Mixed-Criticality Swarm Robotics Through Real-Time Wireless Networks

Sven Lars Signer

PhD

University of York

Computer Science

September 2025

Abstract

Communication will be a vital part of real-world applications of swarm robotics. Prior work has shown the existing swarm robotics prototypes are often designed with unrealistic assumptions on network performance and are therefore prone to failure under real world network conditions. This thesis argues for an introduction of a mixed criticality approach from the real-time systems domain to swarm robotics systems, beginning at the network layer. Mixed criticality communication protocols allow the system designer to reason a priori about the network behaviour and provide graceful degradation in the presence of network faults. This more predictable network behaviour is shown to translate into more predictable robot behaviour at the application level.

Since robot behaviour can influence the network conditions, for example due to robot mobility, the application and network components of a swarm robotics system are mutually dependent. Redefining criticality modes from an internal parameter of the network to an interface between the application and network layers allows the application component to adapt its own behaviour in response to the network conditions. This is shown to result in an overall system being better able to adapt to changing conditions. Raising the concept of criticality further towards the application component of a swarm robotics systems through a proposed mixed criticality wireless communication protocol that make criticality mode more meaningful to an application component make it more feasible to provide application level behaviour guarantees, while avoiding the centralised control of other protocols that is antithetical to swarm robotics systems.

Contents

Acknowledgements

My primary thanks go to my supervisor, Ian Gray, for the invaluable guidance that has made this research possible. I would also like to thank Leandro Indrusiak, Pengcheng Liu, and Alan Millard for the advice they have provide at various points throughout this PhD. Finally I would like to acknowledge my parents for their unwavering support.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for a degree or other qualification at this University or elsewhere. All sources are acknowledged as references. Some of the content in this thesis has been featured in published articles as follows:

- S. Signer et al., "Mixed-criticality wireless communication for robot swarms," in *9th International Workshop on Mixed Criticality Systems*, WMC, Dec. 2022
- S. Signer and I. Gray, "Adaptive application behaviour for robot swarms using mixed-criticality," in *Proceedings of the Third Workshop on Agents and Robots for reliable Engineered Autonomy*, ser. EPTCS, Sep. 2023, pp. 71–82. DOI: 10.4204/EPTCS.391

Chapter 1

Introduction

Swarm robotics is an emerging field with proposed application to a wide variety of domains, including search and rescue [3], agriculture [4], space exploration [5], and defence [6], but without major success in real world deployments [7]. Research project have yielded demonstrations of collaborative robot behaviours [8] but these do not yet implement practical applications. A number of commercial products that are sometime colloquially referred to as robot swarms do exist [9], for example as used by drone light show performances, but these implementations rely on centralised control or fixed infrastructure such that they are not considered pure swarm robotics systems in an academic context.

Early work on swarm robotics emerged from the study of insects behaviour with the aim of allowing multi-robot systems to reproduce insects' ability to collaboratively perform complex tasks from simple individual behaviours, referred to as "swarm intelligence" [10]. Existing literature describes swarm robotics as being robust, scalable, and flexible. Robustness in this context, however, refers primarily to the ability to continue despite the loss of individual robots. Prior work has shown that existing swarm robotics systems are not, in a more general sense "robust" in the presence of network faults. Under realistic communication models even simple swarm behaviours are prone to failure [11].

The traditional approach to designing swarm robotics systems involved either a "top-down" or "bottom-up" approach [12]. In the bottom-up approach the

programmer specifies the behaviour of an individual robot, with the overall behaviour being an emergent property of the interaction of these individuals. While this allows for some very elegant designs of simple behaviours, it is often very difficult to develop the individual behaviours that form more complex emergent behaviour. Swarm behaviours are therefore often developed using a trial and error process [13]. It may therefore not be fully understood how exactly the individual behaviours combine to form the emergent behaviour, and solutions may therefore be fragile when conditions change. The top-down instead starts from the desired overall behaviour and attempts to incrementally break this down into the individual behaviour. This approach can more easily handle complex behaviours, but avoiding centralised control typically requires communication of extensive global state information. These system are therefore particularly dependent on the performance of the underlying network [12].

Even where communication is vital to the correct operation of a swarm robotic system with defined requirements on message delivery time, existing research has long relied on best efforts protocols such as WiFi [14]. Despite improvements in WiFi technology over the years, modern WiFi systems still show poor latency performance in robotics systems [15]. While prior work has presented specialised real-time wireless protocols, including some operating over the 802.11 physical layer, these have not found broad deployment in robotics contexts. The basic WiFi networks typically used in robotics systems remain reliant on CSMA/CA collision avoidance with unbounded maximum latency [16]. The resultant difficulty in predicting network performance, combined with brittle application behaviour, therefore often makes it very difficult to design reliable swarm robotics systems.

Research in the real-time systems domain has studied the provision of timing guarantees for wireless networks even in the presence of interference. By using a real-time communication protocol, swarm robotics system could receive a priori guarantees as to the network behaviour. These can be useful not only as minimum service guarantees, but also as a target performance levels: prior work has shown that in some scenarios improved communication ability may result

in lower application level performance when emergent swarm behaviours receive too much information [17]. Restricting the ability of robots to communicate to exactly the level guaranteed by a real-time protocol may therefore help make the application level performance more predictable.

More recently, significant work in the real-time systems domain has focused on the mixed criticality systems model [18]. The mixed criticality model introduces the concept of criticality which typically defines the relative importance of system components such as tasks or message flows. This notion of criticality is different to the notion of priority, which is generally a scheduling property that broadly controls relative latency requirements. It is therefore not unusual that some tasks might be of high priority and low criticality or of high criticality and low priority. Mixed criticality systems typically have multiple criticality modes, where higher criticality modes drop or deprioritise lower criticality components. By switching to a higher criticality mode when some a priori defined condition is met, a graceful degradation of the system in which the most important components continue to function in the face of faults or unexpected conditions is possible. A priori analysis can take advantage of the modes to provide guarantees on system behaviour subject to the criticality modes tied to these mode switching conditions.

Adopting a mixed criticality approach to swarm robotics systems may therefore aid in the creation of more reliable designs in the face of faults. This thesis introduces the concept of criticality to swarm robotics systems starting at the network layer, then argues for adapting the definition of criticality modes such that they become more meaningful to the swarm applications.

1.1 Research Hypothesis

The research hypothesis posits that adopting a mixed criticality approach to swarm robotics system can improve reliability and predictability in the face of network faults. A simple introduction of criticality can be achieved by adopting a mixed criticality network protocol, but a tighter integration where criticality

modes are assigned application level meaning can provide greater improvements.

1.2 Thesis Outline

This chapter serves as a general introduction and motivation for the thesis. The following chapters are structure as follows:

- **Chapter 2: Literature Review** provides relevant background information from the real-time systems and swarm robotics domains.
- **Chapter 3: Mixed Criticality Communication** conducts an initial investigation into applying the concept of criticality to swarm robotics systems simply by adopting a mixed criticality communications model. Using a simple motivating problem and an existing mixed criticality wireless MAC protocol, it is demonstrated that applying criticality modes purely at the network layer can result in emergent mixed criticality behaviour at the application level.
- **Chapter 4: Adaptive Application Behaviour** argues that to achieve more robust application behaviour, a tighter integration of a swarm robotics system's network and application layers is required. By redefining the criticality mode of the network as an interface for this integration, it is demonstrated that this approach can allow an application to better adapt to different network conditions.
- **Chapter 5: Protocol and Timing Analysis** further redefines criticality modes to better suit swarm robotics applications by considering both message delivery and reception. Timing analysis is derived for a proposed mixed criticality real-time wireless MAC protocol using these redefined criticality modes such that application level guarantees can be provided for some simple behaviours.
- **Chapter 6: Conclusion.** The final chapter concludes by summarising contributions. The research hypotheses are revisited and directions for

future research are discussed.

Chapter 2

Literature Review

This chapter provides an overview of relevant existing literature. Section 2.1 summarises the mixed criticality systems model and instances of mixed criticality wireless networks. In section 2.2 approaches for handling changing node topologies in real-time wireless networks are discussed.

2.1 Mixed Criticality Systems and Networks

The mixed criticality task model, as first introduced in Vestal's 2007 paper [19], aims to enable better resource utilisation in CPU task scheduling problems by allowing the system to run in a potentially optimistic mode under typical conditions, but be able to switch to a pessimistic mode that guarantees only critical behaviour should the optimistic mode be exceeded. Due to the complexities of modern hardware, analytical methods for determining true worst case execution time estimates are typically expensive and pessimistic [20]. The mixed criticality model therefore allows tasks of lesser importance to use the available

Further development of the mixed criticality task model into the model presented for AMC analysis ensures that overruns by LO criticality tasks cannot impact the correct timing behaviour of HI criticality tasks, without needing to validate LO tasks to the same degree as HI tasks [21]. For safety critical systems this allows certification of an overall system with tasks of varying criticality co-located on the same hardware without needing to certify LO tasks to the

higher safety integrity levels or design assurance level that standards require.

The mixed criticality model has since been applied to a number of other application domains [18]. Prior work has studied the application of the mixed criticality model to wireless networks in order to handle the inherent unreliability of wireless links. Addisu et al. [22] considered using criticality levels to allow adaptive quality video streams over a fixed IEEE 802.11 network with specific topology depending on available network bandwidth measured at runtime.

Subsequent work [23] by Jin et al. used mixed criticality analysis for the WirelessHART protocol that has been widely adopted for industrial systems. Rather than handling transmission faults, this work assumes that it is the inter-packet period that is shorter at higher criticality levels. A continuation of this work then studied using redundant paths for critical messages when application exceptions occur [24]. Both of these works retain a network manager used by the WirelessHART protocol to coordinate the network.

AirTight [25] is a real-time mixed-criticality wireless MAC protocol that uses a fault model per criticality level to bound the number of retransmissions that are required due to interference. The key discovery behind this protocol is that it is possible to exploit the equivalence between link scheduling and single core CPU scheduling to build a fully decentralised hard real-time wireless MAC protocol with accompanying response time analysis that is analogous to that of fixed priority single core CPU scheduling. A predefined slot table mediates access to the wireless medium, after which each node makes local scheduling decisions as to which frame should be transmitted during an assigned transmission slot. This protocol is of particular interest since, unlike many other hard real-time protocols, it neither requires an internally selected leader node nor an external network coordinator.

2.2 Node Mobility and Network Topology

Extensive prior work has considered application of real-time wireless networks to domains such as wireless sensor networks and industrial control. Protocols

such as WirelessHART are able to provide reliable timing behaviour for these systems and have found real-world adoption in industry. A particular challenge for networks of swarm robotics systems as compared to industrial systems is in coordinating the network and handling the continuous topology changes caused by node mobility.

In industrial control systems, it can often be assumed that large parts of the network are permanently installed, with a most a small number of moving nodes. Protocols such as WirelessHART use a dedicated centralised network manager that collect global network topology information in order to create and distribute schedules. While this results in reliable links for static networks, it is not able to quickly adapt to changing topologies, resulting in poor performance in scenarios featuring mobility [26]. These protocol are therefore typically not of particular interest to possible swarm robotics deployments with ad-hoc networks which cannot rely on centralised network managers and feature constant topology changes.

More recently, following the 2011 introduction of Glossy [27], there has been significant research interest in protocols that use synchronous transmissions to handle changing topologies [28]. Glossy allows nodes to perform multi-hop communication without any knowledge of the underlying network topology by efficiently flooding packets to all nodes in the network. Network flooding is traditionally expensive as it typically results in significant network contention. Previous protocols such as Flash[29] attempted to facilitate floods through simultaneous transmission by multiple nodes by exploiting the capture effect. While this is effective in some scenarios, it depends on many dynamic factors such as relative timing of transmissions and received signal strength at different nodes in the network topology. For real-time applications in particular it is therefore unlikely to be suitable due to the difficulty in establishing meaningful timing guarantees when allowing such transmissions. Glossy fundamentally changes this calculation by allowing nodes using the IEEE 802.15.4 O-QPSK physical layer to simultaneously broadcast the *same* frame from multiple nodes. With sufficiently tight (i.e. sub-microsecond) time synchronisation this is demonstrated to results

in constructive rather than destructive interference, thus making it much cheaper to flood a frame through the network.

At the start of a Glossy flood, a single node acts as the initiator while other nodes are in receive mode. All nodes that receive a frame immediately change their radio to transmit mode and, after incrementing the frame's relay counter, rebroadcast the received frame. Careful construction of this mode change, accounting for interrupt delays and hardware variation, ensures the time taken for the mode change is deterministic and equal across nodes. Receiving nodes will therefore all simultaneously transmit the received frame. After transmitting a frame, nodes switch back to receive mode such that they can receive the rebroadcasts of other nodes, and thus transmit again while maintaining time synchronisation. By comparing internal clocks to the timestamps of received frames and the corresponding frame relay counters, Glossy also provide time synchronisation between nodes at no further cost.

The Low-Power Wireless Bus (LWB) [30] builds a complete network protocol that purely uses Glossy network floods. A centralised controller node distributes a global communication schedule that assigns transmission slots to nodes such that at most one node initiates a Glossy flood during any transmission slot. A dedicated contention slot is included in the schedule in which nodes can attempt to request to be given additional transmission slots by the controller in future schedules, with the controller relying on the capture effect to receive one of these requests should multiple nodes make simultaneous requests (as is expected during bootstrapping). Since the underlying Glossy floods are delivering all messages to all nodes, LWB requires no knowledge of the network topology and is thus, unlike a majority of previous protocols, able to handle node mobility without any performance loss.

LWB does not consider message response times, making it unsuitable to real-time applications. The Blink protocol [31] replaces the centralised controller used by LWB with an EDF scheduler that is aware of timing requirements, and thus suitable for real-time applications. Since LWB and Blink construct schedules at runtime based on dynamically declared stream requirements, significant effort

is involved in constructing the scheduler in an efficient manner such that is can run on resource constrained wireless nodes. Blink does not attempt to address temporal faults in message delivery besides noting the high packet delivery rates that can be achieved with Glossy and asserting that the intended application can handle small amount of packet loss.

