
Proceedings Paper:
Signer, Sven, Millard, Alan Gregory and Gray, Ian (2022) Mixed-Criticality Wireless Communication for Robot Swarms. In: 9th International Workshop on Mixed Criticality Systems:Proceedings. 9th International Workshop on Mixed Criticality Systems, 05 Dec 2022 WMC, USA.

Mixed-Criticality Wireless Communication for Robot Swarms

Sven Signer
Department of Computer Science
University of York
York, United Kingdom
sven.signer@york.ac.uk

Alan G. Millard
Department of Computer Science
University of York
York, United Kingdom
alan.millard@york.ac.uk

Ian Gray
Department of Computer Science
University of York
York, United Kingdom
ian.gray@york.ac.uk

***Abstract*—In recent years the mixed criticality systems model has been adapted for use in shared-medium communication protocols, but it has not seen deployment into swarm robotics. This paper discusses ongoing work in the application of such a model to this domain, and argues for the benefits of such an approach. In many applications, reliability of communications is essential for the correct and safe operation of the robots. Given the inherently unreliable nature of wireless inter-robot communications, this paper argues for the application of timing- and criticality-aware communication protocols to be able to provide more reliable task-level performance of swarm robotics applications. In this work we define two illustrative swarm applications with two tasks at different criticality levels. Using simulation results we show that in the presence of wireless faults, standard best-effort protocols will cause application errors unpredictably, but a mixed-criticality wireless protocol can maintain important tasks at the cost of less important ones for longer.**

***Index Terms*—real-time wireless, mixed-criticality, swarm communication**

I. Introduction

Distributed autonomous systems rely on wireless communications to implement their functionality. If such systems are to be deployed in high-integrity environments, such as autonomous vehicles, then it is necessary to be able to reason about the performance of the system in situations where such communications are not reliable. Existing approaches in the field of swarm robotics rarely consider timing-aware communication, instead relying on mechanisms such as self-organisation and emergence for information propagation [1]. Failed communications can be corrected through retransmissions, but these also reduce available bandwidth and can cause further transmission failures. A standard best-effort protocol like WiFi does not allow the system integrator to analyse system performance ahead of time.

This paper argues that by implementing timing-aware communications protocols and by adopting a mixed criticality system model, it becomes possible to provide hard timing guarantees within a specified fault model, and to reason about system degradation in a controlled way when that model is exceeded. Our results show this translates into better task-level performance for an example swarm robotics application.

Section II begins by defining a motivating problem for this work to address. We then introduce mixed criticality in Section III and examine existing wireless protocols in section IV. We define our system model (Section V) and our experiment in Section VI, followed by results (Section VII), limitations (Section IX) and conclusions.

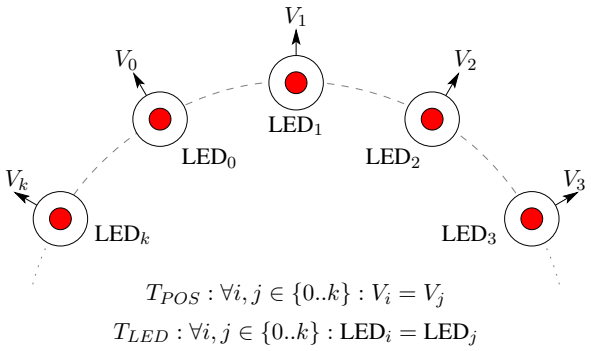

Fig. 1. k robots, each having a velocity V and an LED colour. Task T_{LED} requires all nodes to show the same colour. Task T_{POS} requires all nodes to move with the same velocity. At any given time, both conditions should be satisfied.

II. Motivating Problem

Consider an autonomous wireless swarm robotics platform in which the robots have two tasks:

- T_{LED}: Communicate to form a consensus about what LED colour to display. Each agent may choose to ask the swarm to display a different colour at any time, for example as a response to external stimuli.
- T_{POS}: Coordinate to maintain a circle formation. Each agent may choose to adjust the formation and the others must maintain relative positioning.

