Supporting Critical Modes in AirTight

J. Harbin, D. Griffin, A. Burns, I. Bate, R.I. Davis and L.S. Indrusiak
Department of Computer Science, University of York, UK.

***Abstract*—The AirTight protocol supports mixed criticality wireless traffic and temporal guarantees based on defined fault models. In some systems, following a catastrophic failure, it is necessary to communicate crucial data away from the site of the failure in order to better understand (post-hoc) the reasons why it occurred. To support this action it is necessary for a mode change request to be propagated to all the non-failed nodes in the system, and for these nodes to switch their behaviour so that the crucial data is given high priority in its use of the wireless network. This paper explains how AirTight can support such a critical mode change. A uni-cast protocol is utilised to flood the system with mode change messages, each node then locally prioritizes its use of the available bandwidth to support the defined UC (Ultra-Criticality) packet flows. An aircraft engine control scenario is used to motivate the requirements for the mode change protocol. Protocol-accurate simulations are then used to illustrate and evaluate the approach.**

I. Introduction

AirTight [2] is a wireless protocol (built upon the physical and MAC layers of IEEE 802.15.4) that supports mixed-criticality real-time traffic between computational nodes. With any wireless communication it is not realistic to assume fault-free behaviour. Rather, as in other considerations of fault tolerance, we require that certain levels of performance are delivered when the likelihood and severity of faults is bounded by what is referred to as a *fault model*. We assume that the physical layer of the protocol incorporates the usual methods of increasing resilience (for example spectrum spreading), AirTight therefore supports analysis that models the faults that manifest themselves as unacknowledged frame transmissions at the MAC layer.

Within AirTight the run-time behaviour is controlled via two-level scheduling. A system-wide slot table determines when each node can transmit and when each node must be open to receive (and on which channel). Local to each node is a fixed-priority scheduler that determines which packets to transmit when it has a transmission slot. An application packet (or flow) consists of a small sequence of frames; and it is assigned a criticality level [3]. As frames are transmitted each node keeps a count of the number of its transmission failures. When this number is below a defined limit, frames are simply resent. But if this limit is reached, a local critically mode change is made at the node and only the more critical packets are transmitted.

With a system defined to have two criticality levels, HI and LO, response-time analysis is used to verify that all packet deadlines are met if a lower threshold on the number of faults is satisfied. If this threshold is violated but a higher threshold is satisfied then the analysis will establish that all HI-critical packets are delivered by their deadlines. When there are currently no further frames to transmit then the node's failure count is re-set to zero.

In this paper we extend the scope for AirTight by defining the required behaviour of each node when there are more faults than the higher threshold specified, or when there is a functional mode change to the entire system brought about by a severe failure or attack. Although AirTight is a uni-cast protocol it uses a flooding scheme to communicate the requirement for this mode change to all non-failed nodes in the system. Each of these nodes then switches its local criticality mode to Ultra-Critical, UC. This will impact the set of local tasks that are executed and on the set of packets that are communicated. Within the context of the experienced failure, these tasks and packets may be 'new' (i.e. only occur in this UC mode), be existing HI-criticality tasks/packets or even be LO-criticality tasks/packets that have increased significance in the new mode.

II. Engine Malfunction Use Case

An aircraft engine is a harsh environment for electronics and wireless communication in that there are a lot of moving mechanical parts generating both interference and attenuating radio signals. Nevertheless, wireless sensors have two distinct advantages: (1) the sensors can be put deep inside the engine where it is not feasible to have cabling; and (2) it removes the weight and maintenance of cabling. The difficulty of maintenance may also mean that the designer may want to fit a number of replicas so replacement is not necessary. Current engines have a number of sensors. With a shift towards more intelligent control and monitoring, this number will grow. Internal to the engine there are failures that may affect the wireless communications but also may affect the requirements of the system. For example, in the case of a shaft break, there will be a significant amount of mechanical damage, which may cause nodes to fail and may lead to large pieces of material (including metal) being in unanticipated positions.

