Low Level Error Detection For Real-Time Wireless Communications

Jeferson L. R. Souza and **José Rufino**
Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, 1749-016 Lisboa, Portugal
LaSIGE - Navigators Research Team
Email(s): jsouza@lasige.di.fc.ul.pt, ruf@di.fc.ul.pt

***Abstract*—The use of wireless networks to support communications with real-time restrictions is becoming a common requirement within environments such as industries, autonomous vehicles, and aerospace technologies, including also the support for cyber-physical systems (CPS). An effective real-time support on the wireless realm is still an open issue, relying on the presence of dependable and fault tolerant communication services, which are built upon fundamental mechanisms such as error detection. In this paper we continue to explore the low levels of the networking protocol stack in the design of a robust foundation to efficiently support real-time on wireless networks; focused on low level error detection, we present the innovative idea to combine error protection mechanisms for enhancing the capabilities and accuracy of a wireless node in the detection of node failures. This paper shows how our low level error detection approach can be utilised to detect and differentiate node transient omission failures, node permanent failures (e.g., in the transmitter circuitry), and node crash failures in wireless communications.**

***Index Terms*—error detection; wireless communications; real-time; dependability; timeliness; wireless sensor and actuator networks**

I. Introduction

The provision of real-time guarantees on wireless communications is still an open challenge, which is derived by the lack of a robust abstract communication model and its related supporting technologies offering resilient and real-time communication services, even in the presence of disturbances, such as transient overload and network errors [1], [2].

In wired networks, enhancements in the lowest levels of the networking protocol stack had proven to be highly effective in dealing with network errors and in the provision of real-time communication services [3], [4]. A similar strategy to enhance low levels of the networking protocol stack can also be utilised in the wireless realm, where innovative solutions towards the provision of resilient real-time wireless communications are being designed and developed [1], [2], [5].

Though such solutions are restricted to one-hop wireless network segments, the benefits from that approach can be easily and effectively extended to multi-hop settings with strict temporal restrictions, such as those found (in general) in cyber-physical systems (CPS), having a positive impact in the analysis of end-to-end message schedulability guarantees [6], [7], [8] and in ensuring determinism and bounded timeliness properties in multi-hop networking communications.

Error detection plays a fundamental role in the provision of reliable networking communications, being the frame check sequence (FCS) mechanism the first error filter applied by the medium access control (MAC) sublayer to verify the integrity of incoming frames. FCS is usually a silent filter that discards erroneous frames without any notification; the absence of such error notifications hides details concerning the state of the communication channel and node error patterns.

Traditional approaches to reliable networking communications often disregard the potential of low level error detection. Assuming networking communication models where the lowest levels are unreliable, mechanisms to deal with the failure of networking components (wireless nodes included) are routinely designed using timeout-based approaches, which have just to rely on the accurate dimensioning of timers to avoid violations of the timeliness properties of the entire system.

Without diminishing the usefulness of timers and timeout-based mechanisms, this paper builds upon an extension to the standard FCS mechanism, which adds the ability of providing a management notification if an erroneous frame is received. This simple yet innovative method has proved effective in the detection of communication channel failures [5]. This paper extends such raw mechanism with three new features: with a very high coverage, the ability to extract correct information (e.g., source node address) from an erroneous frame; the ability to detect the permanent failure of source node transmit machinery (such as the FCS generator malfunction); and the ability to detect node's crash failures.

We believe that these new mechanisms will be of fundamental importance to design and develop highly effective node failure detection and membership services, traditionally non-existent in standard wireless technologies, yet instrumental to the provision of reliable real-time communication services [1].

To present our contributions this paper is organised as follows: Section II presents the system model and the detailed description of an abstract communication model dubbed wireless network segment (WnS), which is suitable for one-hop real-time wireless communications; Section III presents our proposed low level error detection mechanisms, and its benefits for error detection in resilient real-time wireless

This work was partially supported by the EC, through project IST-FP7-STREP-288195 (KARYON); by FCT/DAAD, through the transnational cooperation project PROPHECY; and by FCT, through project PTDC/EEI-SCR/3200/2012 (READAPT) and through LaSIGE Strategic Project PEst-OE/EEI/UI0408/2014.