This system has two metrics of quality: timing error for task T_{LED}, and positional error for task T_{POS}. The system is trying to minimise both errors. Errors are discussed in more detail in Section VI. Communications are wireless, and so any given transmission has a probability of successful transmission, which decreases as traffic volume increases (due to collisions) and decreases as inter-agent distance increases (see Section V). A naive approach to this problem simply has each robot communicate with every other robot for either task, retransmitting if a transmission fails. However, this means that as the robots get further from each other, transmissions begin to dominate the available bandwidth and errors increase.

If we now introduce the constraint that T_{POS} is a high-integrity task that must be guaranteed to avoid the potential for injury, we need to be able to guarantee a given level of performance for T_{POS}. This guarantee becomes impossible (or at least incredibly pessimistic) due to unbounded interference from the rest of the system.

III. Mixed Criticality Systems

Given the problem introduced in Section II, this work argues that future autonomous swarm robotics systems should be viewed as a mixed criticality system (MCS). The MCS system model was initially motivated by the expectation of deadline overruns caused by imprecise timing analysis, however in a swarm robotics system a common source of issues is that of wireless communication delays. Even in an otherwise well-formed network, transient faults can cause unpredictable sporadic delays to task execution.

In traditional automotive or avionics systems, criticality levels are kept insulated from each other on dedicated control units and networks to make it easier to demonstrate that lower-criticality tasks cannot interfere with the execution of higher-criticality tasks. In the motivating example from Section II, this is not possible due to the shared communication medium.

This work uses the most commonly deployed form of MCS in which the system has two criticality levels, *LO* and *HI*, although many refinements have been made since [2]. In the basic model, each task is assigned to one of these levels, resulting in a set of *LO* tasks and a set of *HI* tasks. Each task also has a WCET value (C) for each criticality level. A task's *LO* WCET (C_{LO}) might come from a simple measurement-based approach and so therefore may be optimistic. Its C_{HI} on the other hand might come from an analytical approach and so is safe, but pessimistic. For all tasks, $C_{LO} < C_{HI}$.

Whilst this approach was initially developed in the context of tasks executing on a CPU, it can be applied to wireless communications [3]. This paper shows that by thinking of the overall behaviour of a robot swarm in terms of criticality levels, the system performance can be gracefully degraded (for example as communication becomes less reliable) in a way that allows key functionality to be retained for longer.

IV. Related Work

Ad-hoc mesh networking is a well-studied problem in the field of autonomous systems. Early work such as BATMAN [4] looked at the problem of communicating nodes with a topology that is not known *a priori*. However it only provided limited support for mobility and little control over traffic prioritisation.

Real-Time Wireless Multi-hop Protocol (RT-WMP) [5] was a major expansion to the research space through the introduction of a real-time wireless protocol that could support a limited form of end-to-end guarantees on a multi-hop wireless network. RT-WMP includes a significant consensus phase in which nodes can coordinate to determine which has the most important traffic and over what routes it should be sent. This leads to predictable performance, but causes significant bandwidth overheads and requires nodes to expend a lot of power when compared to simpler protocols. Some of these problems have been later addressed through Beluga [6] which exploits flow periodicity to reduce coordination overheads, although this work still assumes fixed traffic flows and cannot support support simultaneous transmissions in different parts of the network.

WirelessHART [7] is an extension of the HART protocol focused on wired communication in industrial automation and process control. It uses time-division multiple access (TMDA) at the physical layer with centralised route planning. This allows it to achieve excellent levels of predictability, but in some situations it suffers from poor utilisation because it also cannot support simultaneous transmissions.

Glossy [8], and the wireless protocols that build upon it such as the Low-Power Wireless Bus [9] and Blink [10], use a different approach that assumes no network topology information is available. Instead, all packets are flooded through the network using simultaneous retransmissions within a single timeslot. With sub-microsecond clock synchronisation, the frames interfere constructively rather than destructively. Since network topology is not considered, Glossy requires transmission slots sufficiently large for such a flood to reach all nodes in the network. While this is acceptable for networks with a small number of maximum hops, it prevents efficient operation in larger networks. Further, the protocol cannot handle simultaneous transmissions.