External to the engine there are a number of controlled interference sources, e.g. from the rest of the aircraft, and un-controlled interference sources, e.g. high-intensity radiated fields including lightning, mobile phones, laptops etc. This leads to complex fault behaviour that cannot be fully defined at design time. We therefore utilise a collection of fault models (one per criticality level) that are, in themselves, bounded. Finally, a number of parts of the overall aircraft system (and logistical support equipment on the ground) may want to use wireless communications and as such the aircraft engine should be designed to share the same parts of the spectrum especially as the whole aircraft could have hundreds if not thousands of sensors.

A good example of the potential deployment of a wireless communication media is within an aircraft engine for the purposes of active health monitoring [4]. Figure 1 shows the communication graph (black lines) for a 25-node wireless network inspired by a possible engine monitoring system; it is clear that the topology of this example is a 5-node subsystem repeated 5 times. Actual data flows are shown as blue arrows. While this may not entirely represent how aircraft engines will ultimately use wireless communications, it is representative. An aircraft engine has a limited amount of space available to mount wireless sensors. In these places there are opportunities to use energy harvesting, e.g. using vibration, to power the nodes. Therefore in these locations there will be a number of smart sensors (i.e. nodes) monitoring different properties of the engine which will then communicate with the rest of the engine via a signal concentrator. Then, in one central location there will be the traditional aircraft engine controls system (termed a FADEC – Full Authority Digital Control system) that takes all the signals, provides the primary control and monitoring, and importantly provides the links to the Avionics Full-Duplex Switched Ethernet (AFDX), i.e. the communications to the rest of the aircraft.

We have used this 5-node subsystem to illustrate the analysis associated with AirTight [2], and have validated this analysis using a prototype network of 5 IEEE 802.15.4 compliant nodes. We also used a protocol-accurate in-house simulator to evaluate AirTight's performance and scalability over the complete 25-node network. In total this network has 55 packet flows mapped to the 25 nodes; 25 of these flows are defined to be of HI-criticality and 30 of LO-criticality.

In this paper we will use this example to illustrated how AirTight supports the need for the criticality mode change that would follow a significant mechanical failure, e.g. a shaft break. A shaft break (or similar catastrophic failure) is a very rare but not unknown event[1]. It is an interesting example within the context of this work for two reasons. Firstly, as the engine is effectively damaged beyond repair then this event is rarely, and certainly not comprehensively, investigated on a test rig which means if/when it does happen for real there is a strong desire to get as much engine data as possible into long-term storage for later diagnosis and understanding. Secondly, from the point at which the shaft break is detected, more complex control algorithms are performed for a limited amount of time but this extra functionality can be at the expense of some of the "normal" functionality including that which is normally HI-criticality. For these reasons, the shaft break mode change can be modelled as follows: (i) the amount of data being communicated from the smart sensors to long-term storage is increased by a factor of, perhaps, 5; (ii) the time for which the best-effort communications must be maintained is, for example, 20 seconds; (iii) a percentage of nodes will be randomly lost, e.g. 10%; (iv) as some nodes may be lost, including those responsible for signal concentration and communications to the airframe, some signals may need to be sent to a number of sinks instead of just one; and (v) a percentage of the "normal" HI-criticality messages will become LO-criticality, e.g. 50%.

[1]An example is reported in https://www.nbcnews.com/storyline/airplane-mode/faa-orders-a380-engine-inspections-after-midair-failure-emergency-landing-n810341.

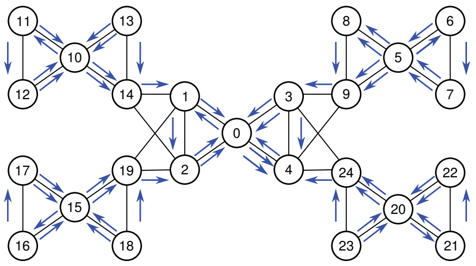

Fig. 1. Communication Graph of a 25 node Health Monitoring System

III. Overview of AirTight

We assume a distributed system of nodes that can each perform any combination of executing tasks, producing/ consuming data from sensors/tasks, writing to actuators and relaying data packets to and from other nodes. The AirTight protocol has the following basic properties (most of them inherited from the parent standard IEEE 802.15.4):

