Adaptive Application Behaviour for Robot Swarms using Mixed-Criticality

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

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

Communication is a vital component for all swarm robotics applications, and even simple swarm robotics behaviours often break down when this communication is unreliable. Since wireless communications are inherently subject to interference and signal degradation, real-world swarm robotics applications will need to be able handle such scenarios. This paper argues for tighter integration of application level and network layer behaviour, so that the application can alter its behaviour in response to a degraded network. This is systematised through the implementation of a mixed-criticality system model. We compare a static application behaviour with that of an application that is able to alter its behaviour in response to the current criticality level of a mixed-criticality wireless protocol. Using simulation results we show that while a static approach is sufficient if the network conditions are known a priori, a mixed-criticality system model is able to adapt application behaviour to better support unseen or unpredictable conditions.

1 Introduction

Swarm robotics platforms usually rely on wireless communications, which by their nature can be unreliable and exhibit unpredictable timing characteristics. The standard approach is to use best-effort protocols with sufficient redundancy so that all important traffic can be conveyed. Conventional protocols such as WiFi, ZigBee, and Bluetooth offer good throughput and robustness, but they rely on features like random backoff and retransmissions which can be disqualifying when attempting to build high-criticality systems that rely on timing correctness. There are similarly a range of swarm-oriented protocols such as Glossy [7], and the related Low-Power Wireless Bus [6] and Blink [17] which all remain topology-agnostic by flooding the network with packets and retransmissions in order to maximise connectivity at the expense of throughput and power-efficiency.

Accordingly there has been a more recent trend towards protocols which are timing-aware, such as WirelessHART [5] and AirTight [3]. These approaches require a priori information about the system, but compensate by allowing for greater timing confidence. They still, however, suffer from the inherent unreliability of wireless links.

This work combines ideas from these timing-aware protocols and from the real-time systems domain to argue for application-level adaptive behaviour to support reliability in the presence of errors and unreliable communications. If a swarm robotics platform employs a wireless protocol that can provide feedback as to whether communication reliability requirements are currently being met, the application can be designed to adapt its own behaviour in order to recover connectivity if packet delivery rates fall below a predefined threshold. This paper argues for the use of a mixed-criticality approach, in which the application switches between two behaviour modes depending on the network conditions.

A. Ferrando, R.C. Cardoso (Eds.): Third Workshop on
Agents and Robots for reliable Engineered Autonomy 2023 (AREA'23)
EPTCS 391, 2023, pp. 71–82, doi:10.4204/EPTCS.391.9

2 Application Adaptivity and Mixed Criticality Systems

A common claim is that robot swarms achieve fault tolerance and redundancy through the size of the swarm. However as is well-understood in the research, communication failures are particularly problematic and can actually be compounded by increasing swarm size [2]. Swarm applications must be designed with the reliability of communications as a priority. Robotic swarm programming models like Buzz [10] allow for developers to discuss higher level concepts such as moving groups of robots as one, but frameworks which tie reliability requirements from the application developer to the reliability state of the running system are less formalised.

This work argues that in order to address this issue, a swarm robotics system with unreliable communications should be viewed as a Mixed Criticality System (MCS), and that such a characterisation can be used to explicitly guide the system's behaviour at times of overload or error. The MCS model is a commonly-deployed system model in the domains of high-integrity or safety-critical systems, and is used to provide guarantees on the behaviour and timing of the system in the presence of unreliability. Informally, the developer provides a description of their system, along with an "importance" for each task, and the guarantees that are required to support these tasks at each criticality level. If the system enters a state when it can no longer guarantee the "important" parts, then a graceful degradation is codified into the model.

The original formulation of MCS [15] was aimed at CPU scheduling problems and defined the system to have two criticality levels: *LO* and *HI*. Each task/process/job is designated either *LO* criticality or *HI* criticality: *HI* tasks perform safety-critical functions and are required for safety assurance, whereas *LO* tasks are less important and may occasionally fail or miss their deadlines. The response time analysis of CPU scheduling requires tasks to be assigned an estimate of their worst-case execution time (WCET) a priori, but determining this value proves difficult in practice. If a task's WCET comes from estimation or measurement it may be optimistic (i.e. not a true worst-case time). This could result in important tasks missing their deadlines if the true WCET at runtime is larger than the assumed value. A task's WCET may also come from a safe, analytical approach which is guaranteed to cover the worst-case but in practice this might be very pessimistic. This could lead to the application designer falsely believing that the system is not schedulable without greater hardware resources, which may in turn be infeasible for other reasons (cost, energy use, etc).