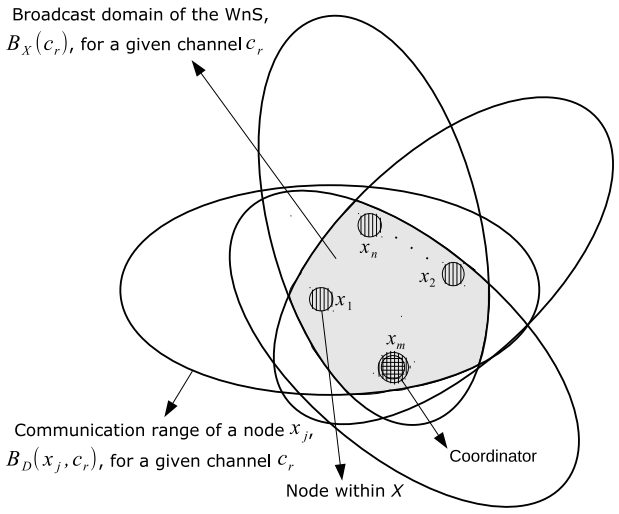

Fig. 1. A graphical representation of the WnS communication model

communications; finally, Section IV presents the conclusions and future directions of the work presented in this paper.

II. System Model and The Wireless Network Segment Approach

All networking communications described within this paper are subjected to the characteristics of a data link layer communication model dubbed Wireless network Segment (WnS). The WnS is a broadcast network segment where communication devices are able to communicate directly and sense transmission from each other (one-hop distance). The communication devices are dubbed networking nodes, which are devices with capability to communicate throughout a shared wireless transmission medium. The terms networking node, wireless node, or simply node, are used interchangeably in the remaining of the paper.

The utilisation of the WnS communication model, introduced in [1], is herein extended and formalised. A formal definition of the WnS is expressed by a 4-Tuple, $WnS = \langle X, x_m, C, W \rangle$, where X is the set of the wireless nodes members of the WnS; x_m is the WnS coordinator, $x_m \in X$; C represents a set of communication channels; and W represents the set of access modes utilised to access the network within the WnS.

The set of WnS members is defined by $X = \{x_1, \ldots, x_n\}$, where the cardinality $\#X$ represents the number of nodes within the WnS. All communications are performed through a set of non-overlapping communication channels, $C = \{c_1, \ldots, c_z\}$, where each $c_r \in C$ is a unique channel, being $1 \leq r \leq z$. In case of a WnS using only one channel, $z = r = 1$, i.e., $\#C = 1$. The access to the network is characterised by a set of access modes $W = \{w_1, \ldots, w_s\}$, where each $w_v \in W$ is utilised by nodes, to control the network access for each channel $c_r \in C$, being $1 \leq v \leq s$, and defining its timing characteristics through φ_{w_v}. Examples of access modes include carrier sense with collision avoidance (CSMA/CA) and time division multiple access (TDMA).

Every node $x_j \in X$ is included in the set of recipients of each transmitted frame, for each channel $c_r \in C$. The broadcast domain of the WnS, for a given channel $c_r \in C$, is defined by: $B_X(c_r) = \bigcap_{j=1}^{\#X} B_D(x_j, c_r), \ \forall x_j \in X$, where $B_D(x_j, c_r)$ is a geographic region that represents the communication range of a node x_j for a given channel c_r.

Let $P(x_j, c_r)$ represent the geographic position of node x_j transmitting on channel c_r. A node $x_j \in X$ if, and only if, $\exists c_r \in C$ where $P(x_j, c_r) \subseteq B_X(cr)$. Otherwise, as a consequence of node mobility, a node $x_j \notin X$ if, and only if, $\forall c_r \in C$, $P(x_j, c_r) \nsubseteq B_X(cr)$.

Figure 1 illustrates a graphical representation of the WnS, where the ellipses characterise the communication range of nodes for a given channel $c_r \in C$, being the grey area the characterisation of the broadcast domain for a given WnS. In practice, the communication range of each node may assume irregular and complex forms [9].

A. Fault Model

Networking components (e.g., a node $x_j \in X$, or a channel $c_r \in C$) either behave correctly or crash upon exceeding a given number of consecutive omissions (the component's *omission degree bound*), f_o, following a given observation criteria (e.g., the duration of a given protocol execution, $\mathcal{T}_{rd}$). Omission faults may be inconsistent (i.e., not observed by all recipients).