- Peer-to-peer packet-switching communication between tasks/nodes is the normal use of the network. Packets are sent as one or more *frames*. Each successful frame transmission is always acknowledged by the receiver through the transmission of a short ACK frame.
- Multi-hop routing is required due to the limited transmission range of each node.
- Buffers exist on each node to store frames in transit (the size of the buffers required on each node can be determined during the offline schedulability analysis).
- Nodes have line power or local harvesting, so energy efficiency/battery life is not a limiting concern.
- Multiple frequency bands (channels) are available in IEEE 802.15.4 (up to 16 in the 2.4GHz band) but a node can only use one channel at a time.
- Node communications are represented by two graphs: the *communications graph* and the *interference graph*:
  - The communications graph $\mathbf{C}$: if there is an edge from $A \rightarrow B$ in $\mathbf{C}$, then the two nodes can communicate directly. This is required to be a symmetric graph due to the necessity for an acknowledgement to be returned to the sender, so $A \rightarrow B$ implies $B \rightarrow A$.
  - The interference graph $\mathbf{I}$: if there is an edge from $A \rightarrow B$ in $\mathbf{I}$, then a transmission from A will prevent B from receiving a frame from any node other than A on that channel at that time.

Note $\mathbf{C}$ is a subgraph of $\mathbf{I}$: if $A \rightarrow B$ is in $\mathbf{C}$ then it will also be in $\mathbf{I}$.

It is assumed that the packets to be communicated have tight timing constraints (i.e. deadlines). We also require that the system supports applications of different levels of criticality.

AirTight is designed to balance efficiency and flexibility. At the system level, its media access control is table-driven, but at the node level it uses criticality-aware priority-based frame scheduling. The protocol is based around the repeated application of the slot tables which, in time, define the activities of each node – either transmission or reception on that channel, or null meaning no usage. The slot (or scheduling) table (ST) consists of a series of slots. Each slot is assigned to a node and can be used by that node to send a single data frame on a designated channel. The slot also accommodates the ACK frame of the respective receiver.

At each node, local scheduling decisions are made to manage the use of the node's slot allocation. We employ a fixed-priority scheme. A set of FIFO queues (buffers), one per priority level, are used to hold the frames that need to be transmitted. Each normal flow has a unique priority and hence a specific buffer. The frames from the same flow are stored in the buffer in FIFO order. Whenever the node has a slot available, it transmits the first frame in the highest priority non-empty buffer. If an ACK is received the frame is removed from the buffer; if no ACK is received, then the frame remains in the buffer and is a candidate for re-transmission when the next available slot for that node becomes available.

AirTight is thus a two level protocol. A collection of slot tables defines the usage of the wireless media. Each slot in a table defines whether the node can transmit in that slot (and on which channel if more than one channel is used), or whether it should listen in that slot (and on which channel), or whether it is off-duty. The collection of tables reflects the properties of the communication and interference graphs.

The fundamental time unit of AirTight is the duration (S) of a slot – the time it takes to communicate a single frame of data and receive an ACK for that frame. In our prototype implementation [2] a slot length of 10ms has been achieved. All parameters of the application, the communication media and the environment (e.g. the usual T_i, C_i, D_i, table length, fault models, etc.) are expressed as an integer number of slot times.

A schedulable AirTight network supporting mode changes is intended to support the following requirements:

- If there are no faults experienced by the system then all packets will meet their deadlines.
- If the faults experienced by the system are no worse than that implied by the LO-criticality fault model then all packets will meet their deadlines. This is defined to be the LO-criticality mode.
- If the faults experienced by the system are no worse than that implied by the HI-criticality fault model then all HI-criticality packets will meet their deadlines. This is defined to be the HI-criticality mode.
- If the faults experienced by the system are worse than that implied by the HI-criticality fault model then we assume that this level of faults implies a permanent degradation to the network and/or the control system it is supporting. This is defined to be the Ultra-Critical (UC) mode, and is the focus of this paper.