A different approach to real-time scheduling using Glossy is taken by TDMH-MAC [32], which uses the relay counter of Glossy frames to allow nodes to deduce information about the network topology. The protocol bootstraps by sending a Glossy flood from a centralised controller node. On receiving this frame, nodes store the observed relay counter value and share this with neighbouring nodes. This allows nodes to route frames towards the controller without global knowledge of the network topology simply by forwarding frames to nodes advertising a lower relay count. Nodes use this routing ability to send local topology information to the controller, which builds a global view of the network topology using the local information from each nodes. The controller then generates a global schedule of point-to-point links which is flooded to all nodes. While this allows a TDMH-MAC controller to more efficiently gather topology information than other protocols, it is not designed for the rapid topology changes caused by frequent node mobility.

RT-WMP[33] is a protocol that attempts to provide hard real-time traffic over multi-hop IEEE 802.11 based networks by using a token-passing scheme compromised of three phases: the Priority Arbitration Phase, in which a token is passed between nodes to determine which node has the highest priority message, the Authorization Transmission Phase, in which this node is informed that it should transmit, and the Message Transmission Phase, in which this highest priority message is transmitted. Each node must maintain a local link quality matrix that is used to determine which nodes can communicate directly and how multi-hop messages should be routed. While the original RT-WMP proposal only allowed for point-to-point communication, a later extension [34] enables multicast messages to piggyback on other RT-WMP frames.

The RT-WiFi protocol[35] is another protocol that enables predictable timing

behaviour for networks running over the 802.11 physical layer. An RT-WiFi access point adapts the WiFi point coordination function such that it can be used to impose a TDMA schedule for access to the wireless medium. While the initial presentation of the RT-WiFi protocol was limited to start topologies, later work considered multi-cluster networks[36], though it is still limited to infrastructure based networks.

2.3 Summary

Existing literature has investigated mixed criticality wireless networks and networks able to adapt to changing network topologies separately, but there does so far not appear to exist protocol that provide both. Neither existing mixed criticality protocols nor existing networks handling rapidly changing topologies are directly aimed at swarm robotics applications, and do therefore not map directly onto the network requirements of these systems.

Prior work has considered adapting robot behaviour according to various link quality metrics such as RSSI or LQI. Tardioli et al. [37] present an application of a spring-damper system that allows robot motion to adjusted according to link quality metrics. While metrics such as RSSI or LQI can provide an indication of link quality, they are inherently imperfect metrics for predicting future network performance. The interpretation of specific values often depends both on the environment and the specific hardware used, making it more difficult to develop generally applicable robot designs. More recent work has considered building more complex models on top of these link quality metrics, for example by using machine learning approaches [38], but fully capturing the properties of a wireless link remains difficult. The mixed-criticality approach is advantageous because it can react to the faults occurring in the delivery or reception specific messages. Since ensuring this communication is the actual obejctive, reacting directly to these faults removes the uncertainty introduced by attempting to model complex physical systems.

Chapter 3

Mixed Criticality Communication

In recent years the mixed criticality systems model has been adapted for use in wireless communication protocols. Existing research in the swarm robotics domain has not used timing aware communication and has often made unrealistic assumptions on network behaviour, such as assuming perfect communications within a given range or that a basic WiFi network will always allow communication between all robots. As a result, swarm robotics deployments are often fragile under changing network conditions. Best efforts protocols such as standard WiFi networks used in these experiments are unable to provide guarantees, making it difficult to analyse system behaviour ahead of time, particularly in respect to partial failure conditions. Extensive prior work in the real-time networks domain has developed a large number real-time wireless network protocols that can guarantee network performance to varying degrees subject to either implicit or explicit constraints on radio conditions. Mixed criticality protocols can provide guarantees at different criticality levels as defined by a fault model, allowing for graceful degradation in the presence of failures. Despite this work, adoption of these protocols within swarm robotics research remains very limited. This chapter conducts an initial investigation into introducing mixed criticality communication to swarm robotics systems by utilising the existing AirTight protocol.

3.1 Illustrative Example ("Circle Problem")

To demonstrate the effect of introducing the concept of criticality at the network layer of a swarm robotics system, a simple illustrative example, henceforth referred to as the "circle problem", is contrived. A group of autonomous wireless robots are arranged in a circle formation as shown in figure 3.1, and are given two tasks: T_{POS} and T_{LED}. Task T_{POS} requires the robots to maintain the circle formation while increasing the circle's radius. Due to the initial orientation of the robots, this simply requires the robots to drive radially outwards at the same speed. Task T_{LED} requires the robots to display a common LED colour. The "correct" movement speed and LED colour to be applied to all robots is determined in a decentralised manner: each robot may change the LED colour or movement speed for the entire swarm so long as the time since it last initiated such a change exceeds some predefined threshold. When a robot chooses a new LED colour or forwards movement speed, the new value and the time at which the change is to take effect must be communicated to all robots in time for it to be applied simultaneously (subject to clock-synchronisation accuracy).

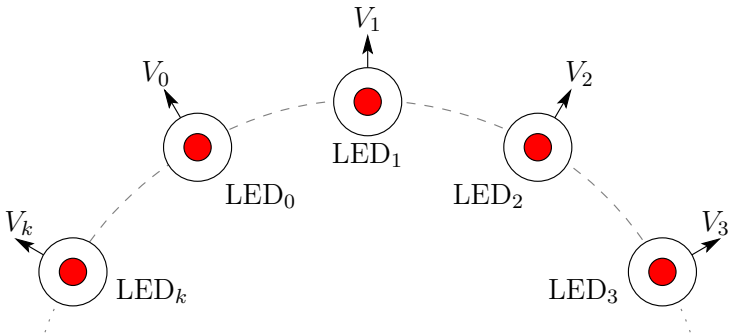

Figure 3.1: The "circle problem": k robots, each having a velocity V and an LED colour, must satisfy tasks T_{LED} and T_{POS}. Task T_{LED} requires all nodes to show the same colour, while task T_{POS} requires all nodes to move with the same velocity. At any given time, both conditions should be satisfied.

These two tasks are associated with corresponding error metrics E_{POS} and

E_{LED}. Positional error E_{POS} is defined as the maximum difference in effective circle radius (i.e. the distance from the original implicit circle centre) of any two robots, as shown in figure 3.2. LED Error E_{LED} is defined as the number of nodes showing an incorrect LED colour, where the correct LED colour is defined as the colour that would be displayed by a robot that immediately received all colour change operations generated by all robots. It is clear that if robots had instant perfect communication, it would be trivial to maintain a zero value for both error metrics. Arbitrarily, task T_{POS} is defined to be of greater importance, and thus the message flows for task T_{POS} are assigned a higher *criticality* than those of task T_{LED}.

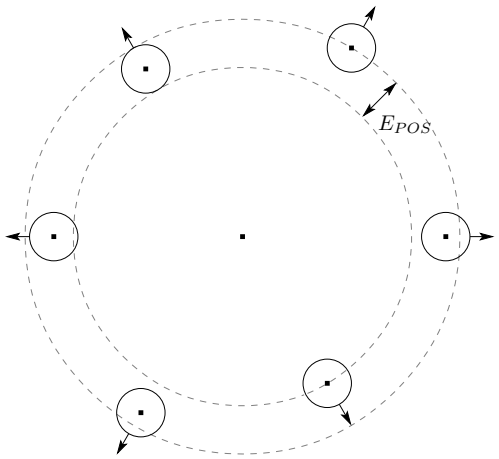

Figure 3.2: Circle formation showing position error E_{POS}, the maximum different in distance of any two nodes from the circle's original central point.

Note that this example is not intended to directly represent a real-world application, but rather to allow an exploration of robot behaviour in the face of challenges that are reflective of those that would be faced by real multi-robot and swarm robotics systems. The notion that robots must simultaneously perform multiple tasks that each have differing levels of importance and potentially conflicting requirements is one that is present in many proposed applications. Requiring robots to maintain connectivity while maximising area coverage and responding to collected data is, in particular, a pattern that has many potential applications in domains such as surveillance and search & rescue. In this example,

in which robots are programmed to move ever further apart, leading to ever lower packet delivery rates, communication between the robots will eventually break down regardless of the chosen MAC layer. The purpose of the evaluation is therefore to explore how behaviour on the brink of this failure is affected by the network MAC, and thus how these applications can be made more resilient in the face of challenging network conditions.

3.1.1 Experimental Instantiation

A concrete instantiation of the problem is created by arranging set of 6 mobile nodes with multicoloured LEDs in a circle facing outwards with an initial radius of 1m and pre-programmed with a default initial LED colour. The LED colour and movement speed is determined the decentralised manner as described above. The minimum inter-arrival period of each robots LED colour changes is configured to one second. When a robot proposes an LED colour change, the swarm must coordinate to ensure all robots change to the new colour within two seconds. The proposing robot is required to transmit a colour change message to all other robots. The colour change message consists of two fields: the new LED colour, and the time, encoded as a specific future TDMA slot, at which the colour change should take effect. Multiple colour changes proposed by different robots can be queued to take effect at different future times: robots store received colour change messages in a data structure sorted by activation time, checking if the timestamp of first entry has been reached yet on each radio slot. Should two robots propose an LED colour change to take effect at the same time instant, the conflict is deterministically resolved by a predetermined static priority ordering. The movement speed for the swarm is determined in an equivalent manner, except that these events may, per robot, only be generated once every five seconds and take effect after ten seconds. Each wireless frame transmission is assumed to be large enough to contain either exactly one LED colour change or exactly one movement speed change.

In a real application, these changes would be triggered by local sensing onboard the nodes, potentially in combination with other shared information.

For the purposes of this simulation, it is simply assumed that after the minimum inter-arrival period has been reached, there is a uniform probability of 2% per (10ms) simulation step for LED colour changes and 1% per simulation step for movement speed changes. New LED colours are chosen as a uniformly random RGB value, whereas movement speed is chosen such that each robot has a bias towards selecting a faster or slower speed such that positions will diverge of time if communication fails.

In order to observe the robot behaviour under controlled and reproducible conditions, the experiment is run using the established ARGoS [39] swarm robotics simulator. The ARGoS simulator features integrated support for radio communications but only models these as perfect communications within a given range and does not handle packet collisions. The simulator is therefore extended by a custom plugin that allows TDMA networks to be simulated at the slot level. This plugin assumes that each simulation step is equivalent to one transmission slot, such that each node can only send or receive a single frame during a simulation step. If a robot were to "receive" multiple frames within a single slot, this is defined to result in destructive interference such that no frame can be successfully decoded. The plugin supports configurable transmission models that define whether other nodes receive transmitted frames; the transmission model for these experiments is described in section 3.1.2.

The experiment compares E_{POS} and E_{LED} over simulation runs of 240 seconds for four different configurations:

AirTight

While AirTight [25] is not designed for swarm robotics applications, it is one of the few existing mixed criticality wireless MAC protocols that is not orchestrated by a centralised controller. An AirTight deployment uses a priori analysis of the flows and topology in order to ensure that system is schedulable. Access to the wireless medium is split in 10ms TDMA slots, with a periodic slot table defining a single node which is permitted to transmit during any given slot. In the example, this slot table is configured with equal length to the number of

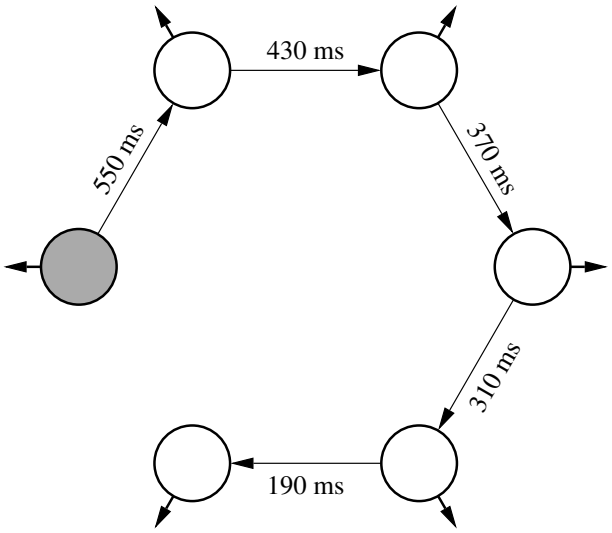

Figure 3.3: Ring topology routing of a message initiated by the shaded (leftmost) node, showing configured per-hop deadlines for LED messages.

nodes with each node assigned a single unique transmission slot.

Since AirTight only supports point-to-point message delivery, while the circle implementation requires message delivery to all other nodes, the AirTight implementation is configured to route messages through the swarm using a ring topology. The initiating robot queues the LED or colour change message to be transmitted to the next node in the ring. Upon receiving such a message, the recipient robot re-queues the message in a dedicated forwarding buffer per initiating robot such that message is delivered to the next robot in the ring. An example of the routing is shown in figure 3.3.

Since the instantiation assumes that movement should be treated as more important the colour synchronisation, LED flows are assigned LO criticality while movement instruction as assigned HI criticality. Conversely, due to the shorter end-to-end deadlines of LED changes compared to movement changes, LED changes are assigned higher fixed priorities than the movement changes. The overall LED flow is partitioned with manually selected per-hop deadlines of 550ms, 430ms, 370ms, 310ms, and 190 ms (giving a maximum end-to-end latency of 1850ms). Similarly, the movement flow is partitioned with manually selected

Name	Recipient	Deadline	Period	Criticality
"led-2"	N_1	190ms	1s	*LO*
"led-3"	N_1	310ms	1s	*LO*
"led-4"	N_1	370ms	1s	*LO*
"led-5"	N_1	430ms	1s	*LO*
"led-0"	N_1	550ms	1s	*LO*
"mov-2"	N_1	1570ms	1s	*HI*
"mov-3"	N_1	1750ms	1s	*HI*
"mov-4"	N_1	1870ms	1s	*HI*
"mov-5"	N_1	1990ms	1s	*HI*
"mov-0"	N_1	2110ms	1s	*HI*

Table 3.1: Example of configured message buffers for node N_0, ordered by assigned fixed priority level. Note that the number in the buffer name corresponds to the ID of the original initiating robot of changes queued in that buffer.

per-hop deadlines of 2110ms, 1990ms, 1870ms, 1750ms, and 1570ms (giving a maximum end-to-end latency of 9290ms). An example of the resulting message flows is shown in table 3.1. These per-hop deadlines were selected such that they match the response time bounds computed by a priori AirTight timing analysis using an assumed fault model (discussed below). Since the resulting end-to-end deadlines are within the timing requirements specified in the beginning of this section, it is guaranteed that the task error metrics will remain zero unless the transmission fault model is exceeded.

