receives $capacity \cdot n + 1$ acknowledgments on $2f + 1$ vertex disjoint paths for the current message with label 1. Note that at least one of these $2f + 1$ paths, $Path$, is correct,

because there are f Byzantine. Since $|Path| \leq n$ and each edge on $Path$ may contain at most $capacity$ messages, we have that at least one of the acknowledgments that includes $Path$ as its visiting path, is sent by the receiver between $c_{sender-clear}$ and configuration $c_{receiver-send} \in R$. We show that $c_{receiver-send} = c_{receiver-clear}$.

This means that after $c_{sender-clear}$, the sender clears the confirmations queue, $Confirmations[receiver][*]$, and fetches the next message, assigning it the label 2, see lines 2 through line 5. By similar arguments, we know that the receiver sends at least one acknowledgment with label 2.

To conclude, there is a configuration $c \in R$ in which the receiver is sending acknowledgments with label 1, and then a configuration c' in which the receiver sends acknowledgments with label 2. Moreover, between two consecutive executions of line 23, the receiver has to go through line 22. Thus, the receiver cleared it's message queues, $Confirmations[sender][*]$, immediately before configuration $c_{receiver-clear}$ and $c_{receiver-send} = c_{receiver-clear}$. □

Let us consider configuration $c_{receiver-clear}$ from the end of proof of Claim 16.

The next message to be sent after $c_{receiver-clear}$, is m_4, the message fetched in c_4, with label 0. Between $c_{receiver-clear}$ and c_4, all messages sent by the sender have the label 2. By arguments stated above, the message, m, that is the next message to be confirmed after $c_{receiver-clear}$, must have been sent by the sender at least once since $c_{receiver-clear}$. The sender, sends only messages with label 0 and 2. Moreover, the last message to be confirmed had a label 2. Thus, $CurrentLabel = 2$, see line 21. Any sent message with label 2 is not inserted to the confirmations queue, $Confirmations[sender][*]$ between $c_{receiver-clear}$ and the configuration that immediately follows the next sender's fetch, see line 20. Thus, by line 4, the next message to be confirmed is a message with label 0, which must be m_4. ■

Theorem 8 (Self-stabilization) *Let R be a fair execution of Algorithm 2 that starts in an arbitrary configuration. Within $\mathcal{O}(|C \cup B|)$ asynchronous rounds, the system reaches a safe configuration c after which: (1) the receiver confirms message m in step $a_r^m \in R$, and (2) for every step a_r^m, there is a corresponding step, $a_s^m \in R$, that occurs before a_r^m and in which the sender sends m.*

Proof. Let c be the configuration that Claim 16 denote as c_4, which the system reaches within $\mathcal{O}(|C \cup B|)$ asynchronous rounds, see Lemma 13. Let m_i be the i-th message fetched.

Suppose that $i \geq 4$. Lemma 14 considers the four consecutive messages $m_{i-3}, \ldots m_i$ and says that the receiver confirms message m_i. Thus, condition (1) holds.

Condition (2) follows from arguments similar to the ones used in the proof of Lemma 11. Namely, for the case of $i \geq 5$, message m_{i-1} is confirmed, see lemma 14. Immediately after the receiver confirms m_{i-1}, it clears the queue $ReceivedMessages[sender][*]$, see lines 20 to 22. Thus, there exists a configuration $c_{receiver-clear}$ in which the receiver is clear (with respect to the sender) before c_i, see Definition 10. Moreover, a message is confirmed only if the queue $ReceivedMessages[sender][*]$ contains $2f + 1$ flooded paths, see line 12. These flooded paths implies that in configuration c_i, the queue $ReceivedMessages[sender][*]$ contains $capacity \cdot n + 1$ indications of m_i on $2f + 1$ node disjoint paths. Thus, m_i is confirmed only after a period that follows $c_{receiver-clear}$ and includes its reception at least $capacity \cdot n + 1$ times on each of the $2f + 1$ vertex disjoint paths.

Recall that we assume that there are at most f Byzantine nodes in the system. At least one path, $Path$, of the above $2f + 1$ paths is correct. Moreover, $|Path| \leq n$ and each edge on $Path$ may contain at most $capacity$ messages. Thus, at least one of the $capacity \cdot n + 1$ message that were relayed on the correct path $Path$ was sent by the sender. This completes the correctness proof. ■

A.3.2 Capture effect based communication primitives

"Capture effect based communication primitives". O. Landsiedel, F. Ferrari, and M. Zimmerling, In SenSys'12: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, Toronto, Canada, November 2012. ftp://ftp.tik.ee.ethz.ch/pub/people/marcoz/LFZ2012.pdf

This page is intentionally left blank.