In the context of networking communications, we define an omission as an error that destroys a frame. In this sense, errors derived from the presence of accidental faults are transformed into omissions, which are accounted for the purpose of monitoring networking components at different levels. For each received frame, every node $x_j \in X$ locally accounts the omissions observed at each level.

Despite of their importance we are not considering the presence of intentional faults in our fault model, being such topic addressed properly in future work.

B. WnS abstract channel properties

The characteristics of the low level layers in the wireless networking protocol stack can be abstracted by a set of correctness, ordering, and timeliness properties, which are in essence independent of each particular networking technology. In our WnS abstraction such properties are offered through the facet of an abstract single communication channel we dubbed WnS abstract channel, as illustrated in Fig. 2. A relevant set of WnS abstract channel properties is defined in Fig. 3.

Property WnS1 (*Broadcast*) formalises that it is physically impossible for a node $x_j \in X$ to send conflicting information (in the same broadcast) to different nodes, within the broadcast domain of the WnS [10], $B_X(c_r)$, for a given channel $c_r \in C$.

Properties WnS2 (*Frame Order*) and WnS3 (*Local Full-Duplex*) are common in network technologies, wireless technologies included. Property WnS2 (*Frame Order*) is imposed by the wireless transmission medium of each channel $c_r \in C$,

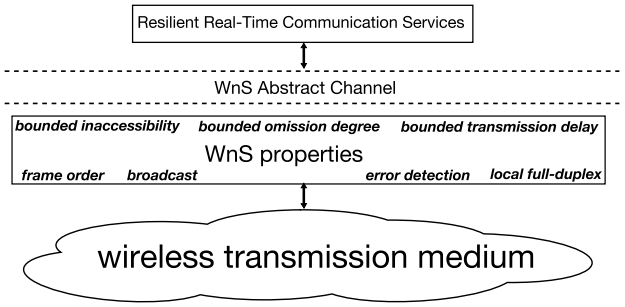

Fig. 2. WnS abstract channel

WnS1 - *Broadcast*: correct nodes, receiving an uncorrupted frame transmission, receive the same frame;

WnS2 - *Frame Order*: any two frames received at any two correct nodes are received in the same order at both nodes;

WnS3 - *Local Full-Duplex*: a correct node may receive, on request, local frame transmissions;

WnS4 - *Error Detection*: correct nodes detect and signal any corruption done during frame transmissions in a locally received frame;

WnS5 - *Bounded Omission Degree*: in a known time interval $\mathcal{T}_{rd}$, omission failures may occur in at most k transmissions;

WnS6 - *Bounded Inaccessibility*: in a known time interval $\mathcal{T}_{rd}$, a wireless network segment may be inaccessible at most i times, with a total duration of at most $\mathcal{T}_{ina}$;

WnS7 - *Bounded Transmission Delay*: any frame transmission request is transmitted on the wireless network segment, within a bounded delay $\mathcal{T}_{td} + \mathcal{T}_{ina}$.

Fig. 3. WnS communication properties

and results directly from the serialisation of frame transmissions on the shared wireless transmission medium. Property WnS3 (*Local Full-Duplex*) specifies that the sender itself is also included in that ordering property, as a recipient.

Property WnS4 (*Error Detection*) has both detection and signalling facets; the detection facet, traditionally provided by the MAC sublayer, derives directly from frame protection through a frame check sequence (FCS) mechanism, which most utilised algorithm is the cyclic redundant check (CRC); the signalling facet is provided by the FCS extension introduced in [5], which is able to signal omissions detected in frames received with errors. No fundamental modifications are needed to the wireless MAC standards, such as IEEE 802.15.4 [11]. The use of such unconventional extension is enabled by emerging controller technology, such as reprogrammable technology and/or open core MAC sublayer solutions, such as the transceivers and the MAC sublayers developed by ATMEL [12]. The residual probability of undetected frame errors is negligible [13], [14].

Property WnS5 (*Bounded Omission Degree*) formalises the failure semantics introduced earlier in the fault model definition, being the abstract omission degree bound, $k \geq f_o$. The omission degree of a wireless network segment can be bounded, given the error characteristics of its wireless transmission medium [14], [15], [16].

The *Bounded Omission Degree* property is one of the most complex properties to secure in wireless communications. Securing this property with optimal values and with a high degree of dependability coverage may require the use of multiple channels. In [5] we have advanced on how this can be achieved by monitoring channel omission errors, and switch between channels upon detecting that the channel omission degree bound has been exceeded.