For this mixed-criticality behaviour response-time analysis has been developed [2] that can be used to verify an application. This analysis is itself based upon the approach developed for mixed criticality task scheduling [1]; it is not repeated here due to space limitations.

The application's characteristics, together with the per channel interference and communication graphs, and the analysis developed for AirTight, are the inputs required to construct the per channel slot tables. The simplest slot table is one that has a single slot per node (with some slots being used by more than one node if they are not linked in the interference graph). More complex slot tables can be constructed, via search techniques such as the use of Genetic Algorithms that also take task placement and routing into account. The use of these techniques to construct optimal, or near optimal, slot tables forms part of future work and is not considered further here.

IV. Supporting Critical Mode Changes

For ease of presentation we will assume that our system is multi-hop and multi-domain, but single channel.

In the LO- or HI-criticality (i.e. not UC) mode of operations each node (n_i) will have a set of other nodes that it sends messages to. Let this set be represented by P_i (for partners). Clearly each member of P_i is linked to n_i in the communication graph, $\mathbf{C}$. Let the larger set of nodes that n_i could communicate with be denoted by P_i^+. So P_i^+ contains all the partners of n_i in $\mathbf{C}$.

A mode change is triggered within n_i by either an application task that has identified a severe physical failure, or attack, or the AirTight protocol stack having monitored more frame communication failures than can be tolerated in the HI-criticality mode; let this value be represented by G_{HI}. Node n_i also undergoes a mode change if it receives an authenticated 'mode-change' packet from another node. This packet is then passed on to all members of P_i^+.

In the protocol described in this paper, the slot table does not change when the node switches to the UC mode. It is possible to envisage a protocol in which a different slot table becomes more appropriate in the UC mode. But to coordinate the simultaneous switching of all nodes to a new table is not without considerable difficulty. We therefore explore in this work the expressive power of an approach that retains the same slot table in this UC mode. This has the advantage that the mode change can be communicated across the system without the need for coordination. Of course the initial (offline) construction of the slot table could take into account the needs of the UC mode. For example, a node that does not transmit any packets under normal operation, and hence does not require a slot in the table, could be assigned a slot so that it could contribute to the communication of the critical mode change request.

Having recognised the need for a system-wide mode change the node follows the following (initial phase) protocol:

- A single 'mode-change' packet of the highest local priority is queued (buffered) ready to be sent to all members of P_i^+. A *distribution queue* is initialised, containing the members of P_i^+. This distribution queue may optionally be sorted in such a way as to direct the mode change message more quickly in a specific direction.
- When transmitting the packet, its next-hop destination is set to the node at the head of the distribution queue. If the transmission is acknowledged successfully, the entry at the head of the distribution queue is removed. If the distribution queue is empty, then all peers have been informed of the mode change, and the mode change packet is deleted from its buffer.
- If a frame from one of these packets fails to be acknowledged then the associated packet is not removed from the buffer – this is the usual behaviour for AirTight.
- If any frame fails to be sent to the same next-hop destination G_{HI} times (determined by the lack of an acknowledgement), the next-hop destination is removed from the distribution queue – the wireless link or designated node is assumed to be permanently broken as a result of the primary cause of the mode change.
- If node n_i receives a 'mode-change' packet from n_j while it is already distributing its mode change, then n_j is removed from the distribution queue (if it is currently present) – clearly n_j does not need to be informed of the mode change.

The above flooding behaviour ensures that all non-failed nodes receive the mode change request within a bounded period of time. The analysis developed for AirTight [2] can be used to compute this value for various system failure scenarios (an example is provided in the Evaluation section).

In the second phase of the protocol the packets associated with the UC mode are queued and transmitted. These packets arise from:

- Packets that are only sent in the UC mode (perhaps emanating from local tasks that only execute in the UC mode).
- HI-criticality packets that are relevant to UC mode; perhaps with an increased number of frames and/or alternative routes.
- LO-criticality packets that are relevant to UC mode; perhaps with an increased number of frames and/or alternative routes.
- UC packets that originate from other nodes and are being routed through this node.