The key insight of the MCS model is that the optimistic execution times will be correct most of the time, and the true worst-case is usually only seen in rare situations. Tasks are therefore assigned two estimates of their worst-case execution time, a *LO* WCET and a *HI* WCET. A system can proceed at the *LO* criticality level most of the time, assuming that tasks will exhibit their expected *LO* WCETs. However, when any task exceeds its *LO* WCET, the system enters the *HI* criticality state and stops executing any *LO* tasks. This means that to guarantee the safety of a system in the presence of unreliability, it is necessary to prove only that the *HI* tasks will not exceed their *HI* WCET times. The application adapts its behaviour based on the current state of the system by dropping less important activities and focusing on only the most important system guarantees.

Prior work has applied mixed-criticality scheduling to wireless networks by assuming that it is the level of interference (i.e. the assumed maximum number of retransmissions required within a time period) that varies by criticality level [3]. This paper takes this idea and applies it to both the transmission of packets in a wireless communications swarm, but also to the behaviour of that swarm. Instead of treating criticality as just a feature of the scheduling and communications layer, this work argues that the application itself can respond to such changes usefully, based on developer-provided criticality requirements, in order to preserve the performance of the overall swarm. Such a characterisation can be exploited to allow for applications which assume communications will mostly operate well, and so behave accordingly, but can respond appropriately when, for whatever reason, communicating tasks start to fail to meet their deadlines. In Section 6 we describe an example of such an application, with Section 6.1 describing how the application modifies its own behaviour based on feedback from the network layer.

3 Prior Work

In existing research, unrealistic assumptions are often made about the reliability of wireless communications, such as assuming perfect transmission within a given radius. Prior research has shown that imperfect communication results in swarm behaviours breaking down [12], and so it is therefore not always appropriate to simply ignore the communications medium and rely on TCP to transmit data 'eventually'.

The MCS system model has been applied to industrial wireless communications [16] to show that such an approach can give more reliable timing bounds by adapting the network routing based on the system's criticality level. This work provides better timing bounds, but is restricted to considering the communications layer only.

Our previous work has argued for the use of an MCS wireless protocol in swarm robotics applications [13]. The system initially operates in *LO* criticality mode, and if the level of successful message transmissions degrades past some threshold this is detected and the network switches to *HI* criticality mode. The result of this change is typically to drop or deprioritise less important traffic. While this allows the application designer to preserve some behavioural elements in the presence of partial network degradation, it is inherently limited to preserving a subset of the full behaviour, and will eventually fail if network conditions continue to degrade.

The AirTight protocol [3] considered in that work, being purely a point-to-point protocol, is impractical for real swarm robotics applications. We therefore introduce a model (Section 4) and simulation implementation (Section 5) for a new protocol that can provide mixed criticality behaviour over broadcast transmissions.

The work in this paper extends this idea to consider the how the application's *overall* behaviour can adapt to criticality. Specifically, we enable an application to observe the network layer's criticality level, so that the application can modify its own behaviour in response to current network conditions (Section 6.1).

4 Communication Model

This paper argues for a closer integration of an application implementation with a mixed-criticality network MAC layer, specifically by allowing the application to be aware of the MAC layer's current criticality mode. This allows the application to detect when the network is no longer able to guarantee reliable message delivery due to communication faults, enabling it to adjust its behaviour to prevent further degradation. We compare the effect allowing a robot to use this criticality mode information to locally adjust its cohesion factor against simply using a static cohesion factor, as described in Section 6.1.

We consider a network of N homogeneous robot nodes in which all transmissions are assumed to be broadcasts that should reliably reach each other node. The network provides multiple buffers of configurable priority and criticality such that the application designer can determine the order of transmissions. In each transmission slot, the node chooses to transmit the first message from the highest priority non-empty buffer at, or above, the current criticality level.