The time domain behaviour of a WnS is described by the remaining properties. Property WnS7 (*Bounded Transmission Delay*) specifies a maximum frame transmission delay, which is $\mathcal{T}_{td}$ in the absence of faults. The value of $\mathcal{T}_{td}$ includes the medium access and transmission delays and it depends on message latency class and overall offered load bounds [17], [18]. The value of $\mathcal{T}_{td}$ does not include the effects of omission errors. In particular, $\mathcal{T}_{td}$ does not account for possible frame retransmissions. However, $\mathcal{T}_{td}$ may include extra delays resulting from longer WnS access delays derived from subtle side-effects caused by the occurrence of periods of network inaccessibility [16].

A period of network inaccessibility is a disturbance that may be induced externally by electromagnetic interference, or by glitches in the MAC sublayer operation, such as those that may result from the omission of a MAC control frame (e.g., beacon). The network cannot be considered failed; it only enters into a temporary state where the communication service is not provided to some or all of the nodes. Hence, nodes may experience a loss of connectivity within a WnS; the loss of connectivity due to node mobility is also treated under the inaccessibility model. Therefore, the bounded transmission delay includes $\mathcal{T}_{ina}$, a corrective term that accounts for the worst-case duration of inaccessibility glitches, given the bounds specified by property WnS6 (*Bounded Inaccessibility*). The inaccessibility bounds depend on, and can be predicted by the analysis of MAC sublayer characteristics [16].

III. Low Level Error Detection For Real-Time Wireless Communications

The ability to detect errors occurred on networking transmissions is the foundation to provide reliable wireless communications, and therefore to establish a set of useful services for distributed real-time systems such as node failure detection and node membership (abstractly represented by the set X of the WnS). This section focus in presenting some fundamental low level error detection mechanisms that are utilised to augment the error detection capabilities of wireless nodes.

A. Enabling low level enhancements for error detection

Enabling low level enhancements for error detection includes the ability to know when a frame has been received, even if such frame contain errors.

The FCS extension presented in [5] introduces minor but fundamental modifications to the standard frame processing, which are essential to secure the signalling facet of the *Error Detection* property of the WnS. For self-containment purposes,

Algorithm 1: The FCS extension presented in [5]

```
1  Initialisation phase;
2  fcs_error ←false;
3  timestamp ← −∞;
4  begin
5    loop
6      when Channel.indication(frame) do
7        timestamp ← MAC.readClock();
8        frame_header ←MAC.get.header(frame);
9        if MAC.FCS.check(frame) is OK then
10         fcs_error ← false;
11         MAC.indication(frame) ;
12       end
13       else
14         fcs_error ← true;
15         MAC.frame.discard(frame) ;
16       end
17       MAC.Mgmt-Ext.indication(timestamp, frame_header, fcs_error) ;
18     end when
19   end loop
20 end
```

let us briefly present the FCS extension [5] that is reproduced in Algorithm 1.

When a frame is received (line 6), the frame header is extracted (first highlighted modification at line 8), and the FCS check is performed at line 9. Should a frame be received without errors, the standard frame processing procedure is followed and a correct frame is delivered above MAC (line 11), at the WnS abstract channel interface (Fig. 2). If a frame is received with errors it will be simply discarded (line 15) and no data is delivered at the WnS abstract channel interface. The extensions introduced in Algorithm 1 to the standard FCS mechanisms specify that a notification is delivered at the MAC management interface, as highlighted at line 17. This means it is possible for a wireless node to be aware when it is receiving frames with errors through a given channel $c_r \in C$. Algorithm 1 specifies that a relevant part of a received frame (e.g., the frame header) is delivered at the management interface for further processing. However, one question is whether this information is useful in the event a frame has been corrupted by errors?

Surely, the simple notification that an error (transformed into an omission) has occurred is useful. In [5] we have exploited this feature to secure an innovative channel selection function. In that case, a given node $x_j \in X$ is able to issue management notifications that are used for channel monitoring purposes: channel omission failures are detected and accounted for; if the allowed number of successive channel omission failures is exceeded, a switch to a different channel is performed.

Nevertheless, the fundamental question remains: could we use the header information extracted from an erroneous frame?