In order to provide timing guarantees, the AirTight implementation requires a fault model that bounds the maximum number of faults that can occur within a given time period. For the purposes of this experiment, a simple deterministic fault model is defined that bounds the maximum number of faults that can occur within a given time period t for a given critical level L.

The original AirTight specification considered faults to occur as a result of external interference causing blackout periods. In this experiment, the fault models are instead used to handle mobility despite the static topology by allowing

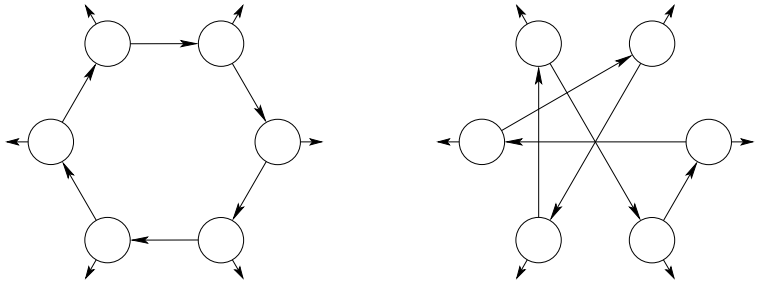

Figure 3.4: Node setup showing optimal routing (left) and an example of possible randomised routing (right).

a reduced packet delivery rate caused by distance between nodes. The AirTight fault bounds are modelled using a simple binomial distribution. For a given criticality level L, an assumed minimum packet delivery rate at that criticality level ($\widehat{\mathrm{PDR}}(L)$) and a desired confidence bound, the fault model $F(L,t)$ computes a maximum number of failed transmission slots as the smallest integer m satisfying the following inequality:

$$\sum_{k=0}^{m} \binom{t}{k} \cdot \left(1 - \widehat{\mathrm{pdr}}(L)^2\right)^k \cdot \left(\widehat{\mathrm{pdr}}(L)^2\right)^{t-k} \geq \mathrm{conf}(L) \tag{3.1}$$

Note that the PDR is squared in the above inequality to account for lost acknowledgements, which are (pessimistically) assumed to occur with the same probability as a lost data frame. It is important to note that this binomial distributions is simply a model used to establish an estimated fault bound over a given number of transmissions. The confidence bound used to create this model is therefore only relevant as a metric for faults occurring in a given window, and does therefore not define a probability of the fault model or computed response times being exceeded.

Note that an optimal implementation of this ring topology requires a priori knowledge of the network topology such that robots are sending messages to neighbour nodes and not across the circle formation. In order to observe the behaviour with and without the inherent advantage posed by this topology information, the protocol is tested in both an optimal routing configuration and a configuration where the ordering of the robots has been randomly shuffled, as

show in figure 3.4.

Broadcast

With this communication scheme nodes simply broadcast each message a fixed number of times, using carrier sensing to reduce collisions. For this experiment each message is broadcast 13 times, since this results in the maximum possible transmissions without exceeding the available bandwidth.

Point-to-point

Nodes transmit each message to each other node in turn. Node use a CSMA/CA like approach, using carrier sensing and random backoff between retransmissions until an acknowledgement is received or a maximum number of retries has been reached. The maximum number of transmissions attempts is set to 5 for this experiment, which was experimentally determined to suitably balance the need for retransmissions with the need to avoid excessive collisions or a buildup of frames in the transmission buffers.

3.1.2 Transmission Models

Simulation of the experiment requires two models: a transmission model which defines the which transmissions are successfully received by which other nodes, and, for the AirTight implementation, LO and HI criticality fault models. Note that even though these model similar properties, they are two fundamentally different models. The transmission model is a property of the simulation while the fault model is a configuration of the implementation. While it may seem tempting in a simulation environment to define the two models together using knowledge of the other, and this would allow the appearance of very powerful guarantees, this would be simply be an exercise in defining the problem away with no real world relevance. A real world implementation must be able to handle the uncertainty from being unaware of true transmission propagation behaviour.

For the purposes of this experiment the transmission model is defined to assume the effective packet delivery rate of a link is fixed at 99% for distances

of less than 1.5m, and inversely proportional to the square of the distance between two nodes with greater separation. The transmission model assumes that successful or unsuccessful delivery is determined independently for each link and transmission. This is a very simplistic model that does not attempt to capture the true complexity of wireless links. Previous experiments have shown that real-world correlation between node distance and packet reception rates is weaker and highly dependent on the specific testing conditions [40]. Baccour et al. describe the communication range of a robot as consisting of distinct connected, transitional, and disconnected regions. A short distances experimental results show uniformly high packet delivery rate. As the distance increases, there is a transitional region where packet delivery rates and node distance are only weakly correlated. Finally, one the distance exceeds a further threshold the nodes enter a disconnect region with very low packet delivery rates. The relative size of these regions depends strongly on the environment: for example the experiments of Baccour et al. show a connected region of 7m indoors but only 3.5m in an outdoor environment.

The simple transmission model used in the simulation arguably broadly captures a connected, transitional, and disconnected region, though the transitions between these regions are more gradual in the simulation than the real-world results. A further significant constraint of the this simulated transmission model is its inability to model temporal or spatial correlation. The experiment is however not designed to rely on specific properties of the fault model, and is instead intended as a demonstration of the utility of a mixed criticality model.

3.1.3 Results

Results for error metrics E_{POS} and E_{LED} are shown in figures 3.5 and 3.6 respectively. The results for E_{POS} show that none of the protocols caused positional errors after 30 seconds. When using the broadcast or point-to-point protocols some instances had however accumulated positional errors by the 60 second point, and all instance had accumulated positional erorrs after 120 seconds. The AirTight implementation were able to avoid positional errors for longer; with

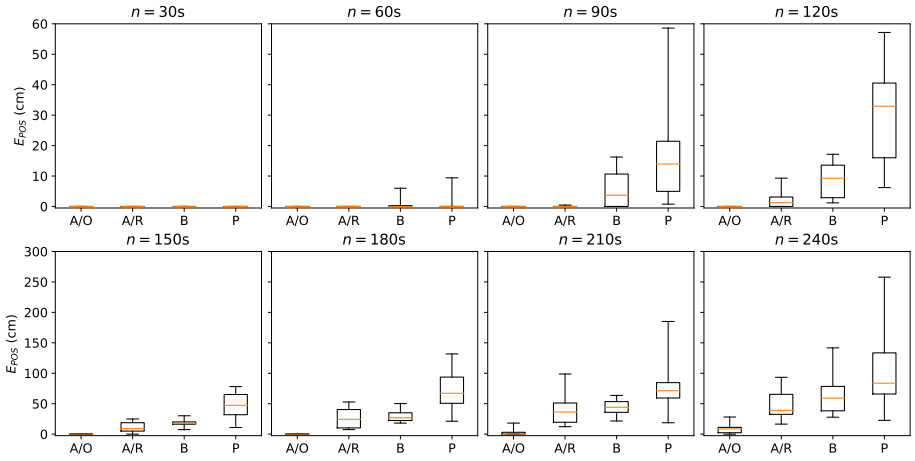

Figure 3.5: E_{POS} after n seconds for simulated nodes using the AirTight protocol with optimal (A/O) and randomised (A/R) routing, Broadcast protocol (B), and Point-to-point protocol (P) over 10 different random seeds. Note that the y-axis scaling changes between rows.

the advantage of knowledge of the network topology the AirTight implementation with optimal routing was able to avoid any positional errors until the after the 180 second mark. Note that by this point the inter-node distance has increased to values greater 3m distance foreseen by the fault model. Positional errors that occur after this point are therefore a result of the fault model being exceeded rather than violation of the timing guarantees provided by the protocol. Without knowledge of the network topology, the AirTight implementation with randomised routing resulted in very minor positional error after 90 seconds in the worst case, and showed errors in the average case after 120 seconds.

For the lower criticality flow, both comparison protocols showed an initial LED error after approximately 30 seconds in some instances, with the first instance of a median error greater than zero after 60 and 100 seconds for the point-to-point and broadcast protocols respectively. The AirTight implementation with optimal routing is able to avoid any LED errors for over 50 seconds, with a non-zero median first occurring after 90 seconds. The implementation with randomised routing fares significantly worse, encountering errors almost immediately and regular errors after 40 seconds. Note that since the fault model is probabilistic, the

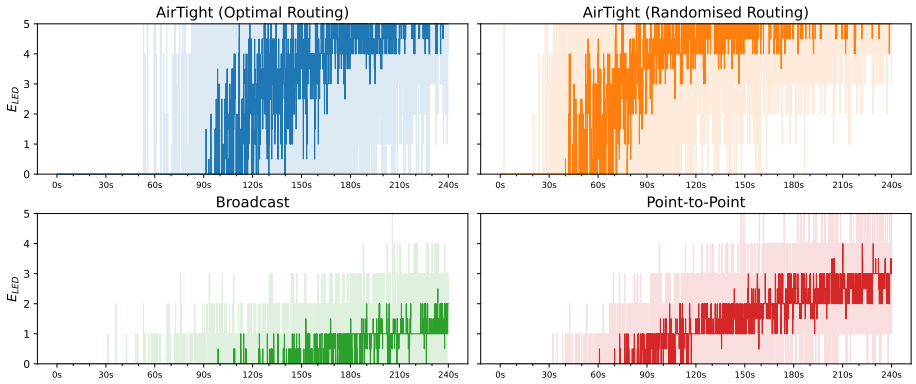

Figure 3.6: Median E_{LED} after n seconds over for 10 different random seeds. Shaded area shows range from minimum to maximum E_{LED} value at the given step over the runs.

immediate error is still as a result of the fault model being exceeded rather than an violation of the protocol's timing guarantees. The AirTight implementations are thus able to provide improved performance for task T_{POS}, at the cost of degraded performance for task T_{LED}. Particularly in the later steps of the experiment the AirTight implementations show significant LED errors since this traffic is being discarded. Using a mixed criticality communication protocol therefore allows the system designer to priorities the motion control component in the presence of networks errors.

3.2 Flocking Application

The "circle problem" presented in section 3.1 demonstrates the advantages of a mixed criticality communications model in a contrived scenario. In this section a flocking application is presented in order to demonstrate the advantages of the mixed criticality model in an application scenario with more immediately recognisable real-world utility. In this scenario a group of 10 robots must communicate in order to form and move as a flock. Each robot is assumed to know its own position and orientation through some localisation system. Nodes should transmit their location and orientation to all other nodes every five seconds, such that each robot can adapt its own velocity to remain part of the

flock [41]. It is further assumed that each robot is performing some environmental sensing task such that is generates up to three data frames per second that should be delivered to an a priori designated sink node[1]. It is again assumed that the mobility control is the more important task, and so the flows transmitting the positional data to each other node are defined as *HI* criticality flows. The environmental sensing task is deemed to be less important and so it defined as *LO* criticality.

3.2.1 Experimental Setup

The experiment is run using a very similar ARGoS simulation to the previous section. The simulation records the number of data frames received by the sink node and the sum of the robot velocity vectors at each time step. Since these velocity vectors add destructively if robots are moving in different directions (i.e. not as a flock), this value serves as a proxy for the stability of the flock that can be . The AirTight implementation uses the same fault model as in the previous section, but here defines point to point flows from each node to each other node since there is no a priori knowledge of the network topology that can be exploited. The broadcast implementation is reconfigured with the maximum number of retransmissions reduced to 3, since this is the maximum value that still ensures available bandwidth is not exceeded. The configuration of the point-to-point implementation is unchanged.

3.2.2 Results

Figure 3.7 shows the effective flock movement speed averaged over the simulation run for different transmission model scaling factors. All the communication protocols show similar flock movement speed when no further scaling of the packet delivery rate is applied. When the transmission model is scaled by a factor of 0.8, 0.6, or 0.4, however, the performance of the AirTight implementations remains

[1]The a priori designation of a single data sink node is somewhat inelegant from a swarm robotics viewpoint, but limitations of the communications protocols make it awkward to present a more distributed behaviour.

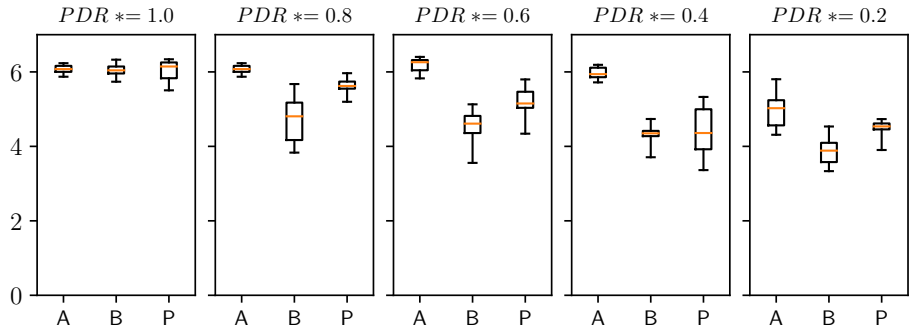

Figure 3.7: Mean length of summed node velocity vectors at each time step over 10 simulation runs with different random seeds when PDR is modified by a constant factor from 1.0 to 0.2.

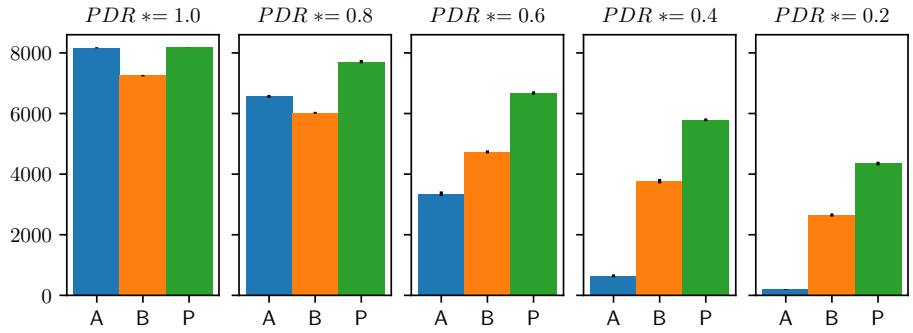

Figure 3.8: Mean number of data samples received by the designated sink node over 10 simulation runs (error bars show min/max value).

broadly unchanged while the implementations using the broadcast and point-to-point protocols show degraded performance. Dropping the transmission model scaling factor to 0.2 causes degraded performance for all three communication protocols.