AirTight [11] is a decentralised mixed-criticality protocol that provides real-time guarantees that are resilient to network interference. Access to the network is mediated by a slot table that is assigned ahead-of-time, but each node uses local scheduling decisions to determine which frame is sent during an assigned transmission slot. Unlike CPU scheduling applications, where the criticality level influences the worst case execution time, AirTight assumes that the size of a frame is constant. Instead, it is the level of interference a task must be able to endure that varies by criticality level. A "criticality-aware" fault model bounds the maximum level of interference for a given criticality level. The protocol then guarantees that the worst-case response times computed at a given criticality level will not be exceeded so long as the actual number of transmission failures experienced does not exceed the value predicted by the fault-model for that criticality level. If, at runtime, the failure bound computed by the fault model at a given criticality level is exceeded, the system moves to the next higher criticality level. Nodes in high criticality mode drop low criticality transmissions in order to increase the number of slots available for the high criticality flows. AirTight allows simultaneous transmissions [12], both from spatially separated nodes on the same channel, and through the use of multiple channels.

Inter-robot wireless communication in the field of swarm robotics often defaults to protocols such as WiFi, ZigBee, and Bluetooth. These all can offer potentially very high throughput with excellent robustness, but they rely on protocol or MAC-layer features like random backoff and retransmissions, and

result in priority inversion (where high-importance flows can be interrupted or blocked by low-importance flows). Additional robustness is sometimes added at the application layer, such as through distributed data structures that are resistant to imperfect communication [13], but these are typically not suitable for timing critical data. Prior work [14] illustrates how realistic, i.e. imperfect, communications can prevent swarm robotics applications from functioning correctly.

This paper will explore how swarm applications can fail due to wireless communication without timing awareness, and discuss how the application of AirTight and a mixed criticality system model can aid robustness.

V. Simulated Transmission Model

In order to demonstrate the use of these models, we employ an established swarm robotics simulator, ARGoS [15]. The ARGoS simulator provides integrated support for radio communication through the *simple_radio* interface, but this assumes perfect communications within a given range and does not handle packet collisions. In order to better model realistic communications, we have developed a custom radio communication plugin using an alternative transmission model

We assume 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 is able to 'hear' multiple frames within a single slot, we define that the frames interfere destructively such that no frame can be correctly decoded.

Our model is such that the effective packet delivery rate of a link is inversely proportional to the square of the distance between two nodes, and that successful or unsuccessful delivery is determined independently for each link and transmission. We note that is an extremely simple model which does not capture the true complexity of real wireless communications. Previous experiments [16]–[19] have produced conflicting results, but generally show a weak correlation between node distance and packet reception rates, which is highly dependent on the specific testing environment.

Attempting to accurately model a wireless radio's physical layer and resulting radio performance is beyond the scope of this paper, however our experiments do not rely on the specifics of the fault model beyond determining when packets are received. Rather we use this model to show how swarm behaviour changes as packet reception rates decrease. Therefore, we believe the results of the simulation should be broadly applicable regardless of the simplification to the fault model.

Using this simulation, we compare the performance of AirTight with two baseline protocols:

- Broadcast: Nodes broadcast each message a fixed number of times using carrier sensing to reduce collisions.
- Point-to-Point: Nodes transmit messages to each other node in turn, a CSMA/CA like protocol using carrier sensing and random backoff between retransmissions until an acknowledgement is received or a maximum number of retries has been reached.

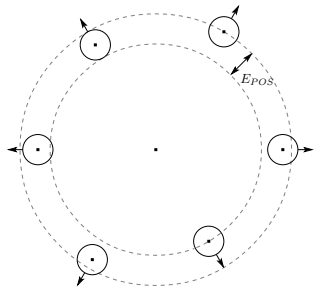

Fig. 2. Circle formation showing positional error E_{POS}, the maximum difference in distance from the circle's central point of any two nodes.

A. AirTight Fault Model

If arbitrarily many transmissions can fail over an indefinite time period, it is not possible to provide any timing guarantees. Therefore, the AirTight protocol [11] requires the number of failures within a time period to be bounded by a fault model that is provided ahead of time during response time analysis.