Poster Abstract: Capture Effect Based Communication Primitives

Closing the Loop in Wireless Cyber-Physical Systems

Olaf Landsiedel[1], Federico Ferrari[2], Marco Zimmerling[2]
olafl@chalmers.se, ferrari@tik.ee.ethz.ch, zimmerling@tik.ee.ethz.ch

[1]Computer Science and Engineering, Chalmers University of Technology, Sweden
[2]Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland

1 Motivation and Principle

Wireless control systems consist of sensing and actuating devices that are commonly driven by a central controller. Wireless communication protocols for Cyber-Physical Systems (CPS) match this design by employing a "sense → collect → process → disseminate → actuate" flow [6], where typically different protocols are employed for collecting sensor data and disseminating actuation signals.

In this paper, we depart from this traditional design and introduce CaptureCom. By relying on capture effects and in-network processing, it omits the need for a central controller and for distinct collection and dissemination phases. In CaptureCom, each node transmits its current data (*e.g.*, temperature reading). Upon receiving, nodes integrate (*e.g.*, aggregate) the received data with previously received data and concurrently forward the result. Due to capture effects, neighboring nodes correctly receive one of the concurrently sent packets with high probability. Repeating this process, the network converges to a stable state where all nodes have received the same data (consensus). The impact of our approach is threefold:

1. CaptureCom closes the loop in CPS: data are processed within the network. Upon completion, all nodes have received with high probability the same data as the base for actuation. Thus, it departs from the widespread architecture of collecting information at a central controller for processing and then disseminating the results.
2. CaptureCom exploits spatial diversity in low-power wireless networks: Consecutive, concurrent transmissions spread out across the network allow for data distribution at very low delays and high energy efficiency.
3. Relying solely on concurrent forwarding and capture effects for communication, CaptureCom simplifies the networking stack by obviating the need for link estimation, neighbor discovery, and routing protocols.

2 CaptureCom: Background and Design

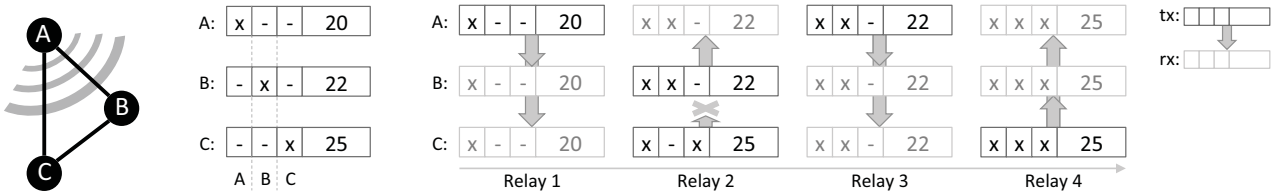
Background and related work. CaptureCom relies on the capture effect, which is a well known phenomenon in wireless communication [4]: If two or more packets are transmitted concurrently, the stronger one is received, assuming it is sufficiently stronger than the contending signals; that is, the signal-to-noise ratio (SNR) is higher than a certain threshold. For instance, the capture effect has been used for fast flooding in sensor networks: nodes concurrently forward received packets to quickly propagate them in the network [5].

Different from CaptureCom, Glossy [3] and LWB [2] rely also on constructively interfering baseband signals. While this allows packets to be received independently of the SNR threshold, it requires that all simultaneously transmitted packets are the same. Thus, Glossy and LWB cannot exploit spatial diversity and in-network aggregation as CaptureCom.

Basic design and sample execution. To introduce the basic design of CaptureCom and demonstrate its benefits, we use a sample application that aims at determining the maximum sensor reading among all nodes in the network (*e.g.*, occupancy or temperature) and disseminating this maximum to all nodes. We note that aggregation is optional in CaptureCom and that its design is independent of the particular aggregation scheme. In the end, all nodes are aware of the maximum temperature reading in the network and can trigger actuation. To illustrate CaptureCom, we assume a network of three nodes (see Fig. 1(a)). Initially, each node is only aware of its own sensor reading (see Fig. 1(b)). A coordination node, node *A* in our example, triggers the communication round by broadcasting its local temperature reading.

The payload of packets consists of two parts: a *bit field*, indicating from which node data have already been integrated, and a *payload field* (see Fig. 1(b)). In this example, the payload is the maximum temperature reading a node has received so far, including its own sensor reading. Upon receiving, a node merges its local bit field with the received one via a bitwise OR and determines the maximum of the local and the received data (see Fig. 1(c)). After a constant delay, which is sufficient for both processing and I/O operations, receiving nodes transmit the resulting data. These nodes transmit concurrently, enabling neighboring nodes to receive the stronger of multiple, concurrently transmitted packets due to the capture effect. In this way, CaptureCom triggers a chain reaction, where each relay increases the number of participating nodes, eventually covering the complete network with very high probability. To avoid starvation, a node transmits when the local and received bitfields differ, that is, the node learned something new (see Fig. 1(c)).