The simultaneous performance of the data collection task is shown in figure 3.8. These results show the mixed-criticality tradeoff made by the AirTight implementation: the number of received data samples drops rapidly when the packet delivery rate is decreased. With a packet delivery rate scaling factor of 0.4 very few data samples are delivered, since there are very few data sample transmission left to discard, this explains why a further drop in packet delivery rates causes a degradation in the flocking behaviour. Note that the broadcast protocol implementation

shows a comparatively lower number of data samples collected when no scaling factor is applied to the transmission model. This is caused by the lack of timing guarantees on the protocol, which means that some data samples remain in message transmission buffers awaiting delivery at the end of the simulation.

3.3 Summary

This chapter studies the application of mixed criticality communications in a swarm robotics context. Assigning criticality levels to communication flows purely at the network layer is shown to transfer into mixed-criticality behaviour for associated application behaviours in both a contrived and more plausible application scenario. By allowing graceful degradation of network dependent robot behaviour in the face of network faults, the important components or a robots behaviour can be made more robust. The AirTight protocol used in this evaluation is, however, not particularly suited to swarm robotics applications leading to limitations in the example applications. The requirement to define network topology in advance, and the limitation of only send one to one messages does not satisfy the requirements of most swarm robotics systems.

Chapter 4

Adaptive Application Behaviour

While swarm robotics systems can, as shown in the previous chapter, exhibit application level mixed criticality behaviour simply through adapting criticality modes at the communication layer, this chapter argues that a closer integration between the application and the network components could allow the system to better adapt to changing network conditions. It is clear that the application component will, either directly or indirectly through some other layer, place requirements on the network component, for example simply by requesting the delivery of some message. In order for the network component to be able to satisfy the requests made by the application, the network in turn has some requirements that must be satisfied. The circle problem in the previous chapter demonstrated how action by the application (directing nodes to move further apart) can affect the network such that the applications own communication requirements can no longer being met. The application should therefore not consider its own behaviour in isolation, but also consider the effect of its actions on its own communication needs. Since radio performance is highly complex [42] and dependant on a multitude of factors, it is typically not feasible to model this impact ahead of time and so it can only be judged from observed network performance at runtime. In traditional systems models, however, the network component has limited ability to communicate back to higher layers. Depending on the network protocol in use, it may not be possible to determine if messages are being delivered. While some

protocols can provide acknowledged delivery of messages, and it is often possible in practice to retrieve information such as received signal strength of incoming frames, it often remains difficult to determine what these mean in terms of an application's higher level requirements and therefore in turn what action, if any, the application should take. This chapter therefore proposes the use of criticality modes as an interface between the network and application components such that the application can detect failures at the network level.

In the task scheduling domain, a key motivation for developing the mixed criticality model was allowing tasks of differing importance to be scheduled on the same system without needing to validate lower criticality tasks to the same level as higher criticality tasks [21]. For many such mixed criticality systems the criticality mode is therefore primarily a feature of the scheduling analysis, i.e. it is often not actually expected that the systems will ever switch criticality mode at runtime [18]. This chapter instead argues that a swarm robotics system could be designed such that mode changes are not only expected, but that the applications reaction to these criticality mode changes might be integral to the overall system behaviour.

Due to the lack of exisiting mixed criticality wireless network protocols that are suitable for swarm robotics scenarios, this chapter introduces a prototype protocol such that a network criticality level can be exposed to a swarm robotics application. This prototype protocol adapts AirTight to use broadcast rather than point-to-point transmission, piggybacking group acknowledgements onto future transmissions.

4.1 Motivating Problem

This chapter considers a swarm exploration system in which a swarm of robots should collaborate to explore an unknown area. The to be explored area is partitioned into a two-dimensional grid in which robots should visit each cell and determine whether or not there is an object present. Using this information, robots each build a map of clear or occupied cells. Objects are assumed to be

stationary such that it is unimportant when a cell is visited or whether multiple robots visit a cell. Robots are able communicate the state of explored cells over the networks allowing other robots to insert this value into their own map without needing to visit the cell.

A concrete instantiation is derived by considering a set of 10 Pi-Puck [43] robots exploring a 6x6 meter grid of 10x10 cm square cells. Pi-Puck robots only have simple infrared range-finder sensors that can determine the distance to the closest object within a short range, but are unable to determine the nature of any detected object. Therefore, to avoid falsely detecting other robots as an obstacle in the environment, robots must maintain a minimum separation. Each robots is assumed to have access to additional hardware that allows it to determine its current location and orientation. In order to avoid the near-collisions that cause false detections, robots must communicate their current position to other nodes. It is assumed that robots will not be able to store their entire location and object detection history, such that robots are unable to retrospectively correct any such errors. The position messages are therefore intuitively subject to soft real-time constraints, since robots must learn the positions of other robots before the minimum separation distance is violated.

The exploration implementation is loosely inspired by an existing algorithm [44], but adapted to the much simpler Pi-Puck robots and greatly reduced communication ability. The exploration behaviour is implemented by robots picking and driving towards a target cell, which is always chosen as one of the nine cells within a three by three grid centred over the robot's current position. A new target cell is chosen once the target cell has been reached, or the robot encounters an obstacle in the target cell. The new target cell is selected as the cell for which the sum of the following weights results in the smallest value.

- Diagonal movements are assigned a weight of +1.
- The cell the robot is currently in is assigned a linearly increasing weight the longer it remains in that cell, and the cell it was in immediately previously is assigned a weight of +1.

- An avoidance score of +1000 if the cell is known to contain an obstacle.
- An attraction score of -10 if it is an unexplored cell.
- A separation score of 400000, 200000, 100000, 4, 1, 0.25, or 0.1 respectively if the distance to the closest target cell of another robot, counted as a number of cells, between 0 and 6.
- An alignment score, given by the dot product of the robot's forwards vector and the vector from it's current position to the potential target.
- An attraction score equal to the distance to the closest reachable unexplored cell, counted as a number of cells.
- Optionally, a cohesion force of $8d^3$, where d is the distance to the computed centroid of all robots using the most recent position information the robot has received. This force is applied according to the rules defined in Section 4.2.

4.2 Robot Cohesion

A key parameter of the robot behaviour is the cells selection cohesion element, as shown in figure 4.1. Without a cohesion element applied, robots disperse throughout the environment, greedily exploring new cells. In contrast, when a cohesion element is applied, the robots move as group. Under perfect communication, robot cohesion reduces the overall application performance. This can be intuitively understood by considering the robots that disperse maximally will not block each other, giving robots maximal freedom to select unexplored cells. The closer robots are pulled together, the more often the minimum separation constraint may effect robots, for example by requiring them to return to already explore cells.

With a communication model that deteriorates with distance, the application performance decreases as the distance between robots increases, since sensing information from other robots is lost. In extremis, this could effectively require

each robot to individually explore the entire area. There is therefore a trade-off in which some amount of cohesion is useful to preserve communication, but too much cohesion decreases performance by reducing the exploration ability. If the application designer knows the properties of the wireless medium in advance, then an ideal level of cohesion can be assigned. In reality, however, changing environmental factors can affect message propagation such that the exact message propagation behaviour cannot be known a priori. As a result, an application designer more choose a sub-optimal level of cohesion resulting in degraded application performance.

Using the criticality mode of the network as an interface between the network and application components of a swarm robotics system instead allows the robots to adapt to the conditions encountered at runtime. When the network is in LO criticality mode the robots can therefore disperse to maximise their exploration potential, but a cohesion force is reintroduced to pull the nodes back together should the network switch to HI criticality mode. Once the network has recovered it switches back to LO criticality mode so that robots can resume exploration. While at any given moment robots are in either HI or LO criticality mode, so cohesion either is or is not applied respectively, the effective cohesion force is determined by the average criticality mode over time. The resulting equilibrium between time spend in LO and HI criticality mode therefore allows robots to effectively adapt to the encountered conditions. In the following simulated experiments the following four cohesion weightings are explored:

- No cohesion: The cohesion weight is completely disabled.
- Constant cohesion: The cohesion weight is always applied.
- Half cohesion: The cohesion weight is always applied, but is computed using half of the true centroid distance.
- Mixed criticality: The application applies a cohesion force only when the network protocol has been in HI criticality mode at some point within the last three seconds.

Note that in mixed criticality mode the cohesion force is applied not only when the network protocol is currently in *HI* criticality mode, but also when the network has recently been in *HI* mode. This helps to ensure that the application applies the cohesion force for a minimum amount of time, even if the network can very quickly switch back to *LO* mode. Together with the alignment score that otherwise discourages robots from sharply changing direction, this additional period in which the cohesion force is applied helps to reduce oscillatory behaviour that might otherwise occur. There exists a tradeoff as to the configuration of this duration: shorter values result in oscillation on a single robot level where a robot spins on the spot as it switches between *LO* and *HI* criticality mode, while a longer duration might lead to swarm scale oscillation as robots move in and out of formation. This oscillation is a property of the application

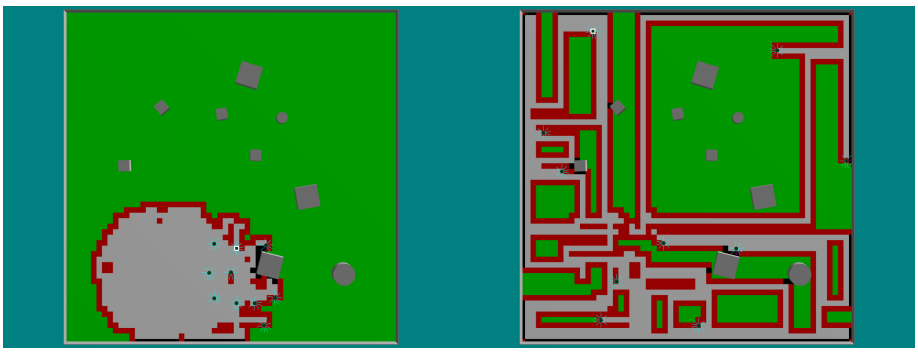

Figure 4.1: Visualisation of exploration state after 95s when applying constant cohesion (left) compared with no cohesion (right) under perfect network conditions. Explored cells are shown in grey if empty or black if containing/bordering an obstacle; unexplored cells are shown in red when bordering explored cells, else in green.

4.3 Communication Prototype

In order to evaluate the mixed criticality cohesion behaviour it is necessary to use a mixed criticality wireless protocol that can expose the requisite network criticality level to the application. Since existing mixed criticality wireless protocols are not particularly suited to swarm robotics applications, a custom communication

prototype is developed. This communication prototype adapts the AirTight protocol studied in the previous chapter to use broadcast transmissions rather than point to point links, allowing messages to be delivered to all nodes. The network runs using time-division multiplexing over 10ms time slots. There is a periodic slot table assigned a priori, such that each node is assigned exactly one exclusive transmission slot. Clock synchronisation such that nodes agree on the current slot is assumed to be provided by some background service. Note that clocks do not need to be accurate over extended periods with respect to any real time, but only need to ensure that transmissions take place sequentially in the correct order, and that there is sufficient spacing between transmissions in order for receiving robots to process received frames. For the suggested 10ms TDMA slots a clock synchronisation accuracy in the order of milliseconds could therefore be sufficient, though more accurate synchronisation may be desirable in order to improve energy efficiency by placing the radio into a sleep mode between transmissions. Clock synchronisation could either piggyback on data transmissions or take place in dedicated reserved TDMA slots. A large number of protocols for clock synchronisation in wireless sensor networks have been proposed in prior work: protocols such RBS[45], FTSP[46], FBS[47], or FLOPSYNC-2[48] are able to provide clock synchronisation with microsecond to sub-microsecond accuracy.

AirTight relies on frame acknowledgements in order to determine whether a retransmission is necessary, but since using broadcast transmission prevents the use of radio hardware auto-acknowledgements, group acknowledgements are instead piggybacked on future transmissions. Each node maintains a bitfield of length $N - 1$ where each bit encodes whether the node successfully received a transmission in the last occurrence of the corresponding slot in the slot table. This bitfield is then included in the header of all transmission such that each receiving node can determine if its own last transmission was received by the sender.

For each transmission buffer, the node maintains a further bitfield of length $N - 1$ in which each bit encodes whether confirmation of successful delivery

has been received from a corresponding other node. When a node j receives a transmission from another node i, it checks its corresponding bit (bit j, where $j < i$, or bit $(j-1)$, where $j > i$) to determine if node i is acknowledging the last transmission of node j. If this bit is set, node j sets the corresponding bit of node i (bit i, where $i < j$, or bit $(i-1)$, where $i > j$) in the transmission bitfield of the last transmitted frame.

Switching between criticality modes is considered during each assigned transmission slot by using two counter, a counter of the length of the current busy-period, and an counter of the number of retransmissions, as show by the flowchart in figure 4.2. At the start of an assigned transmission slot, a node first checks whether the frame it broadcast in its previous transmission slot has been acknowledged by all other nodes, i.e. it checks if all bits are set in the corresponding buffer's transmission bitfield. If all bits are set, then all other nodes have acknowledged delivery of the frame at the head of the buffer, so the buffer's bitfield is cleared and the frame removed. Otherwise the frame will need to be retransmitted, so the retransmission counter is incremented. If the retransmission counter was incremented and the node is still in LO criticality mode, it considers switching to HI criticality mode by comparing the retransmission counter with the maximum number of faults the LO criticality fault model accepts for the current busy-period duration, switching to HI criticality mode if this bound is determined to have been exceeded. Switching to HI mode involves dropping all frames in LO buffers. The node selects the first frame from the highest priority non-empty buffer as the frame to be transmitted, increments its busy-period counter, then transmits the selected frame. Should no such frame exist (i.e. if all buffers are empty) the node resets its busy-period counter, retransmission counter, and criticality mode, before transmitting an empty frame such that the group acknowledgement bitfield can be delivered.