All UC packets are transmitted with a priority higher than those used for the usual HI-criticality and LO-critically traffic. It is assumed that there is a finite number of packets to be communicated within the UC mode. Perhaps a single packet per originating task (i.e. these tasks are single-shot rather than recurrent). Analysis can again be used to determine how long it will take for such flows to reach their destinations when there are parts of the network unavailable and faults being experienced in the operational parts. Of course if the network is partitioned then it will not be possible to deliver the UC packets unless each partition has a relevant sink.

In the UC mode if there are currently no UC packets to transmit then other HI-criticality packets can be sent. They would have lower priority and hence would not interfere with newly arrived UC packets (from either the host node or being forwarded from other nodes); in general they would not however be guaranteed to arrive before their deadlines. It is assumed that LO-criticality packets, other than those promoted to UC, are not transmitted in the UC mode.

V. Evaluation

In this section, we consider the evaluation of the AirTight ultra-criticality mode change via simulation. The simulator is a discrete event simulation which allows analysis of the latencies of packet flows, and transmission of the mode change. It allows various faults to be defined with different probabilities and locations affected, and individual nodes to be disabled during simulation. The simulator also supports GUI visualisation of the network in the process of simulation, indicating the status of the nodes and their transmission buffers.

In order to evaluate the performance of the protocol, it is important to consider the length of time taken to deliver the UC mode change packet throughout the network, and the UC packets. In addition, we can also assess the impact upon the delivery rates and deadlines of the originally present HI-critical and LO-critical packets.

The case study described in Section II and our previous AirTight work [2] is used for the evaluation.

The slot table size used in this example case study is 30, which is equivalent to 5 copies of the 6-slot table used in [2]. The example topology is shown in Figure 1. The topology has been modified from that used in [2] by the addition of a number of additional links from nodes 9 to 4, 14 to 2, 19 to 1 and 24 to 3 (and since links are symmetric, the reverse). This provides additional redundancy which is required for providing fault tolerance in the event of a shaft break failure disconnecting the original primary wireless link.

The fault case selected for the experimental case study models a shaft break event occurring in the upper right section of the topology. Its effects upon the network are as illustrated in Figure 2. Nodes 8 and 6 fail permanently and the link from node 9 to node 3 is permanently disconnected. This loss of nodes fits with the requirements for the mode change in the aircraft case, in that a small proportion of the network nodes and connection links are lost as a result of the failure. The transmission of the UC mode change is initiated by node 5, which is informed of the shaft break by a reading from one of its directly connected sensors. Upon entering UC mode, nodes 5, 7, 10, 15 and 20 begin

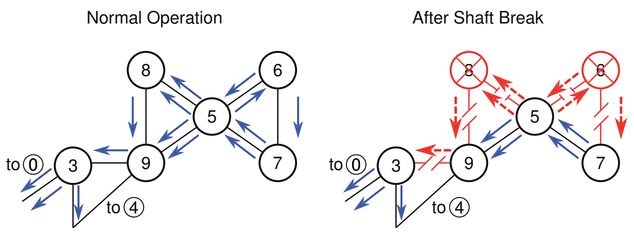

Fig. 2. Effects of the fault on the network (only the affected section shown)

executing a special processing task to gather logging data about the shaft break failure. When this task completes, these nodes transmit *UC traffic* – a data flow destined for node 0 for the central module to convey the logged data.

In an emergency situation, it is important for the mode change to be propagated across the network rapidly. The time taken from the mode change event occurring to the notification propagating across the network is considered in this section. For each case, two metrics are used: the time for the central node which has the wired link to the rest of the network (identified as node 0) to be notified, and the time for every node in the network to be notified. During the flood propagation, the network is subject to different levels of unrelated 'normal' faults, which manifest in a given probability of a transmission failing. This probability is uniform across the network, regardless of location. The length of these fault bursts is increased in the series of experiments performed.