When considering networks with stationary nodes, the original AirTight specification considered failed transmission slots as a result of external interference causing blackout periods. Here, we instead consider failed transmission due to a reduced packet delivery rate (PDR), presumably as a result of distance between nodes.

Since the PDR is simply the delivery probability for a single packet, and the delivery of each packet is defined to be independent, our fault model must be probabilistic. The probability of a given number of failed transmissions occurring within a busy-period of t transmissions can be modelled using a simple binomial distribution. For a given criticality level L, minimum assumed PDR 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 inequality 1, in which we ensure the desired confidence bound is met by accumulating the probability of exactly k failures occurring for all $k \leq m$.

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

Note that the PDR is squared in the above equation to account for lost acknowledgements, which are assumed to occur with the same probability as lost data frames. We do not consider dependent PDRs, i.e. blackout periods due to external interference, in this model but note that it would be possible to extend this fault model to include them. In the most basic case, an additional fault-model accounting for static blackout periods with a length and period could simply be summed to the existing fault-model, albeit at the cost of some pessimism.

VI. Experimental Setup

Building on the motivating problem in Section II, we contrive a concrete instantiation. A set of 6 mobile nodes with multicoloured LEDs are arranged in a circle facing outwards,

1m apart, and pre-programmed to assume the same initial LED colour. To satisfy tasks T_{LED} and T_{POS}, all nodes should show the same LED colour and must move radially outward at the same speed in order to maintain the circle formation.

The LED colour and movement speed is determined in a decentralised manner by the nodes. Each node may change the LED colour for the entire swarm if it has been at least one second since it last initiated an LED colour change. This reflects real-world tasks such as a swarm responding to distributed sensing of an environment. When a node proposes an LED colour change, the swarm must coordinate to change to the new colour in unison within two seconds. Multiple such colour changes can be queued to take effect at different future times. If multiple nodes propose an LED colour change for the same time, 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 only be generated once every five seconds, and take effect after ten seconds. We assume that each frame is only large enough to contain either exactly one LED colour change message or one movement speed change message.

The performance of the system is quantified by measuring two errors: E_{POS}, the maximum difference in effective circle radius of any two nodes, as shown in Figure 2, and E_{LED}, the number of nodes showing an incorrect LED colour. If all nodes could communicate perfectly, E_{POS} and E_{LED} would remain zero.

In a real application, these changes would be triggered by local sensing onboard the nodes. For the purposes of this simulation, we simply assume that after the minimum inter-arrival period has been reached, there is a uniform probability of 2% per node per simulation step for LED changes, and a uniform probability of 1% per node per simulation step for movement speed changes. New LED colours are chosen as a uniformly random RGB value, whereas movement speed is chosen with each node having a bias towards a slower/faster speed, such that positions will diverge if communication fails.

As in Section II, the task T_{POS} is a high-integrity task that should be protected in the event of a system degradation. The AirTight protocol allows this by setting the packet flows of T_{POS} as high-criticality, while those of T_{LED} as low-criticality. The other two protocols have no notion of criticality[1], thus all packets are handled equally.

Since the circle setup results in a constantly increasing distance between agents, the packet delivery ratio is constantly decreasing according to our transmission model. Thus, regardless of protocol, communication between the nodes will eventually fail, causing E_{POS} to increase indefinitely. The effectiveness of a protocol can therefore be determined by the length of time that E_{POS} remains below a threshold value.

[1]Note that some carrier sense protocols have the notion of *priority* that allows smaller inter-frame times for packets that need low latency. This is not same as *criticality* which affects how the system degrades under failures.

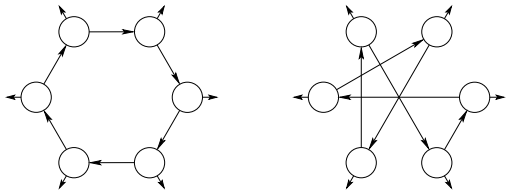

Fig. 3. Node setup showing optimal routing and an example of possible randomised routing.