Figure 4.2: Flowchart showing criticality mode change triggers and management of related busy-period and retransmission counters.

4.4 Simulation Setup

The robot behaviour is again evaluated using the ARGoS robot simulator with an extension of the custom communication plugin developed in the previous chapter. Robots are configured with two network flows, one for position messages and one for sensed cell status. Since the focus of these experiments is on the effect of the application level behavioural changes based on observed network criticality level, both flows are configured as *HI* criticality such that criticality mode changes do not have an affect at the network level. Since positional data is time sensitive in order to ensure the minimum robot separation is maintained, this flow is assigned the higher priority. The network is configured such that positional messages are set to be retried for up to a maximum of 0.8 seconds in order to prevent old position data filling up transmission buffers, while cell status message are set not to expire since these are not particularly time sensitive.

The robot behaviour is evaluated across 100 randomly generated simulated arenas across four different transmission models. The transmission models, shown in figure 4.3, are configured to assume the packet delivery rate is capped at a maximum value of 95% for distances of less than 0.5m, after which the packet delivery rate is inversely proportional to the square distance between the nodes. A constant scaling factor k is used to modify the rate at which the packet delivery rate decreases with node distance. By modifying this scaling factor, the effect of different packet delivery rates on the application behaviour can be observed.

$$\text{PDR} = \frac{0.95}{1 + (k \cdot X)^2}$$

Where:

$$X = \begin{cases} 0 & d \leq 0.5 \\ d - 0.5 & d > 0.5 \end{cases}$$

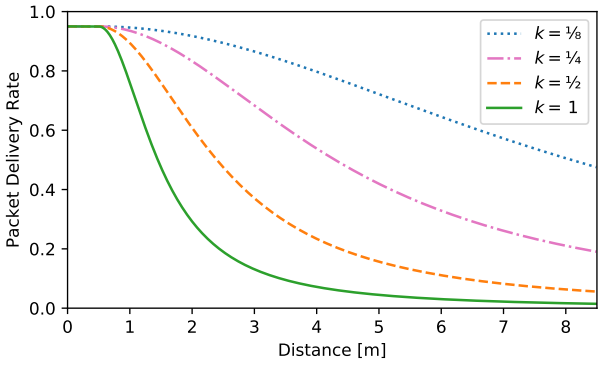

Figure 4.3: Transmission model defining packet reception probability in relation to distance, for scaling factors $k = \frac{1}{8}$, $k = \frac{1}{4}$, $k = \frac{1}{2}$, and $k = 1$.

This is again a simple model that does not attempt to capture the true complexity of real-world wireless communication. Prior research has shown that observed packet delivery rates do not correlate as strongly with distance in real-world experiments. Nonetheless, observed results broadly show that there exists a safe distance cutoff up to which the communication is generally reliable. Baccour et al. describe this at the "connected" region [40], which is followed by the "transitional" and "disconnected" regions where packet delivery first becomes intermittent and then mostly unsuccessful. The assumption that robots in the mixed criticality cohesion configuration will eventually encounter packet loss when dispersing, and packet loss will eventually reduce once robots move closer back together is therefore broadly supported by real-world experiments.

4.5 Evaluation

The simulation results in figure 4.4 show that the transmission model configuration with the most gradual drop-off in packet delivery rates, where $k = \frac{1}{8}$, the robot configuration that does not apply a cohesion force results in the highest performance. Since robots can adequately communicate over the entire arena there is no advantage to moving as a cohesive group. The half and constant cohesion configurations show reduced application performance since robots are less easily able to reach unexplored cells. With a mixed criticality configuration the robots are able to to remain in LO criticality mode most of the time, such that this mode, while slightly less performant than the no cohesion mode, still provides greater performance that either the half or constant cohesion modes.

Increasing the transmission model's distance scaling factor to $k = \frac{1}{4}$ causes a maximal dispersion of the robots to impede communications. The no cohesion robot configuration therefore shows much lower application performance than in the $k = \frac{1}{8}$ scenario. In contrast, the half and constant cohesion configurations are broadly unaffected, since the robots are kept sufficiently close together that packet loss remains insignificant. The mixed criticality configuration switches between LO and HI criticality mode, leading to application performance similar

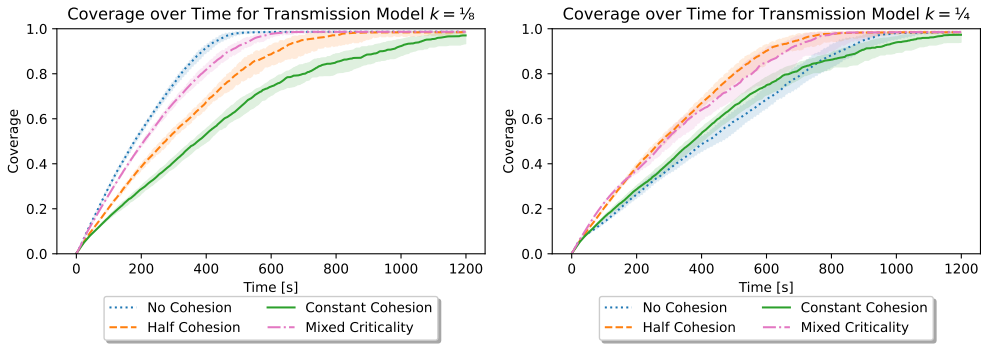

Figure 4.4: Map coverage over time across the four cohesion modes for transmission models with scaling factor $k = \frac{1}{8}$ and $k = \frac{1}{4}$. Line shows median value, shaded region shows interquartile range across 100 simulation runs.

to that of the half cohesion configuration.

A further increase of the transmission model's scaling factor to $k = \frac{1}{2}$, shown in figure 4.5, causes the no cohesion configuration to break down. Unable to effectively communicate, different robots need to re-explore the same area, yielding low map coverage. The half cohesion configuration is now also affected by reduced packet delivery, showing a wider variation in performance depending on specific runtime scenario. With a constant cohesion configuration, robot performance is again unchanged. The mixed criticality configuration is able to adapt to the conditions, spending slightly more time in *HI* criticality mode and yielding the highest performance of the tested robot configurations.

With the most restrictive transmission model configuration where $k = 1$, the mixed criticality configuration results in a *HI* criticality mode for large proportion of time, and thus performs similarly to the constant cohesion configuration, which still shows unchanged performance. The half cohesion and no cohesion configuration both show very poor performance, with the half cohesion mode no longer resulting in a sufficient tight formation to maintain reliable communication between nodes.

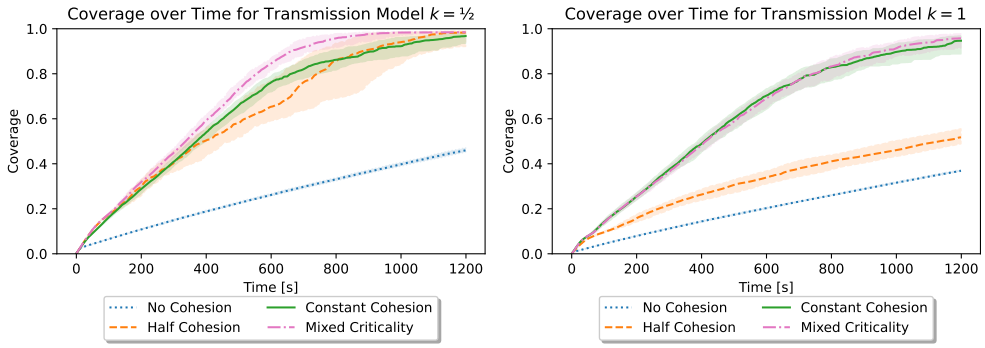

Figure 4.5: Map coverage over time across the four cohesion modes for transmission models with scaling factor $k = \frac{1}{2}$ and $k = 1$. Line shows median value, shaded region shows interquartile range across 100 simulation runs.

4.6 Summary

This chapter presents a mixed criticality approach to swarm robotics application behaviour, using the criticality mode of the network layer as an interface between the application and network components of a swarm robotics system. The parameters of the application behaviour may have different optimal values depending on the conditions encountered by the robot. If the system designer could know these conditions a priori, then static optimal values could be selected. In reality, however, the conditions may deviate from those expected by the system designer, resulting in degraded performance. A mixed criticality approach allows a systematic way for the application to define its requirements and adapt to the current conditions.

Chapter 5

Protocol and Timing Analysis

The previous chapter discussed using a criticality mode as an interface between the application and networks components in a swarm robotics system with a prototype mixed criticality network protocol. This chapter proposes a mixed criticality real-time wireless MAC protocol complete with timing analysis such that the criticality modes can act as a contract between the application and network components. As a result, the application can be guaranteed different levels of network performance so long as faults remain within predefined bounds or, in failure scenarios, at least be aware that these guarantees can no longer be provided because fault bounds have been exceeded. This allows the application to adapt its behaviour even under failure conditions. The protocol provides, subject to criticality modes bounded by fault models, strong timing guarantees over message delivery to all nodes, which can in turn be extended to an atomic broadcast property through additional local buffering.

5.1 Discussion of Fault Models

The nature of the fault model used by a real-time wireless protocol has a significant impact on the timing guarantees that can be provided. In order for these guarantees to be useful the fault model must be neither too permissive nor too restrictive. A fault model that is too permissive of the fault patterns that can occur is likely to provide very large, and therefore not particularly useful,

response time bounds. Equally, however, a fault model that is too restrictive might prohibit outcomes that commonly occur in practice, and thus quickly be exceeded, invalidating any timing guarantees. A further challenge in designing useful fault models is creating these such that an eventual application developer can determine suitable concrete values.

From a timing analysis perspective, when attempting to guarantee message delivery to all nodes, the most useful fault model would, as with the original AirTight analysis, directly bound the number of retransmissions required in a given time period. Adapting this to broadcast transmissions would, however, require a fault to be defined as a transmission for which the transmitting node does not, over the course of the next slot table iteration, receive an acknowledgement from all nodes that had not yet acknowledged the frame. This description is not immediately intuitive and furthermore is difficult to model because it introduces dependencies between transmissions as shown in figure 5.1. In case 1, where both the initial transmission and retransmission deliver the frame to node B but not node C, a fault occurs after both transmissions since after both transmissions there remains a node (C) that has not acknowledged. In case 2, a fault does *not* occur during after the retransmission (assuming the group acknowledgements are received) since node C has already acknowledged the frame after the original transmission. Thus a temporal component is introduced such that it is necessary to have knowledge of the outcome of previous transmission to determine whether a fault has occurred during the retransmission. Modelling these probabilities in order to select suitable concrete values for such a fault model is complicated by the presence of both spatial and temporal correlations. This is particularly problematic for *HI* fault models that should be able to handle node failures, since these failures a likely to change the nature of these correlations.

A simpler fault model definition can be achieved by introducing the notion of pairwise-faults. A pairwise-fault is defined as a transmission for which the transmitting node does not, for any specific possible receiving node, receive an acknowledgement during the receiving node's next transmission slot. The overall fault model is then defined as a bound on the maximum number of pairwise-fault

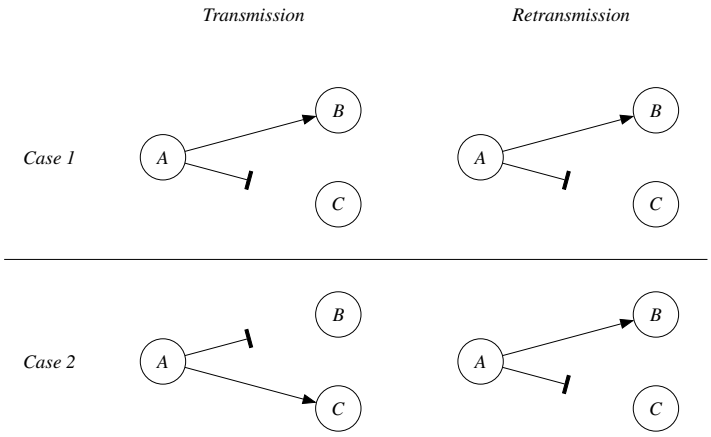

Figure 5.1: Naive fault model applied to two cases. In case 1, a fault occurs after both the initial transmission and the retransmission, while in case 2 a fault only occurs after the initial transmission (assume the group acknowledgements are successfully received).

that can occur between any pair of nodes within any fixed-size sliding window of n consecutive transmission. While this model is more intuitively understandable and amenable to practical measurements that establish baseline values, it allows pathological case fault patterns where a different node experiences the maximum number of faults for each frame. While it is presumably prudent to handle such cases in a *HI* fault model, it may be desirable to restrict the ability of such patterns to occur in a LO fault model in order to achieve tighter response time bounds and instead trigger a mode change in the unlikely event that they occur. Making this tradeoff requires knowledge of the application requirements, and therefore depends on the scenario.

Therefore, in order to provide flexibility to an application designer, the proposed protocol takes the following three fault models:

- A pairwise *HI* fault model, that bounds the maximum number of pairwise faults between any pair of nodes to f_{HI} within any sliding window of size n.

- A pairwise LO fault model, that similarly bounds the maximum number of pairwise faults between any pair of nodes to f_{LO} within an equivalent

sliding window of size n, where $f_{LO} < f_{HI}$.

- An overall LO fault model, that bounds the maximum number of overall faults to F_{LO} within the sliding window of size n, where $F_{LO} \geq f_{LO}$.

The pairwise fault models are used to derive message reception guarantees such that so long as f_{LO} or f_{HI} are not exceeded, all nodes will receive all transmitted LO or HI frames respectively. The overall fault model F_{LO} is used to control transmission guarantees, i.e. node will only attempt to deliver LO frames until this bound is exceeded. This therefore means that if pairwise-fault bounds are exceeded, the receiving node detects that it is (potentially) missing some message, while if the overall fault bound is exceeded the transmitting node is informed that further LO frames will be abandoned. The application designer can choose whether to use both the LO fault models, or can choose to use only one by either assuming the pairwise fault model is equal to the overall fault model (effectively disabling the LO pairwise-fault bound), or defining an overall fault model that allows arbitrarily many faults (effectively disabling the overall fault bound).