SenSys'12, November 6–9, 2012, Toronto, ON, Canada.
ACM 978-1-4503-1169-4/11/12

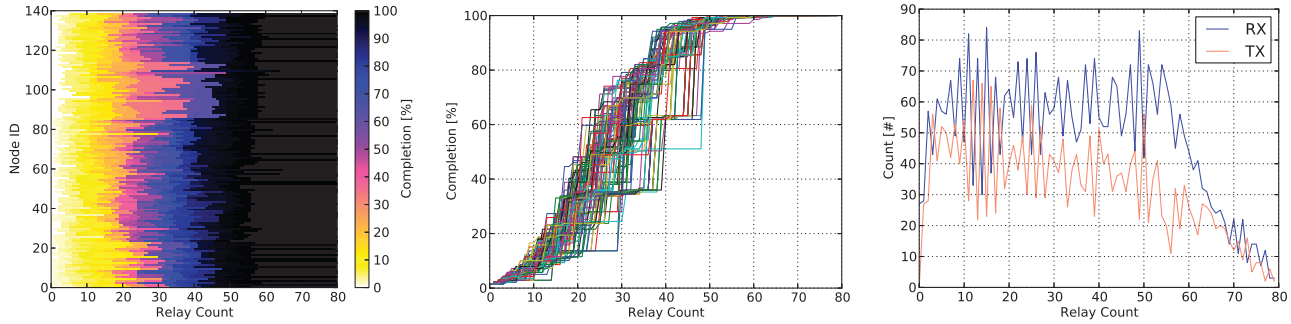


(a) Sample network. Three nodes forming a full mesh.

(b) Initial setting. Each node adds its local reading and marks its bitfield.

(c) Operation of CaptureCom. Node *A* initiates a round by transmitting its sensor reading. Receivers combine this with their own data and transmit the result after a constant delay (to enable the capture effect). In this example, nodes transmit if the received packet contained new information. At relay 2, we assume that node *A* receives the packet sent by node *B*.

Figure 1. Basic idea of CaptureCom. By exploiting the capture effect, CaptureCom performs data collection, aggregation, and dissemination within the network. As a result, it closes the control loop in wireless cyber-physical systems.



(a) CaptureCom converges after about 60 relays. All nodes have received the reading from all others and computed the maximum.

(b) Completion per node over time. Information spreads epidemically in the network.

(c) Successful receptions (RX) and transmissions (TX) per relay. Upon completion at relay count 60, both are slowly fading out.

Figure 2. In-network aggregation of a maximum temperature reading (preliminary results for one representative run).

In our example in Fig. 1(c), we reach completion after four relays. At this point, the temperature readings from all three nodes have been integrated because the corresponding flags in the bitfield are set. Thus, all nodes are aware of the maximum sensor reading. Note that we achieve this without any prior transmission schedules or slot assignments. We merely rely on the capture effect to enable nodes to successfully receive data during concurrent forwarding.

3 Preliminary Evaluation

In this section, we discuss preliminary evaluation results with our prototype implementation of CaptureCom. The experiments are performed on Indriya [1], a testbed of 139 nodes deployed on three floors in an office building.

Fig. 2 shows that CaptureCom computes the maximum temperature reading in a network of 139 nodes within about 60 relays at high reliability, achieving an average delay and radio on-time of about 125 ms per round and node. When requesting a temperature reading, say, once per minute, this would result in an average radio duty cycle of about 0.2%.

4 Summary and Future Work

We argue in this paper that CaptureCom closes the control loop in wireless CPS: it provides efficient collection, processing (in our example aggregation), and dissemination within the network, making it suitable for widespread control applications. Our initial evaluation shows that CaptureCom efficiently exploits spacial diversity. It converges within 125 ms at high reliability, leading to low communication delays and high energy efficiency. Motivated by these promising initial results, we are currently investigating the theoretical and experimental foundations of CaptureCom as a novel primitive for wireless CPS.

Acknowledgments

This work was supported by Nano-Tera, NCCR-MICS under SNSF grant number 5005-67322, and partially by the EC, through project FP7-STREP-288195, KARYON.

5 References

[1] M. Doddavenkatappa, M. C. Chan, and A. Ananda. Indriya: A low-cost, 3D wireless sensor network testbed. In *ICST TridentCom*, 2011.

[2] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-power wireless bus. In *ACM SenSys*, 2012.

[3] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with Glossy. In *ACM/IEEE IPSN*, 2011.

[4] K. Leentvaar and J. Flint. The capture effect in FM receivers. *IEEE Trans. Commun.*, 24(5), 1976.

[5] J. Lu and K. Whitehouse. Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks. In *IEEE INFOCOM*, 2009.

[6] M. Pajic, S. Sundaram, J. L. Ny, G. J. Pappas, and R. Mangharam. Closing the loop: A simple distributed method for control over wireless networks. In *ACM/IEEE IPSN*, 2012.