B. Securing protection of relevant frame content

The use of FCS protecting the entire content of a frame as a single unit allows the detection of errors during networking communications with a very high probability [13], but one cannot know which part of the frame was corrupted, and therefore is not possible to know which bits of a frame were modified. As the corruption may be in any part of the received frame, if one wants to extract important information from a frame received with errors, the specific content to be extracted has to be protected by an additional integrity verification mechanism.

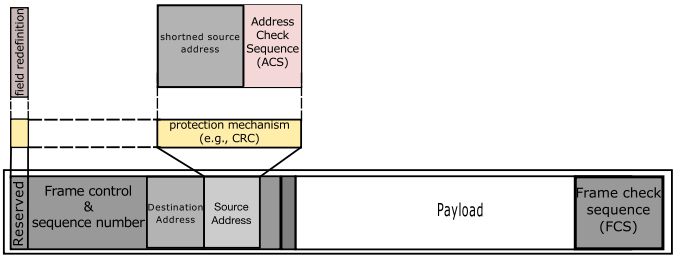

Fig. 4. Representation of our additional integrity verification mechanism for particularly important frame contents (partial frame header protection)

How our proposed integrity verification mechanisms works: Use case of partial frame header protection

The protection provided by the integrity verification mechanism can be applied to any important part of the frame content. In an abstract perspective, the introduction of an additional integrity verification mechanism for important contents of a frame is similar to put an "extra shield" over such contents to protect them against external disturbances on the communication channel, particularly improving the dependability of its transfer.

For many networking standards, some frame formats can be extended or redefined to accommodate the required extra integrity verification data, without any change in the overall frame format, and therefore without any frame overheads.

To illustrate our approach let us use the protection of the source address and its related reserved control field as an example. In Fig. 4, a reserved control field is utilised to signal the source address field is now protected and shortened; and in the place of the source address a compound field is stored, consisting in a shortened version of the source address plus an additional integrity verification field dubbed address check sequence (ACS), which protects the source address and the reserved control field altogether. The total length of the standard source address field is unchanged, implying no extra overhead to the frame length.

In Fig. 4, when the integrity of the reserved control field and the shortened source address of a frame are not compromised, we are able to know which node transmitted the received frame, even if FCS check has failed; the additional protection of just few bits offered by ACS has the benefit of allowing a dependable extraction of important information from frames received with errors.

Algorithm 2 describes the ACS check procedure, which is utilised to materialise our dependable extra shield for practical purposes, being used as a complement of the FCS extension presented in [5]. We assume here all nodes, $\forall x_j \in X$, start to use our dependable extra shield mechanism after being confirmed as a member of a given WnS.

For all frames received and notified by the management

Algorithm 2: Frame header integrity verification

```
1  Initialisation phase;
2  X;                          // Representing the set of nodes members of the WnS.
3  node_id;
4  reserved_field;
5  begin
6    loop
7      when MAC.Mgmt-Ext.indication(timestamp, frame_header, fcs_error)
       do
8        reserved_field ← MAC.Mgmt-Ext.extractReserved(frame_header);
9        node_id ← MAC.Mgmt-Ext.extractNodeID(frame_header);
10       if reserved_field is ACS ∧ MAC.ACS-Ext.check(frame_header) is
         OK ∧ node_id ∈ X then
11         MAC.Mgmt-Ext.indication(timestamp, node_id, fcs_error) ;
12       end
13     end when
14   end loop
15 end
```

Algorithm 3: Detecting node permanent failures

```
1  Initialisation phase;
2  nodeOd[x_1, ..., x_n] ← 0;      // Omission failures for each node of the WnS.
3  k_perm; // Defines the threshold to detect a permanent node failure. A reasonable
   value for such bound has to be greater than the omission degree bound to detect
   channel failures [5].
4  begin
5    loop
6      when MAC.Mgmt-Ext.indication(timestamp, node_id, fcs_error) do
7        if fcs_error is true then
8          nodeOd[node_id] ← nodeOd[node_id] + 1 ;
9          if nodeOd[node_id] > k_perm then
10           MLA.Mgmt.indication(node_id, node_perm_failure) ;
11         end
12         else
13           nodeOd[node_id] ← 0 ;
14         end
15       end
16     end when
17   end loop
18 end
```

indication of line 7 (originated from the FCS extension), the format and integrity of the protected content is verified through the ACS field (line 10). If the integrity of any of the protected content has been not compromised, a management indication is generated on its turn (line 11) to inform the availability of such content (e.g., $node\_id$).