A. AirTight

An AirTight deployment will analyse the flows and topology ahead of time to ensure that flows are schedulable [11]. For this example the protocol is set up with a slot-table of equal length to the number of nodes, with each node assigned a single unique transmission slot of 10ms. For each node, there are LED and movement flows that begin at that node and proceed around the outside of the circle. We assume that the system integrator wants to prioritise movement accuracy, so assigns LED flows to low-criticality and movement instructions to high-criticality. The overall LED flow is partitioned with the manually selected per-hop deadlines of 190ms, 310ms, 370ms, 430ms, and 550ms (giving an end-to-end deadline of 1850ms), whereas the movement flow is partitioned with the manually selected per-hop deadlines of 1570ms, 1750ms, 1870ms, 1990ms, and 2110ms (giving an end-to-end deadline of 9290ms). Note that these end-to-end deadlines are within the timing requirements as specified at the beginning of this section.

The AirTight protocol in this scenario has the inherent advantage that it has been pre-programmed with *a priori* knowledge of the network topology. To observe its behaviour when this advantage is removed we also test the AirTight protocol where the ordering of the nodes has been randomised, as shown in Figure 3.

The fault model (described in Section V-A) is configured to require confidence bound of 99% given a minimum PDR of 96.3% (corresponding to a distance between nodes of 2 metres) in low criticality mode, and a confidence bound of 99.999% given a minimum PDR of 79.2% (corresponding to a distance between nodes of 3 meters) in high criticality mode.

B. Broadcast

The broadcast protocol is setup to transmit each message 13 times. This number is chosen as the maximum fixed number of transmissions without exceeding the available bandwidth.

C. Point-to-Point

The point-to-point protocol is setup to transmit each message to each other node in turn, until it receives and acknowledgement frame or it has sent the frame 5 times. This maximum number of transmission was determined experimentally as a value that suitably balances the need for retransmissions without causing excessive collisions or triggering a continuous buildup of frames in the transmission buffers.

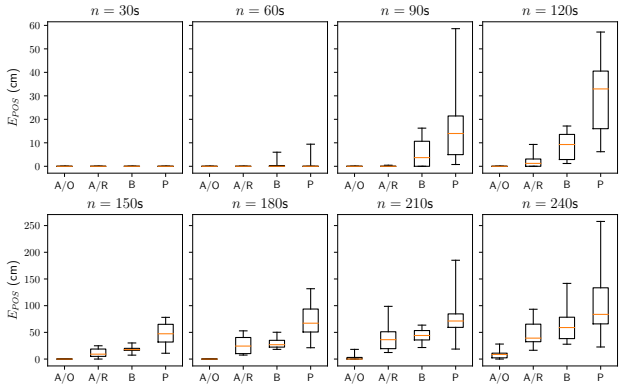

Fig. 4. E_{POS} after n seconds for simulated nodes using the AirTight protocol with optimal (A/O) and randomised (A/R) routing, Broadcast protocol (B), and Point-to-point protocol (P) over 10 different random seeds. Note that the y-axis scaling changes between rows.

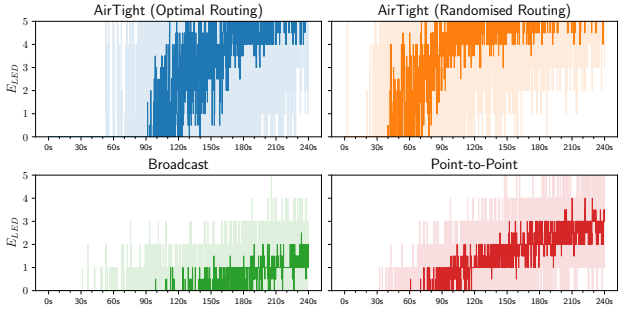

Fig. 5. Median E_{LED} after n seconds over for 10 different random seeds. Shaded area shows range from minimum to maximum E_{LED} value at the given step over the runs.