5.2 Protocol Description

The proposed protocol guarantees message delivery and reception of message between all nodes, subject to their criticality modes. Each node has two criticality modes: a transmission criticality mode that is a feature of the network and provides guarantees on whether message delivery will be attempted, and a reception criticality mode which can be viewed as a feature of the application and provides guarantees on reception of transmitted messages. A node in LO transmission criticality mode must attempt delivery of all messages, while a node in HI transmission criticality mode will only attempt to deliver HI messages. Similarly, a node in LO reception criticality mode must receive all messages for which a delivery attempt was made, while a node in HI reception criticality mode must receive all transmitted HI messages. The effective criticality mode of communication between any pair of nodes is therefore determined by the greater of the transmitting node's transmission criticality mode and the receiving

node's reception criticality mode. By partially shifting responsibility from the receiving to the transmitting node, the provided timing guarantees can become more meaningful to the application. Nodes can be certain that, subject only to their local reception criticality mode, they have received the same set of messages as other nodes. This can be particularly useful for sporadic messages where the absence of a message after the message inter-arrival period is not by itself unusual. Furthermore, by locally buffering incoming messages until the end of the message response time window, nodes can not only be certain that they have received that same set of messages as other nodes, but also guarantee that the messages are delivered to the application at the same time on all nodes (again subject to criticality modes). The reception criticality modes are viewed as a feature of the application rather than the network because the current reception criticality mode has no impact on the network behaviour, it is simply property to which the application may wish to react.

The protocol is implemented as a time-division multiplexing network and assumes there exists some background clock synchronisation such that nodes agree on the current slot. An a priori assigned periodic slot table with slots of some fixed length allows at most one node to broadcast a single frame in any given slot. Each node is assigned exactly one transmission slot in each slot table iteration. Nodes use the same system of delayed acknowledgements as the previous prototype protocol, where a bitfield included in each transmission encodes whether the transmitting node received a valid frame in the last occurrence of each slot in the slot table. Receiving nodes can therefore use this bitfield to determine the number of pairwise faults that have occurred with other node by checking each received frame to determine whether the corresponding node is acknowledging a previous transmission. Since the fault bounds are defined as reception of a successful acknowledgement, failure to receive a frame containing this bitfield is equivalent to receiving explicit negative ACKs for these accounting purposes.

In addition to this acknowledgement bitfield, each frame contains a LO and a HI frame sequence number and a node ejection bitfield. The sequence numbers encode the number of unique frames that have been transmitted at the respective

criticality level, while the ejection bitfield encodes a vote on whether other nodes should be removed from the network due to exceeded *HI* fault bounds. This bitfield is used to detect failed nodes such that the rest of the network can continue operating.

Each message flow is assigned a unique fixed priority level and buffers frames to be transmitted in a FIFO buffer. Nodes then use non-preemptive fixed priority scheduling to select a frame to be broadcast during their assigned transmission slot. The first frame from the highest priority non-empty buffer at or above the criticality level of the node's current transmission criticality mode is selected for broadcast, and will then be retransmitted until either all healthy nodes have acknowledged reception, or the number of pairwise faults exceeds the bound given by the relevant fault model for all nodes that have not acknowledge reception. During idle periods where no such frame exists, a node must broadcast an empty frame so that it can acknowledge frames received from other nodes, as well as acting as a heartbeat signal. Since such idle slots mean the end of any busy period, it is safe for nodes to return to LO transmission criticality mode at such time.

The protocol requires that broadcasts from each node can be received by all other nodes, subject to faults bounded by the fault models, and so does not immediately handle multi-hop transmissions. Note, however, that the properties of the protocol mean it lends itself to adaption into a multi-hop design. The specific implementation of such a system is left to future work, and is strongly dependent on both the chosen physical layer and the requirements of the application. Broadly, it appears plausible that multi-hop behaviour might be handled at either a higher or lower layer:

- The protocol does not depend on a specific underlying layer, requiring only the possibility for frames to be broadcast according to an a priori defined TDMA schedule and assuming that some form of background clock synchronisation is possible. Therefore, these broadcasts do not necessarily need to be implemented as just a simple physical layer transmission, but could, subject to satisfaction of tighter clock and timing requirements,

and further subject to restrictions on the ratio between slot length and transmission duration, involve in-slot retransmissions from receiving nodes. A simple scheme might, through some heuristic or shared link quality information, select one or more specific nodes to perform in-slot retransmissions. A more ambitious approach could be to implement each broadcast using a synchronous transmission layer such as Glossy, such that this protocol provides mixed criticality semantics and decentralised control while the underlying layer flattens the dynamic network topology. Note that this approach could also allow the use of clock synchronisation protocols such as FLOPSYNC-2 [48] that builds upon Glossy.

- Alternatively, multi-hop communication could be enabled at a higher later. The real-time atomic broadcasts enabled by the protocol could be used to reconfigure the network where necessary, such that nodes that cannot directly communicate are present in different network instances, with higher level logic then responsible for aggregating and distributing information across these instances. Note that protocol does not prevent the presence of unused idle slots in the slot table, potentially allowing a single radio to simultaneously be part of multiple different protocol instances.

The reception criticality mode change logic of the protocol is show by the flowchart in figure 5.2. A mode change from LO reception criticality mode to HI reception criticality mode takes place immediately if any of the following conditions hold:

- There exists some other node from which no frame is received for $f_{LO} + 1$ consecutive transmission slots.
- The node receives a frame from some other node where the sequence numbers indicate some frame is missing through any of the following conditions:
  - The sum of the sequence numbers has increased by more than one compared to the sum of the sequence number of the last received frame.

  - The sum of the sequence numbers has increased by one, but the received frame was an acknowledgement only frame.

A mode change to failure mode is queued to take place after a further $f_{HI}+1$ transmission slots if any of the following conditions occur:

- There exists some other node from which no frame is received for $f_{HI}+1$ consecutive transmission slots.
- The node receives a frame from some other node where the sequence numbers sequence number indicates a *HI* frame is missing through any of the following conditions:
  - The *HI* sequence number has increased by more than one compared to the last received frame.
  - The *HI* sequence number has increased by one, but the received frame is not a a *HI* frame.

During this window the node will vote, via the ejection bitfield in all further transmissions, to eject the other node from the network. A queued mode change can then be cancelled if, within this window, the node receives, from a majority of nodes, an ejection vote for the other node from which a frame is missing.

5.3 Proofs of Correctness

This section provides a series of lemmas that upon which the timing analysis presented in the following section is constructed. Lemmas 1 and 2 form the basis of the LO criticality analysis, while lemmas 3 and 4 form the basis of the *HI* criticality analysis.

Lemma 1 *Nodes will remain in LO reception mode unless the LO fault bound is exceeded.*

Consider each possible case in which a mode change from LO to *HI* mode can occur:

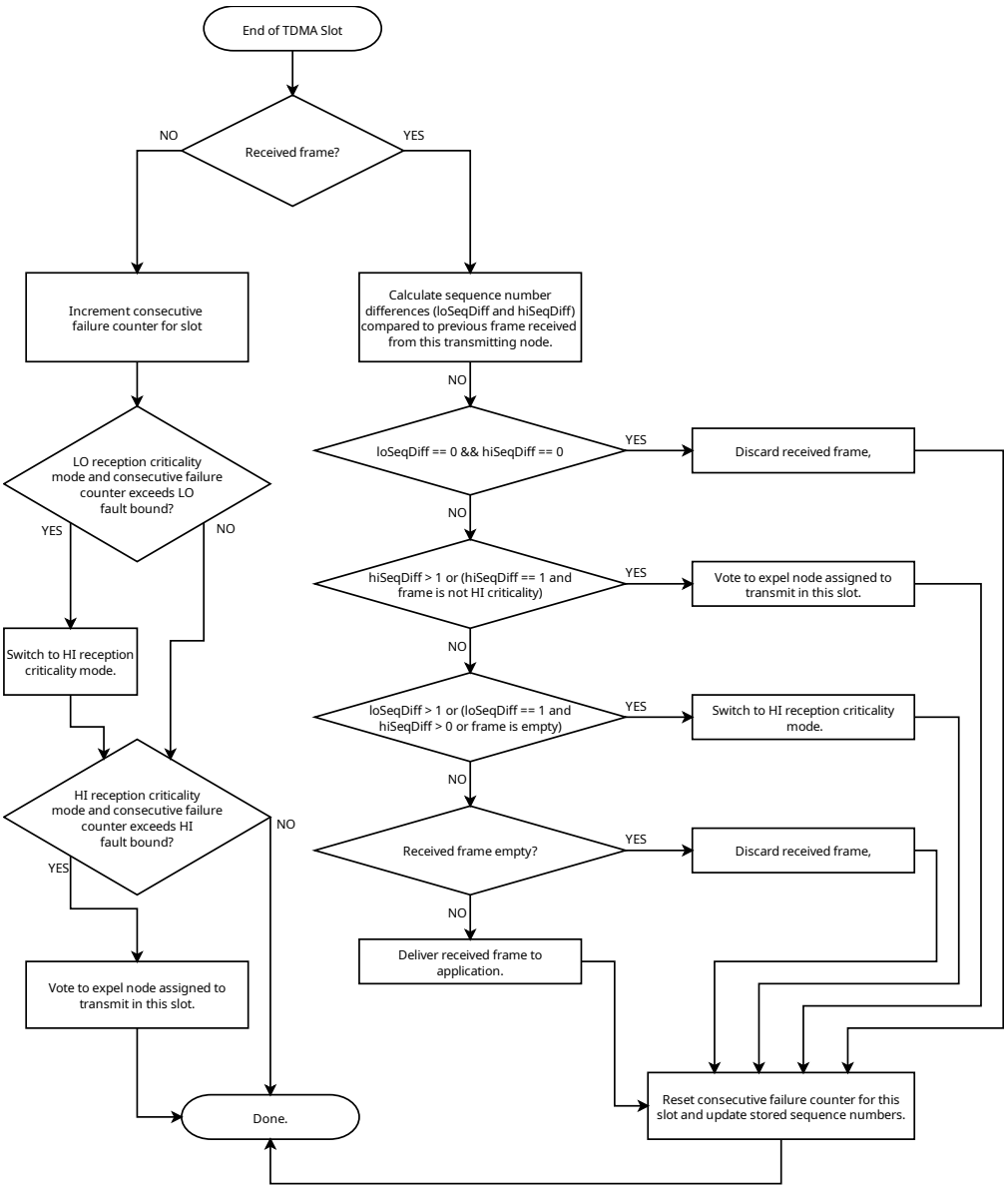

Figure 5.2: Flowchart showing reception criticality mode change logic of the protocol

- For some other node, the node does not receive any message within $f_{LO}+1$ slot table iterations. Failing to receive a message from some other node for $f_{LO}+1$ slot table iterations means that this node cannot, during that window, have received any acknowledgements for its own frames from that node. Thus the LO fault bound must have been exceeded.

- The node receives a frame from some other node, where the LO and HI sequence numbers indicate it has missed a frame transmitted between this frame and the last received frame. The transmitting node will, however, have broadcast that frame until it received an acknowledgement from all nodes that have not exceeded the pairwise-fault bound. Therefore, for this node to not have received that frame, the pairwise-fault bound must have been exceeded.

All cases which trigger a mode change imply that the LO fault bound has been exceeded, thus it is not possible for a mode change to HI mode to occur unless the LO fault bound has been exceeded.

Lemma 2 *For nodes that remain in LO reception mode, any frame broadcast from nodes remaining in LO transmission mode will be received within* $f_{LO}+1$ *slot table iterations from the instant the frame was first broadcast.*

Proof by contradiction: assume a node remains in LO reception mode and does not receive a frame from some other node. Consider the $f_{LO}+1$ slots from the first time that frame was broadcast, for which there are two possible cases:

- The node does not receive any frames from that transmitting node within that window. This means the node has not received any frame from that node within $f_{LO}+1$ slot table iterations, which triggers a mode change to HI reception mode.

- The node receives some other frame within that window. Since frames are broadcast non-preemptively and the window began at the instant when the frame was first broadcast, the received frame must be a subsequent frame.

The sequence numbers will therefore indicate that a frame has been missed, triggering a mode change to *HI* reception mode.

The assumption always leads to contradiction, hence it is not possible for a node to remain in LO reception mode and fail to receive, within $f_{LO} + 1$ slot table iterations, a frame that is broadcast by some other node in LO reception mode.

Lemma 3 *Nodes will not queue a reception mode change to the failure mode unless the HI fault bound is exceeded.*

Consider the possible cases which trigger a reception mode change to the failure mode to be queued:

- For some other node, the node does not receive any message within $f_{HI} + 1$ slot table iterations. Failing to receive a message from some other node for $f_{HI} + 1$ slot table iterations means that this node cannot, during that window, have received any acknowledgements for its own frames from that node. Thus the *HI* fault bound must have been exceeded.
- The node receives a frame from some other node where the *HI* sequence number indicates it has missed a frame transmitted between this frame and the last received frame. The transmitting node will, however, have broadcast that frame until it received an acknowledgement from all nodes that have not exceeded the pairwise-fault bound. Therefore, for this node to not have received that frame, the pairwise-fault bound must have been exceeded.

All cases which trigger a mode change imply that the *HI* fault bound has been exceeded, thus it is not possible for a mode change to failure mode to be queued unless the *HI* fault bound has been exceeded.

Lemma 4 *For nodes that are neither queued to enter the reception failure mode, nor already in the reception failure mode, any HI frame broadcast from any other node will be received within $f_{HI} + 1$ slot table iterations from the instant the frame was first broadcast.*

Proof by contradiction: assume a node is neither in the failure mode nor queued to enter the failure mode and has not received a *HI* frame from some other node. Consider the $f_{HI} + 1$ slots from the first time that frame was broadcast, for which there are two possible cases:

1. The node does not receive any frames from that transmitting node within that window. This means the node has not received any frame from that node within in $f_{HI} + 1$ slot table iterations, which triggers a mode change to the failure mode to be queued.

2. The node receives some other frame within that window. Since frames are broadcast non-preemptively, and the window began at the instant when the frame was first broadcast, the received frame must be a subsequent frame. The sequence numbers will therefore indicate that a *HI* frame as been missed, triggering a mode change to the failure mode to be queued.