C. Detecting node permanent failures

Permanent failures are characterised by the consecutive reception of frames with errors from the same node. Assuming both ACS generation and checking procedures use methods that are not affected by a failure in the transmitter circuitry, a permanent node failure indicates a malfunction in the transmitter of that node, which can induce a bad FCS for outgoing frames from such transmitter. However, at a recipient, individual omission failures with origin in a transmitter node cannot be distinguished from omission failures with origin in the communication channel.

The value defined for k_{perm}, the threshold to detect a permanent node failure, must then include two distinct contributions: the allowed number of consecutive omissions with origin in the node being monitored, as produced by the node's transmitter circuitry, k_{txfail}; and the channel omission degree bound, k, as defined by property WnS5 (*Bounded Omission Degree*). Thus, $k_{perm} = k_{txfail} + k$.

Algorithm 4: Detecting node crash failures

```
1  Initialisation phase;
2  node_live_period[x_1, ..., x_n]; // Aliveness period for each node of the WnS.
3  begin
4    loop
5      when MLA.Mgmt.indication(node_id, max_idle_period,
       has_joined_WnS) do
6        node_live_period[node_id] ← max_idle_period + T_td + T_ina ;
7        MLA.Mgmt.startTimer(node_live_period[node_id], node_id) ;
8      end when
9      when MLA.Mgmt.indication(node_id, has_left_WnS) do
10       MLA.Mgmt.stopTimer(node_id);
11     end when
12     when MAC.Mgmt-Ext.indication(timestamp, node_id, fcs_error) do
13       MLA.Mgmt.restartTimer(node_live_period[node_id], node_id) ;
14     end when
15     when MLA.Mgmt.indication(node_id, timer_expired) do
16       MLA.Mgmt.indication(node_id, node_crash_failure) ;
17     end when
18   end loop
19 end
```

Algorithm 3 has been designed to detect permanent failures of nodes within the WnS. In the context of Algorithm 3, an omission is accounted for a given node (line 8) by the reception of a frame with FCS error (line 7). When such node exceeds k_{perm}, a permanent failure is detected (line 9), and notified through a management notification (line 10). Otherwise, if a correct frame is received from a given node, $node\_id$, its omission degree is cleared (line 13).

D. Detecting node crash failures

Although earlier in this paper we have disregarded the utilisation of timeout-based approaches to detect transient and permanent node failures, the only possibility to detect node's crash failures (i.e., the absence of node's activity on the WnS) is using the notion of time. In concrete, we define a timeout-based approach (Algorithm 4) to detect the crash of nodes on the WnS, where timers are dimensioned consonant with the temporal behaviour of the WnS, which is dictated by the WnS properties WnS6 (*Bounded Inaccessibility*) and WnS7 (*Bounded Transmission Delay*).

In Algorithm 4, a node starts to be monitored (line 7) when it joins the WnS (line 5). To be able to detect node crash failures we assume each node of a given WnS, $x_j \in X$, has to inform other nodes of its maximum idleness period (represented by the max_idle_period parameter), which is utilised to monitor that node activity; the interval of the idleness period of a given node can be derived, for example, from the periodic transmission of heartbeat control frames for membership maintenance.

For each node $x_j \in X$ (represented by $node\_id$) a timer is started (line 7) with a timeout value (line 6) equal to the maximum time interval between two consecutive frame transmission requests from a given node plus the worst case time required to transmit the frame, which is $\mathcal{T}_{td} + \mathcal{T}_{ina}$, as specified by the WnS7 property (*Bounded Transmission Delay*).

The bounded transmission delay of the WnS (property WnS7) accounts for the worst case transmission delay, $\mathcal{T}_{td}$, and

for the presence of periods of network inaccessibility (property WnS6 of the WnS), being the value assumed for $\mathcal{T}_{ina}$ the worst case duration derived from the network technology utilised.

For each frame received from a given node represented by $node\_id$ (line 12), the timer instantiated to monitor the activity of such node is restarted (line 13). If a node crashes, no more frame transmissions are received from that node and the corresponding timer expires (line 15). The crash of $node\_id$ is declared through a management notification (line 16).