VII. Simulation Results

The simulation results for E_{POS} demonstrate that all protocols are able to keep $E_{POS} = 0$ for at least 30 seconds of simulation time (see Figure 4). After 60 seconds, both of the comparison protocols have accumulated small positional errors in some of the simulation runs, and by 120 seconds both show an error in the minimum case. The AirTight implementation with optimal routing maintains E_{POS} until after 180s. We note that by 170 seconds the circle radius has exceeded 4m in all simulation runs, meaning the input assumptions to the fault model (maximum distance of 3m) have been significantly exceeded. Thus, the errors that show in later time steps are as a result of "incorrect" input data rather than a violation of the protocols timing guarantees. The AirTight implementation with randomised routing shows a very small but non-zero maximum error at 90 seconds and larger errors from 120 seconds.

For the lower criticality flow, both comparison protocols show an initial LED error after approximately 30 seconds, 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. With optimised routing the AirTight protocol first shows an LED error after 50s, with a non-zero median first occurring after 90s. With randomised routing the first LED error occurs almost immediately[2], and regular errors start after 40s.

The AirTight protocol results in both higher median and maximum LED error values, particularly in later simulation steps. The protocol has allowed the system designer to prioritise motion control messages in the presence of errors, resulting in lower positional error at the cApplications:ost of less critical tasks.

VIII. Flocking Application

The circle problem clearly demonstrates the advantages that a mixed criticality system model can bring to unreliable shared-medium wireless communications in a swarm robotics context. Rather than causing unpredictable application-level faults, the mixed criticality model can be used to maintain service for longer in tasks of most importance. To illustrate this in a more realistic application, we present a flocking system in which a group of 10 nodes need to communicate to form and move as a flock. Each node is assumed to know its own position and orientation through some localisation system. Every five seconds, node should transmit their location and orientation to all other nodes such that each can compute it is own velocity to maintain its position in the flock [20].

We assume that each node is also performing some environmental sensing task, that generates up to three data frames per second that should be sent to an a priori designated sink node. We again assume that mobility control is the more important task, and so define this as a low priority flow in the AirTight implementation. We use the same AirTight fault model configuration as for the circle problem, but here define point to point flows from each node to all other nodes for communication. In the broadcast implementation the maximum number of retransmissions is reduced to 3, as this is here the maximum value while ensuring the available bandwidth is not exceeded. The configuration of point-to-point implementation is unchanged.

At each simulation time step, we sum the velocity vectors of all nodes, and compute the mean length of this vector over each simulation run of 30s. Since these vectors add destructively if nodes are moving in different directions (i.e. not as a flock), this value serves as proxy for the stability of flock. Since the distances between nodes once a flock has been formed remain relatively small, we further scale down the packet delivery rate of the fault model presented in section V by a constant factor. The results present in figure 6 show that AirTight is able to maintain an almost optimal flock velocity of approximately 6 cm/s at lower PDR scaling values than the two baseline comparison protocols. We note that this advantage is, as in the circle problem, a result of prioritising the positional information. As shown in figure 7, the performance of the data collection task under AirTight decreases rapidly when packet transmission is made less reliable.

[2]We note that with a starting radius of 1m, the maximum distance between robots does not exceed the input to fault model. This early error is the other 1% from the requested 99% confidence bound provided to the low criticality fault model.

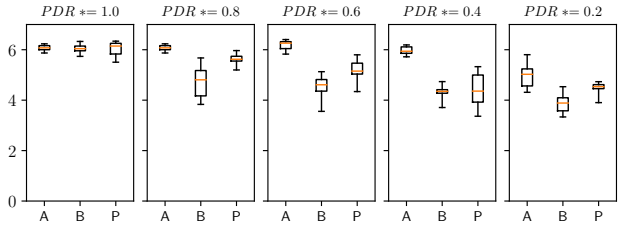

Fig. 6. Mean length of summed node velocity vectors at each time step over 10 simulation runs with different random seeds. PDR is modified by a constant factor from 1.0 to 0.2.

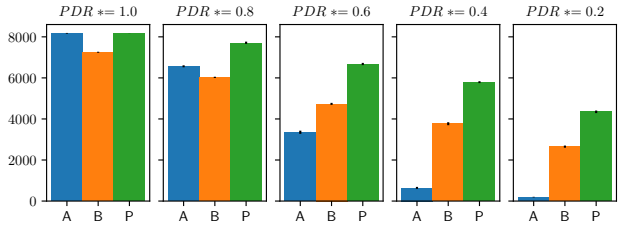

Fig. 7. Mean number of data samples received by the designated sink node over 10 simulation runs (error bars show min/max value). Note that broadcast protocol results in a lower number of received data samples even under good network conditions, as a lack of timing guarantees on delivery mean some frames are not yet delivered by the end of the simulation.