The assumption always leads to contradiction, hence it is not possible for a node to fail to receive, within $f_{HI} + 1$ slot table iterations, a *HI* frame broadcast from some other node without entering or being queued to enter the failure mode.

Theorem 1 *By lemmas 1 and 2, all nodes that do not exceed the LO fault bound will remain in LO reception criticality mode and therefore receive all LO messages, while by lemmas 3 and 4 all nodes that do not exceed the HI fault bound will not enter failure mode and will thus receive all HI messages. Therefore, the protocol will always satisfy the claimed message delivery guarantees.*

5.4 Timing Analysis

5.4.1 Guarantee Function

Guarantee function $G_L(A, B)$ provides a bound on the maximum number of transmission slots required to deliver A *LO* frames and B *HI* frames to all other nodes at given criticality level L. The *HI* guarantee function $G_{HI}(A, B)$ is constructed using the pairwise-fault models. The minimum number of successful

Table 5.1: Table of notation for the proposed protocol's timing analysis

Notation	Definition
R_i	Computed response time for flow τ_i
T_i	Minimum inter-arrival period for messages from flow τ_i
C_i	Maximum size (in frames) of messages from flow τ_i
X_i	Maximum number of interfering *LO* frames
Y_i	Maximum number of interfering *HI* frames
Z_i	Maximum duration of interference
$hpL(i)$	Set of higher priority *LO* criticality flows
$hpH(i)$	Set of higher priority *HI* criticality flows
$lpL(i)$	Set of lower priority *LO* criticality flows
$lpH(i)$	Set of lower priority *HI* criticality flows
N	Number of nodes in the system
n	Number of slot table iterations that create a fault-model window
α	The minimum number of successful transmissions within the fault model window for the *LO* pairwise fault bounds
β	The minimum number of successful transmissions within the fault model window for the *HI* pairwise fault bounds
γ	The minimum number of successful transmissions within a fault model window for the overall fault model

transmissions within a fault model window, α and β for the LO and HI fault bounds respectively, is given by subtracting the maximum number of faults from the number of slot table iterations per window. Where multiple frames are guaranteed to transmit during a fault window, the overall guarantee bound is then computed from the necessary number of complete fault model windows, plus the additional slots required to guarantee delivery of the remaining frames. Otherwise, if the fault bound only implies guaranteed delivery of a single frame during a fault window, then each frame might require up to $f_L + 1$ transmissions to guarantee delivery. Thus, the guarantee bound is defined as follows:

$$\alpha = \max(1, n - f_{LO} \cdot (N - 1)) \tag{5.1}$$

$$\beta = \max(1, n - f_{HI} \cdot (N - 1)) \tag{5.2}$$

$$G_{HI}(A, 0) = \begin{cases} n \cdot \left\lfloor \frac{A}{\alpha} \right\rfloor + f_{LO} \cdot \min(N - 1, A \bmod \alpha) + (A \bmod \alpha) & A > 0 \wedge \alpha > 1 \\ A \cdot (f_{LO} + 1) & \text{otherwise} \end{cases} \tag{5.3}$$

$$G_{HI}(0, B) = \begin{cases} n \cdot \left\lfloor \frac{B}{\beta} \right\rfloor + f_{HI} \cdot \min(N - 1, B \bmod \beta) + (B \bmod \beta) & B > 0 \wedge \beta > 1 \\ B \cdot (f_{HI} + 1) & \text{otherwise} \end{cases} \tag{5.4}$$

$$G_{HI}(A, B) = G_{HI}(A, 0) + G_{HI}(0, B) \tag{5.5}$$

The same bound also applies to the LO guarantee bound, but here a further bound is provided using the overall fault model. This bounds the maximum number of transmissions required for any number of frames LO and HI frames:

$$\gamma = n - F_{LO} \tag{5.6}$$

$$G_{LO}(A, B) = \min\left(G_{HI}(A, B), n \cdot \left\lfloor \frac{A + B}{\gamma} \right\rfloor + F_{LO} \cdot \min(1, (A + B) \bmod \gamma) + (A + B) \bmod \gamma\right) \tag{5.7}$$

5.4.2 LO-Criticality Flows

Timing analysis for messages from LO flow i is constructed by first determining the maximum interference from other LO and HI frames that need to be transmitted

before the final frame of a message in i, then adding the maximum time required to transmit the final frame. The number of interfering LO frames, $X_{i,LO}$, is derived from $C_i - 1$, i.e. all but the last frame of the message, as well as the maximum number of interfering frames from higher priority LO buffers following typical response time analysis for fixed priority scheduling. Similarly, the number of interfering HI frames, $Y_{i,LO}$, is the maximum number of interfering frames from higher priority HI buffers. Hence, where $hpL(i)$ is the set of higher-priority LO flows, and $hpH(i)$ is the set of higher-priority HI flows, $X_{i,LO}$ and $Y_{i,LO}$ are defined as follows:

$$X_{i,LO} = C_i - 1 + \sum_{j \in hpL(i)} \left\lceil \frac{Z_{i,LO}}{T_j} \right\rceil C_j \tag{5.8}$$

$$Y_{i,LO} = \sum_{j \in hpH(i)} \left\lceil \frac{Z_{i,LO}}{T_j} \right\rceil C_j \tag{5.9}$$

The overall maximum interference, Z_i, is then determined by guarantee function $G_{LO}(A, B)$ (described in section 5.4.1) that bounds the maximum number of slot table iterations required to transmit A LO frames and B HI frames without exceeding LO fault model. In addition to interfering frames, initial blocking caused by non-preemptive transmission of a lower priority frame must also be accounted for. This blocking time is, in the worst case, the full transmission of a single frame at the maximum criticality level of a lower priority buffer. Therefore, the maximum interference is computed as follows:

$$Z_{i,LO} = \begin{cases} G_{LO}(X_{i,LO}, Y_{i,LO} + 1) & lpH(i) \neq \emptyset \\ G_{LO}(X_{i,LO} + 1, Y_{i,LO}) & lpH(i) = \emptyset \wedge lpL(i) \neq \emptyset \\ G_{LO}(X_{i,LO}, Y_{i,LO}) & \text{otherwise} \end{cases} \tag{5.10}$$

Overall response time is then given by adding $f_{LO} + 1$ slot table iterations, providing a response time bound that satisfies lemma 2 such that it is certain that the final frame has been delivered to all other nodes, or that any node that has not received the frame has detected a fault model violation.

$$R_{i,LO} = Z_{i,LO} + f_{LO} + 1 \tag{5.11}$$

5.4.3 HI-Criticality Flows

Timing analysis for messages in a *HI* flow i is constructed similarly. The maximum number of interfering LO frames $X_{i,HI}$ is derived from the higher priority LO buffers, while the number of interfering HI frames consists of all the frames of the message bar the last, as well as interfering frames from higher priority HI frames. Interference from frame in LO buffers is restricted to frames released during $Z_{i,LO}$

$$X_{i,HI} = \sum_{j \in hpL(i)} \left\lceil \frac{Z_{i,LO}}{T_j} \right\rceil C_j \tag{5.12}$$

$$Y_{i,HI} = C_i - 1 + \sum_{j \in hpH(i)} \left\lceil \frac{Z_{i,HI}}{T_j} \right\rceil C_j \tag{5.13}$$

Overall interference Z_i is determined by guarantee function $G_{HI}(A, B)$ that bounds the maximum number of slot table iterations required to transmit A LO frames and B HI frames.

$$Z_{i,HI} = \begin{cases} G_{HI}(X_{i,HI}, Y_{i,HI} + 1) & lpH(i) \neq \emptyset \\ G_{HI}(X_{i,HI} + 1, Y_{i,HI}) & lpH(i) = \emptyset \wedge lpL(i) \neq \emptyset \\ G_{HI}(X_{i,HI}, Y_{i,HI}) & \text{otherwise} \end{cases} \tag{5.14}$$

Overall response time is then given by adding $2 \cdot (f_{HI} + 1)$ slot table iterations in order to ensure that the final frame has either been delivered to all other nodes, or that any node that has not yet received the frame has either switched into the reception failure mode, or received votes to eject the transmitting node from the network.

$$R_{i,HI} = Z_{i,HI} + 2 \cdot (f_{HI} + 1) \tag{5.15}$$

5.5 Discussion

The proposed protocol redefines criticality modes such that nodes that remain in LO reception criticality mode are guaranteed to have received all messages for which transmission was attempted, and nodes that do not exceed the HI reception criticality mode are guaranteed to have received at least all HI messages. These guarantees are particularly useful because they can allow a swarm robotics system to provide application level guarantees for some behaviours. Recall the "circle problem" presented in section 3.1 where robots attempted to agree on LED colour and forwards velocity. The earlier definition of problem purposefully makes it impossible over time since it requires robots to communicate with ever lower packet delivery rates. The guarantees provided by the new proposed protocol can, however, provide application level guarantees for a variation of the problem with relaxed conditions

Consider a variation on task T_{LED} where the requirements are relaxed such that robots may discard some colour changes so long as all robots agree on which changes take place, and robots are able to turn off their LEDs when unable to determine the correct LED colour. For the protocols presented in chapter 3 this variation does not make the problem tractable except for the trivial cases of permanently turning off LEDs or discarding all colour changes. The criticality modes used by AirTight do not enable an implementation of this behaviour because its criticality modes only consider transmission faults as necessary to provide delivery guarantees from the perspective of the transmitting node. In contrast, the criticality modes of the proposed protocol map to application level behaviours: robots in LO reception criticality mode can be sure that all attempted colour changes have been received, while robots that switch to HI reception criticality mode must turn off their LEDs. The reception criticality mode is primarily a feature of the application since it has no impact on the network behaviour. Therefore, the application can safely choose to return to LO reception criticality mode once it is has recovered from the "fault" caused by the missing message. In this case, once the application has not been triggered to enter HI

reception criticality mode for over two seconds, it can switch back to LO reception criticality mode on the next LED colour change. By redefining the meaning of criticality modes, it is therefore possible to both provide the application level guarantee that all nodes with enabled LEDs will always show the same colour, as well as that all nodes without a fault model violation will have their LEDs enabled. Note that this application level guarantee holds regardless of the number of network faults that occur, and remains satisfied in the face of node failures so long as node fail with their LEDs off.

Figure 5.3 shows simulation results for this variant, showing the number of unique LED colours displayed at each time step as well as the number of nodes with LEDs enabled. The left chart shows results for the standard scenario as described in section 3.1, while the right side shows the equivalent application with a simulated node failure taking place after 30 seconds. The results in both scenario shows that regardless of the network fault that occur, there are never multiple different LED colours displayed at the same time. At the start of the simulation all robots have their LEDs enabled. As the robots move further apart, node begin switching between LO and HI reception criticality mode such that not all nodes have LEDs enabled. In the scenario featuring a simulated node failure, the remaining nodes all switch to HI criticality mode until the failure has been detected. Once the failure has been detected, the remaining five nodes can switch back to LO criticality mode.

5.6 Summary

This chapter proposes a mixed criticality real-time wireless MAC protocols suitable for some swarm robotics applications and complete with timing analysis. By redefining criticality modes as part of the proposed protocol, criticality modes can be raised closer to the application and provide more meaningful information as to the state of the network. As a result, it becomes possible to provide some application level guarantees for simple swarm behaviours.

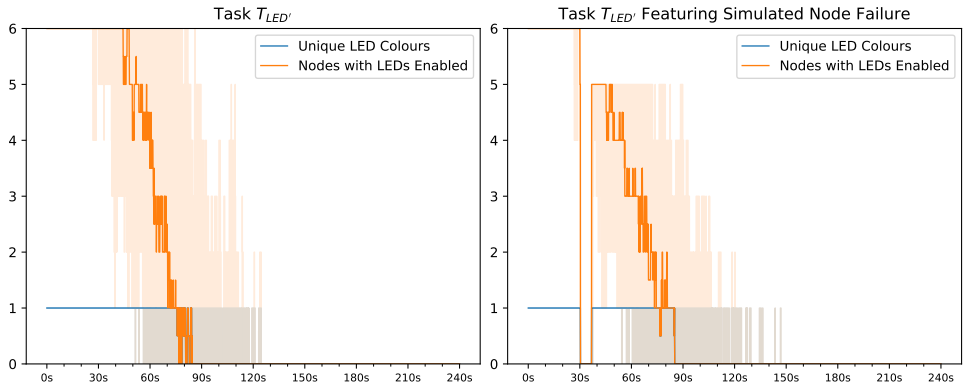

Figure 5.3: Median E'_{LED} after n seconds over for 20 different random seeds. Shaded area shows range from minimum to maximum E'_{LED} value at the given step over the runs.

Chapter 6

Conclusion

This chapter summarises the contributions of the previously presented work, revisits the research hypothesis, and suggests areas for future research.

6.1 Contributions

In this thesis the concept of mixed criticality is explored in the context of swarm robotics systems. Chapter 3 introduces the concept of mixed criticality to the swarm robotics domain by adopting a mixed criticality network layer. The evaluation shows that assigning criticality levels to communication flows naturally results in mixed criticality behaviour of the corresponding application behaviour components. This enables graceful degradation under network failure conditions and allows the system designer to better able to reason about the behaviour of the system.

In chapter 4 the criticality mode of the network is extended as an interface between the application and network components. This allows the application layer to be aware of the level of service it can receive from the network layer such that it can choose to adapt its own behaviour. The results show that a mixed criticality exploration behaviour can opportunistically adapt the cohesion parameter of the exploration algorithm to suit the network conditions encountered at runtime. This allows the overall exploration behaviour to show better average case performance without overly negatively affecting the worst case performance

as compared to the static cohesion methods.

Chapter 5 refines the communications prototype of the previous chapter into a complete protocol with timing analysis. Applying separate criticality modes for the transmission and reception side allows the protocol to provide real-time atomic broadcast semantics. By strengthening the meaning of criticality modes to be more useful to a swarm robotics application, it become possible to provide application level guarantees for simple swarm behaviours.

6.2 Research Hypothesis

The research hypothesis predicted that adopting a mixed criticality model to the swarm robotics domain could improve the reliability and predictability of these systems with regards to network faults. Improvement to predictability in terms of failure modes were demonstrated by the mixed criticality approach in chapters 3 and 4. The proposed protocol in chapter 5 further allows for higher level application behaviour guarantees regardless of the network faults that occur, allowing for a more reliable system. The research hypothesis has therefore been validated for at least some types of swarm robotics systems.