E. Integration in IEEE 802.15.4 networks

The algorithms we have just discussed can be directly applied to IEEE 802.15.4 networks: the original source address field has a size of 16 bits; we define the shortened address with a size of 10 bits (allowing the identification of up to 1024 nodes), leaving 6 bits for the ACS. A CRC algorithm, such as CRC6 [19], can be used for that purpose with reasonable coverage [20]. Given we are using a standard frame format, there is no need to include any integrity verification data in the payload part of the frame and therefore no frame overhead penalties arise. Since the size of the address field and the typical length of a IEEE 802.15.4 (approx. 924 bits using 16-bit addressing) we roughly estimate that a source address extraction will not succeed only in 1% of the frames.

Additionally, we use $k_{perm} = 4$ as a reasonable value of the threshold for detecting node permanent failures in IEEE 802.15.4 networks, being derived from the sum of the IEEE 802.15.4 omission degree bound, $k = 3$, utilised to detect channel failures [5], and the number of consecutive omission produced by the same node, which is assumed $k_{txfail} = 1$.

IV. Conclusion

This paper has presented a forward looking innovative idea of an extension to the frame check sequence procedures that enables the detection and signalling to high protocol layers, such as protocols implementing resilient real-time communications, that frame have arrived with errors.

Furthermore, we have presented three additional algorithms that allows: to extract correct relevant information (e.g., node source address) from an erroneous frame; the ability to detect the permanent failure of a node (e.g., due to the failure of its transmit circuitry); and the capability to detect the crash failure of a node when it simply stops to operate.

These algorithms are particularly useful and relevant to build a node failure detection and membership services, as a part of a wider and comprehensive approach to support resilient real-time communications over wireless networks.

References

[1] 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, Portugal, 2011.

[2] ——, "Towards resilient real-time wireless communications," in *25th Euromicro Conference on Real-Time Systems (ECRTS-WiP)*, Paris, France, July 2013.

[3] 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.

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

[5] J. L. R. Souza and J. Rufino, "Analysing and reducing network inaccessibility in IEEE 802.15.4 wireless communications," in *38th IEEE Conference on Local Computer Networks (LCN)*, Sydney, Australia, October 2013.

[6] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Priority assignment for real-time flows in WirelessHART networks," in *23rd Euromicro Conference on Real-Time Systems (ECRTS)*, 2011, pp. 35–44.

[7] W. Shen, T. Zhang, M. Gidlund, and F. Dobslaw, "SAS-TDMA: A Source Aware Scheduling Algorithm For Real-Time Communication In Industrial Wireless Sensor Networks," *Wireless Networks*, vol. 19, no. 6, pp. 1155–1170, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11276-012-0524-2

[8] W. Dong, C. Chen, X. Liu, K. Zheng, R. Chu, and J. Bu, "Fit: A flexible, lightweight, and real-time scheduling system for wireless sensor platforms," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 1, pp. 126–138, Jan 2010.

[9] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of radio irregularity on wireless sensor networks," in *in MobiSYS 04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*. ACM Press, 2004, pp. 125–138.

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

[11] IEEE 802.15.4, "Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs) - IEEE standard 802.15.4," 2011.

[12] ATMEL, *ATMEL AVR2025: IEEE 802.15.4 MAC Software Package - User guide*, ATMEL Coorporation, May 2012.

[13] T. Fujiwara, T. Kasami, A. Kitai, and S. Lin, "On the undetected error probability for shortened hamming codes," *IEEE Transactions on Communications*, 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 *Annual Conference of the Special Interest Group on Data Communication (SIGCOMM)*, 1996.

[15] 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.

[16] J. L. R. Souza and J. Rufino, "Characterization of inaccessibility in wireless networks - a case study on IEEE 802.15.4 standard," in *IFIP 3th International Embedded System Simposium IESS*, September 2009.

[17] 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

[18] M. Hameed, H. Trsek, O. Graeser, and J. Jasperneite, "Performance investigation and optimization of IEEE 802.15.4 for industrial wireless sensor networks," in *IEEE 13th International Conference on Emerging Technologies & Factory Automation (ETFA)*, September 2008.

[19] 3G 3rd Generation Partnership, "Physical layer standard for CDMA2000 spread spectrum systems - revision d - version 2.0," 2004.

[20] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," in *International Conference on Dependable Systems and Networks (DSN)*, June 2004, pp. 145–154.