The role of the simulator is to enable different scenarios and fault models to be explored. Clearly this is much easier to do with a simulator than a test-bed. One of the options available with the simulator is to either simulate worst-case fault behaviour, or to model fault arrivals via various stochastic processes. Another choice is whether to assume that each 'link' in the wireless network has dependent or independent faults. For dependent behaviour a fault hits all links at the same time - thus a routed message will perhaps only suffer interference from faults on one of its hops. With independent faults each hop could suffer this interference.

In the following examples of runs of the simulator we first force the faults to occur at the 'worst possible time' but assume faults are dependent. We then consider independent but stochastically modeled faults.

Figure 3 demonstrates the increasing time taken for the distribution of the mode change with increasing length of transient faults. In all experiments time is measured in numbers of slots.

For this experiment, the probability of transmitted data packets being interfered with during the fault interval is 100% - it is assumed, for a worst case, that the transient fault is completely destructive of ongoing traffic. Obviously, the time taken to inform the central node 0 is lower than the time taken to inform every node within the network. It is notable that as the fault length is increased, several discontinuities occur in which the elapsed time required to propagate the mode change increases suddenly. These occur as a result of the interaction between the periodic scheduling table of AirTight and the fault definition.

In this experiment the distribution queues which control the order for transmission of the mode change messages are setup to direct the mode change messages towards the central node 0. We found that this static property almost halved the delivery time by comparison with an arbitrary ordering of the distribution queues.

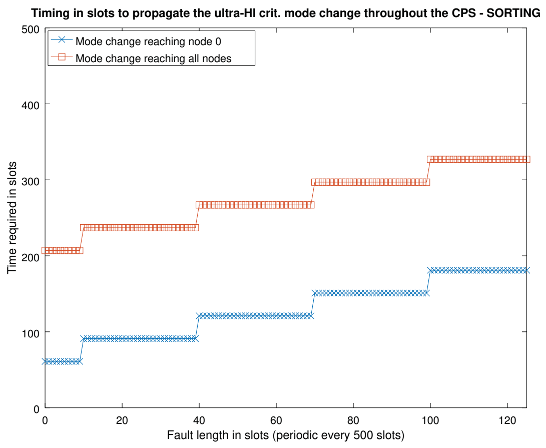

Fig. 3. Timing with ordered queues

Assuming the mode change in this case study represents an aircraft engine shaft break, a number of actions would be required in order to respond to the mode change. Firstly, we would assume that the network node detecting and signalling the mode change would have to perform some processing to determine the nature and effects of the fault that triggered the mode change. Then it would communicate with its coordinator node 0, transmitting some UC traffic in order to transmit additional data via a multi-hop route. Also, the other central nodes from each of the functional regions illustrated in Figure 1 (nodes 10, 15 and 20) would, on receiving the communicated mode change, determine the control effects which are necessary and then transmit the relevant data back to node 0. Given that these communication data flows could involve instructions to other engines that detail how they would have to respond to compensate, this would be transmitted at the highest priority after the mode change itself.

We now consider the latencies involved in the transmission of this ultra-criticality (UC) data traffic. Five UC data flows are activated, referred to as UC1 to UC5.

Figure 4 shows the latencies experienced for the 5 UC data flows, indicating the time following their injection into the network for the complete packets to reach their destination at node 0. It is assumed in Figure 4 that the processing delay to generate UC traffic is very short, effectively less than a single table, so the packets are ready to be transmitted and present in the buffer the next time the source node has a transmission slot.