6.3 Future Work

This section outlines directions for possible future research into a mixed criticality approach to swarm robotics. Preliminary work has been undertaken in some of these directions but has not been completed due to timing constraints.

Application Level Guarantees

While the proposed protocol presented in chapter 5 enables some application level guarantees, one would ideally be able to provide end-to-end timing guarantees over larger swarm robotics applications. For example, an exploration systems such as the one described in chapter 4 might aim to be provide bounds on the maximum amount of time taken to explore a given area. These sorts of guarantees

might then in turn enable guarantees for real-world applications, for example by constraining the amount of time required to explore an area in a search and rescue application.

Future work could consider which conditions, both on the network and otherwise, may need to be satisfied to provide such guarantees. While this thesis has considered criticality mode changes as a result of network faults, once the criticality mode has been extended from purely a network property to an application level property, one might consider what other conditions the application might consider as a fault. Future work could therefore consider the categories of faults that may occur, and how different types of faults might interact with regards to higher level timing guarantees and criticality levels.

Scalability and Network Configuration

Proposed swarm robotics systems range from deployments with a single digit number of robots through to thousands. The timing analysis for the protocol presented in chapter 5 of this thesis assumes a TDMA protocol with a fixed slot table, resulting in timing guarantees that broadly scale linearly with the number of nodes. This creates a practical limit on the overall network size in order to provide useful timing bounds.

In many scenarios, however, it may not be necessary to maintain a single network that guarantees message delivery to all robots. For example, robots might not all be active simultaneously or the application might choose to partition the swarm into multiple groups that handle different tasks. Future work could therefore consider how the application could, either by predicting the impact of its own future actions on network conditions, or according to task allocation, choose to reconfigure the network into multiple sub-networks before fault bounds are exceeded.

Adopting multiple sub-networks that can transmit simultaneously without interfering (either through multiple channels or physical separation) allows the network to adapt to application requirements and allows a larger overall number of robots to be deployed. Future work could consider robots that act as a bridge

enabling multi-hop communication between multiple sub-networks, and what timing guarantees might therefore be possible for inter-network flows.

Real-World Experiments

The work in this thesis is primarily validated using simulation results. Real-world evaluation, both in terms of using physical robots in real-word environments, and in terms of using real-world swarm algorithms that can be evaluated in terms of some real-world benchmark, could allow future work to strengthen confidence in real-world applicability. Experiments could consider applications of the protocol using different underlying physical layers, such as IEEE 802.15.4 and IEEE 802.11, with testing in environments with different packet reception behaviour.

Bibliography

[1] S. Signer, A. Millard and I. Gray, "Mixed-criticality wireless communication for robot swarms," in *9th International Workshop on Mixed Criticality Systems*, WMC, Dec. 2022.

[2] S. Signer and I. Gray, "Adaptive application behaviour for robot swarms using mixed-criticality," in *Proceedings of the Third Workshop on Agents and Robots for reliable Engineered Autonomy*, ser. EPTCS, Sep. 2023, pp. 71–82. DOI: 10.4204/EPTCS.391

[3] D. Stormont, "Autonomous rescue robot swarms for first responders," in *CIHSPS 2005. Proceedings of the 2005 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety, 2005.*, 2005, pp. 151–157. DOI: 10.1109/CIHSPS.2005.1500631

[4] D. Albani, J. IJsselmuiden, R. Haken and V. Trianni, "Monitoring and mapping with robot swarms for agricultural applications," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6. DOI: 10.1109/AVSS.2017.8078478

[5] L. A. Nguyen, T. L. Harman and C. Fairchild, "Swarmathon: A swarm robotics experiment for future space exploration," in *2019 IEEE International Symposium on Measurement and Control in Robotics (ISMCR)*, 2019, B1-3-1-B1-3–4. DOI: 10.1109/ISMCR47492.2019.8955661

[6] R. Arnold, K. Carey, B. Abruzzo and C. Korpela, "What is a robot swarm: A definition for swarming robotics," in *2019 IEEE 10th Annual Ubiquitous*

Computing, Electronics & Mobile Communication Conference (UEMCON), 2019, pp. 0074–0081. DOI: 10.1109/UEMCON47517.2019.8993024

[7] M. Kegeleirs and M. Birattari, "Towards applied swarm robotics: Current limitations and enablers," *Frontiers in Robotics and AI*, vol. Volume 12 - 2025, 2025. DOI: 10.3389/frobt.2025.1607978

[8] S. Greengard, "Swarm robotics moves forward," *Commun. ACM*, vol. 65, no. 12, pp. 12–14, Nov. 2022. DOI: 10.1145/3565979

[9] M. M. Shahzad et al., "A review of swarm robotics in a nutshell," *Drones*, vol. 7, no. 4, 2023, ISSN: 2504-446X. DOI: 10.3390/drones7040269

[10] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *Swarm Robotics*, E. Şahin and W. M. Spears, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 10–20, ISBN: 978-3-540-30552-1. DOI: 10.1007/978-3-540-30552-1_2

[11] M. Selden, J. Zhou, F. Campos, N. Lambert, D. Drew and K. S. J. Pister, "Botnet: A simulator for studying the effects of accurate communication models on multi-agent and swarm control," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2021, pp. 101–109. DOI: 10.1109/MRS50823.2021.9620611

[12] V. Crespi, A. Galstyan and K. Lerman, "Top-down vs bottom-up methodologies in multi-agent system design," *Auton. Robots*, vol. 24, no. 3, pp. 303–313, Apr. 2008, ISSN: 0929-5593. DOI: 10.1007/s10514-007-9080-5

[13] C. Pinciroli and G. Beltrame, "Buzz: A programming language for robot swarms," *IEEE Software*, vol. 33, no. 4, pp. 97–100, 2016. DOI: 10.1109/MS.2016.95

[14] S. Hayat, E. Yanmaz and R. Muzaffar, "Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2624–2661, 2016. DOI: 10.1109/COMST.2016.2560343

[15] J. Krejčí, M. Babiuch, J. Suder, V. Krys and Z. Bobovský, "Latency-sensitive wireless communication in dynamically moving robots for urban mobility applications," *Smart Cities*, vol. 8, no. 4, 2025. DOI: 10.3390/smartcities8040105

[16] J. Gielis, A. Shankar and A. Prorok, "A critical review of communications in multi-robot systems," *Current Robotics Reports*, vol. 3, no. 4, pp. 213–225, Dec. 2022. DOI: 10.1007/s43154-022-00090-9

[17] C. C. Imrie, J. M. Herrmann and O. Witkowski, "The paradox of choice in evolving swarms: Information overload leads to limited sensing," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '21, Lille, France: Association for Computing Machinery, 2021, pp. 22–30, ISBN: 9781450383509. DOI: 10.1145/3449639.3459369

[18] A. Burns and R. I. Davis, *Mixed Criticality Systems - A Review*, 13th ed. Feb. 2022. [Online]. Available: https://eprints.whiterose.ac.uk/id/eprint/183619/

[19] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, 2007, pp. 239–243. DOI: 10.1109/RTSS.2007.47

[20] A. Burns and A. Wellings, *Real-Time Systems and Programming Languages*, 3rd ed. Addison Wesley, 2001, ISBN: 0201729881.

[21] S. Baruah, A. Burns and R. Davis, "Response-time analysis for mixed criticality systems," in *2011 IEEE 32nd Real-Time Systems Symposium*, 2011, pp. 34–43. DOI: 10.1109/RTSS.2011.12

[22] A. Addisu, L. George, V. Sciandra and M. Agueh, "Mixed-criticality scheduling applied to jpeg2000 video streaming over wireless multimedia sensor networks," in *1st International Workshop on Mixed Criticality Systems*, WMC, Dec. 2013.

[23] X. Jin, J. Wang and P. Zeng, "End-to-end delay analysis for mixed-criticality wirelesshart networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 3, pp. 282–289, 2015. DOI: 10.1109/JAS.2015.7152662

[24] X. Jin, C. Xia, H. Xu, J. Wang and P. Zeng, "Mixed criticality scheduling for industrial wireless sensor networks," *Sensors*, vol. 16, no. 9, 2016, ISSN: 1424-8220. DOI: 10.3390/s16091376

[25] A. Burns, J. Harbin, L. Indrusiak, I. Bate, R. Davis and D. Griffin, "Airtight: A resilient wireless communication protocol for mixed-criticality systems," in *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2018, pp. 65–75. DOI: 10.1109/RTCSA.2018.00017

[26] S. Montero, J. Gozalvez, M. Sepulcre and G. Prieto, "Impact of mobility on the management and performance of wirelesshart industrial communications," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, 2012, pp. 1–4. DOI: 10.1109/ETFA.2012.6489704

[27] F. Ferrari, M. Zimmerling, L. Thiele and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2011, pp. 73–84.

[28] M. Zimmerling, L. Mottola and S. Santini, "Synchronous transmissions in low-power wireless: A survey of communication protocols and network services," *ACM Comput. Surv.*, vol. 53, no. 6, Dec. 2020, ISSN: 0360-0300. DOI: 10.1145/3410159

[29] J. Lu and K. Whitehouse, "Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks," in *IEEE INFOCOM 2009*, 2009, pp. 2491–2499. DOI: 10.1109/INFCOM.2009.5062177

[30] F. Ferrari, M. Zimmerling, L. Mottola and L. Thiele, "Low-power wireless bus," ser. SenSys '12, Toronto, Ontario, Canada: Association for Computing Machinery, 2012, pp. 1–14, ISBN: 9781450311694. DOI: 10.1145/2426656.2426658

[31] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari and L. Thiele, "Adaptive real-time communication for wireless cyber-physical systems," *ACM Trans. Cyber-Phys. Syst.*, vol. 1, no. 2, Feb. 2017, ISSN: 2378-962X. DOI: 10.1145/3012005

[32] F. Terraneo, P. Polidori, A. Leva and W. Fornaciari, "Tdmh-mac: Real-time and multi-hop in the same wireless mac," in *2018 IEEE Real-Time Systems Symposium (RTSS)*, 2018, pp. 277–287. DOI: 10.1109/RTSS.2018.00044

[33] D. Tardioli and J. L. Villarroel, "Real time communications over 802.11: Rt-wmp," in *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems*, 2007, pp. 1–11. DOI: 10.1109/MOBHOC.2007.4428607

[34] D. Tardioli and J. L. Villarroel, "Adding multicast capabilities to wireless multi-hop token-passing protocols: Extending the rt-wmp," in *2009 IEEE Conference on Emerging Technologies & Factory Automation*, 2009, pp. 1–10. DOI: 10.1109/ETFA.2009.5347209

[35] Y.-H. Wei, Q. Leng, S. Han, A. K. Mok, W. Zhang and M. Tomizuka, "Rt-wifi: Real-time high-speed communication protocol for wireless cyber-physical control applications," in *2013 IEEE 34th Real-Time Systems Symposium*, 2013, pp. 140–149. DOI: 10.1109/RTSS.2013.22

[36] Q. Leng, W.-J. Chen, P.-C. Huang, Y.-H. Wei, A. K. Mok and S. Han, "Network management of multicluster rt-wifi networks," vol. 15, no. 1, Feb. 2019, ISSN: 1550-4859. DOI: 10.1145/3283451

[37] D. Tardioli et al., "Ground robotics in tunnels: Keys and lessons learned after 10 years of research and experiments," *Journal of Field Robotics*, vol. 36, no. 6, pp. 1074–1101, 2019. DOI: https://doi.org/10.1002/rob.21871

[38] G. Cerar, H. Yetgin, M. Mohorčič and C. Fortuna, "Machine learning for wireless link quality estimation: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 696–728, 2021. DOI: 10.1109/COMST.2021.3053615

[39] C. Pinciroli et al., "ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012. DOI: 10.1007/s11721-012-0072-5

[40] N. Baccour et al., "RadiaLE: A framework for designing and assessing link quality estimators in wireless sensor networks," *Ad Hoc Networks*, vol. 9, no. 7, pp. 1165–1185, 2011, ISSN: 1570-8705. DOI: 10.1016/j.adhoc.2011.01.006

[41] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '87, New York, NY, USA: Association for Computing Machinery, 1987, pp. 25–34, ISBN: 0897912276. DOI: 10.1145/37401.37406

[42] N. Baccour et al., "Radio link quality estimation in wireless sensor networks: A survey," *ACM Trans. Sen. Netw.*, vol. 8, no. 4, Sep. 2012. DOI: 10.1145/2240116.2240123

[43] A. G. Millard et al., "The pi-puck extension board: A raspberry pi interface for the e-puck robot platform," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 741–748. DOI: 10.1109/IROS.2017.8202233

[44] V. P. Tran, M. A. Garratt, K. Kasmarik, S. G. Anavatti and S. Abpeikar, "Frontier-led swarming: Robust multi-robot coverage of unknown environments," *Swarm and Evolutionary Computation*, vol. 75, p. 101 171, 2022, ISSN: 2210-6502. DOI: 10.1016/j.swevo.2022.101171

[45] J. Elson, L. Girod and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, Dec. 2003, ISSN: 0163-5980. DOI: 10.1145/844128.844143

[46] M. Maróti, B. Kusy, G. Simon and Á. Lédeczi, "The flooding time synchronization protocol," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, ser. SenSys '04, Baltimore, MD, USA: Association for Computing Machinery, 2004, pp. 39–49, ISBN: 1581138792. DOI: 10.1145/1031495.1031501

[47] J. Chen, Q. Yu, Y. Zhang, H.-H. Chen and Y. Sun, "Feedback-based clock synchronization in wireless sensor networks: A control theoretic approach,"

IEEE Transactions on Vehicular Technology, vol. 59, no. 6, pp. 2963–2973, 2010. DOI: 10.1109/TVT.2010.2049869

[48] F. Terraneo, L. Rinaldi, M. Maggio, A. V. Papadopoulos and A. Leva, "Flopsync-2: Efficient monotonic clock synchronisation," in *2014 IEEE Real-Time Systems Symposium*, 2014, pp. 11–20. DOI: 10.1109/RTSS.2014.14