IX. Limitations and Future Work

In order to get the most benefit from the AirTight protocol [11], it requires *a priori* knowledge of the network topology and slot tables in order to determine packet routing and perform schedulability analysis of the flows. This requirement significantly impacts the ability of AirTight to be applied in many swarm robotics applications. The illustrated application in this paper presents a best case scenario since the network topology remains constant. The degraded performance observed when the node positions are randomised demonstrates the impact of the network topology not matching the prior assumptions, which would also occur if robots were permitted to move in ways that changes the network topology. In order to be able to apply the AirTight protocol to more general swarm robotics applications, the protocol must be extended such that routing can be updated at runtime. We intend to address this extension in future work.

X. Conclusion

In this paper we have demonstrated the value of using mixed criticality timing-aware wireless protocols in swarm robotics applications. Wireless communications will always be subject to error, but protocols such as AirTight allow the system designer to trade off errors intelligently, according to the relative importance of the various tasks in the system. Furthermore, such protocols allow timing behaviour to be analysed ahead of time, which allows guarantees to be made as to the timing correctness of a swarm robotics application within parameters provided by a fault model. In future work we intend to extended existing real-time protocols like AirTight in ways that allow it to specifically target swarm robotics applications.

References

[1] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[2] A. Burns and R. I. Davis, "A survey of research into mixed criticality systems," *ACM Comput. Surv.*, vol. 50, nov 2017.

[3] C. Xia, X. Jin, L. Kong, and P. Zeng, "Bounding the demand of mixed-criticality industrial wireless sensor networks," *IEEE Access*, vol. 5, pp. 7505–7516, 2017.

[4] D. Johnson, N. Ntlatlapa, and C. Aichele, "Simple pragmatic approach to mesh routing using batman," in *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries*, 2008.

[5] 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*, pp. 1–11, IEEE, 2007.

[6] D. Tardioli, "A wireless communication protocol for distributed robotics applications," in *2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 253–260, 2014.

[7] I. E. Commission *et al.*, "Industrial networks—wireless communication network and communication profiles—wirelesshart (iec 62591: 2016)," *IEC: Geneva, Switzerland*, vol. 3, pp. 1–1043, 2016.

[8] 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*, pp. 73–84, 2011.

[9] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, (New York, NY, USA), p. 1–14, Association for Computing Machinery, 2012.

[10] 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, feb 2017.

[11] 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)*, pp. 65–75, 2018.

[12] J. Harbin, A. Burns, R. I. Davis, L. S. Indrusiak, I. Bate, and D. Griffin, "The airtight protocol for mixed criticality wireless cps," *ACM Trans. Cyber-Phys. Syst.*, vol. 4, dec 2019.

[13] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "A tuple space for data sharing in robot swarms," *EAI Endorsed Transactions on Collaborative Computing*, vol. 2, 5 2016.

[14] 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)*, pp. 101–109, 2021.

[15] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[16] A. Cerpa, N. Busek, and D. Estrin, "Scale: A tool for simple connectivity assessment in lossy environments," tech. rep., September 5 2003.

[17] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, (New York, NY, USA), p. 1–13, Association for Computing Machinery, 2003.

[18] N. Baccour, A. Koubâa, M. Ben Jamâa, D. do Rosário, H. Youssef, M. Alves, and L. B. Becker, "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.

[19] K. Brun-Laguna, A. L. Diedrichs, D. Dujovne, R. Léone, X. Vilajosana, and T. Watteyne, "(not so) intuitive results from a smart agriculture low-power wireless mesh deployment," in *Proceedings of the Eleventh ACM Workshop on Challenged Networks*, CHANTS '16, (New York, NY, USA), p. 25–30, Association for Computing Machinery, 2016.

[20] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *SIGGRAPH Comput. Graph.*, vol. 21, p. 25–34, aug 1987.