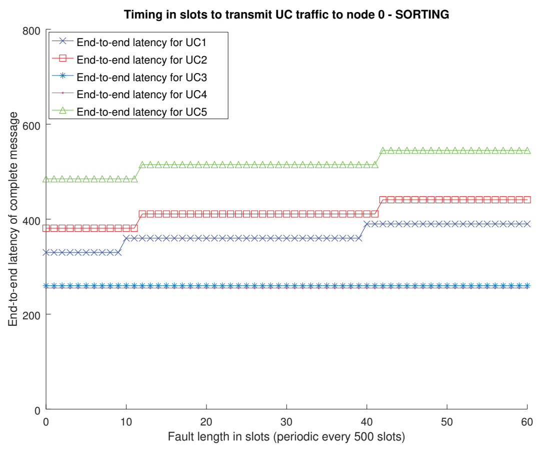

Fig. 4. Latencies for UC data transmission

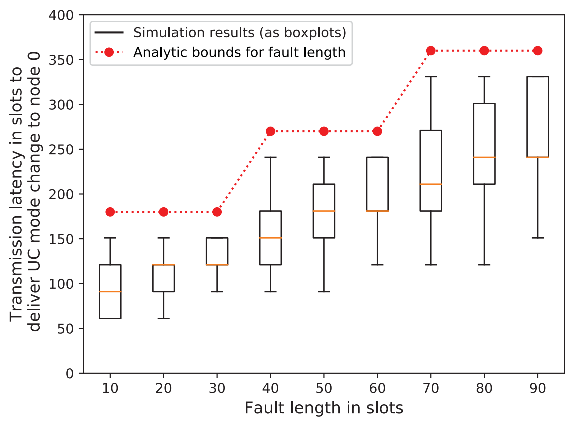

Fig. 5. Latencies for UC Mode Change Message

However, since UC flood packets (with the highest priority) are present in the network, the generated UC data will not be immediately transmitted.

The data series for the latency values is generated by varying the transient fault length which occurs in the network. These faults are assumed to begin immediately upon the mode change, i.e. upon the injection of the UC data traffic. The period of the faults is 500 slots, and the faults are assumed to recur at the beginning of this interval, network-wide (that is, all links in the network are affected simultaneously).

Since flows UC1 and UC2 share a majority of the same route, it is as expected that UC2 has a higher latency than UC1. UC3 and UC4 have disjoint routes, so they do not mutually interfere with each other, and achieve broadly the same latency even though they have different injection times. UC5 experiences the highest latencies since it may receive interference from UC1 and UC2 (due to their requirement to route via node 4).

We now show a result from simulating independent faults that arrive stochastically. Clearly many different arrival patterns can be experimented with. Faults hitting each link of a routed message are likely to be rare; but should nevertheless be investigated. In the following experiment each link experiences a fault that arrives randomly between the arrival of the UH mode change packet and that time plus the table length. So this packet can potentially propagate through the network without interference from faults but the probability of it suffering multiple faults is not negligible. Figure 5 shows a set of box plot results for increasing durations of faults. For each fault duration 1000 simulations were undertaken. Also shown on this figure is the analytical upper bound calculated using the analysis reported in previous work on AirTight [2]. Note, as expected, the longest propagation time observed in the simulation experiments is less than this bound.

VI. Conclusion

In a CPS system, a permanent fault may occur in such a way as to require a different protocol response from those normally assumed in the case of transient faults, such as retransmission and alternative routes. Specifically, it may require a functional mode change to be signalled throughout the network in order to inform the entire system of an emergency situation, as well as triggering the dropping of LO-criticality work. This paper has demonstrated the modification of the AirTight protocol in order to support these more challenging fault scenarios, with only a minor modification of the logic for data distribution at the highest priority level. The timing characteristics of this mode change data and associated logging traffic have been investigated via simulation to demonstrate its performance in the presence of a variety of fault intensities.

Acknowledgements

The research described in this paper is funded, in part, by the EPSRC grants MCCps (EP/P003664/1). No new primary data were created during this study.

References

[1] S.K. Baruah, A. Burns, and R. I. Davis. Response-time analysis for mixed criticality systems. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, pages 34–43, 2011.

[2] A. Burns, J. Harbin, L.S. Indrusiak, I. Bate, R.I. Davis, and D. Griffin. Airtight – a resilient wireless communication protocol for mixed-criticality systems. In *Proc. RTCSA*, 2018.

[3] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proc. of the IEEE Real-Time Systems Symposium (RTSS)*, pages 239–243, 2007.

[4] X. Zhao, H. Gao, G. Zhang, B. Ayhan, F. Yan, C. Kwan, and J.L Rose. Active health monitoring of an aircraft wing with embedded piezoelectric sensor/actuator network: I. defect detection, localization and growth monitoring. *Smart Materials and Structures*, 16(4):1208, 2007.