The network layer observes the number of "failed" transmissions, defined as transmissions after which it cannot prove that all nodes have received its last message. If the number of failed transmissions within a busy-period exceed a predefined threshold, the network switches to *HI* criticality mode. Messages in *LO* criticality buffers are discarded if the node enters *HI* criticality mode. Time sensitive messages in *HI* criticality buffers can optionally be assigned a maximum time-to-live (TTL), such that they expire and are not retransmitted after a given time period has elapsed since they were first queued.

5 Communication Implementation

In order to implement the behaviour described by the model in Section 4, we introduce an implementation of AirTight [8] modified to use broadcast transmission rather than point-to-point links. AirTight is a slot-based real-time, mixed criticality protocol which can guarantee the time-sensitive transmission of a set of traffic flows using ahead of time analysis under a given fault rate assumption.

The network runs using time-division multiplexing over 10ms time slots. There is a periodic slot table assigned a priori, such that the slot table is of length N and each node has exactly one exclusive transmission slot. It is assumed that background clock synchronisation takes place such that nodes agree on the current slot.

Each node maintains a bit-field 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 (excluding the node's own transmission slot). This bit-field is included in the header of each transmission as a type of delayed acknowledgement, 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 bit-field of length $N-1$ in which each bit encodes whether confirmation of successful delivery has been received from a corresponding other node. Upon reception of a frame, a node can thereby set the transmitting node's bit in its last transmission buffer's bitfield if the received header indicates that the transmission was received. Once all bits have been set, the message at the head of the buffer has been delivered to all other nodes. The message can then be removed from the buffer and the bit-field is cleared.

The node keeps two counters: a counter of the length of the current busy-period, and a counter of the number of retransmissions. The busy-period counter is incremented on all transmissions, whilst the retransmission counter is incremented whenever the node rebroadcasts a frame that had previously been transmitted. If the number of retransmissions exceeds a given threshold, the node switches to *HI* criticality mode.

During its assigned transmission slot, a node broadcasts a single frame from the highest priority non-empty buffer at or above the current criticality level. If no such frame exists the counters are reset and, if the node was in *HI* criticality mode, it switches back to *LO*. The node then broadcasts a no-op frame such that the bit-field used for delayed acknowledgements is always transmitted. The real-time timing analysis of this implementation is future work but can be based on the structure of the original AirTight analysis.

The implementation of the simulation plugin, communications layer, and other artefacts associated with this paper can be found in our code repository [1].

[1] https://github.com/yorkrobotlab/argos3-airtight

6 Exploration Application

We consider an autonomous wireless swarm robotics platform in which a swarm of robots should collaborate to explore an unknown area. The environment is partitioned into a two-dimensional grid. Robots should visit each cell in the grid and detect if there is an object present in that cell, building a map of clear/occupied cells in the area. It is assumed that obstacles are stationary, such that it is unimportant when a robots visits the cell, or whether a cell is visited multiple times by one or more robots. Each robot has its own local copy of the map, which it should complete for all accessible cells[2]. Robots can communicate the sensed values for a given cell over the network, allowing other robots to insert this value into their map without needing to visit the cell.

For a concrete instantiation we consider a set of 10 Pi-Puck [9] robots exploring a 6x6m grid of 10x10cm square cells. Pi-Puck robots only have simple infrared range-finder sensors that 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 robot is assumed to be able to determine its own location, which it must communicate to the other robots to avoid such near-collisions. Robots cannot store their complete location history, so they cannot retrospectively determine that incorrect data may have been sensed where position messages have been lost or delayed. These 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 robot behaviour is loosely inspired by an existing algorithm [14], but adapted to the much simpler Pi-Puck robots and to much reduced communication ability. It is implemented by picking and driving towards a target cell, which is always chosen as one of the nine cells within a 3x3 grid centred on the robot's current position. A new target cell is chosen once the current target is reached, or the robot encounters an obstacle such that it deems the current target to be unreachable. The 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 6.1.

[2] The positioning of obstacles in the environment may render some cells inaccessible.

6.1 Robot Cohesion

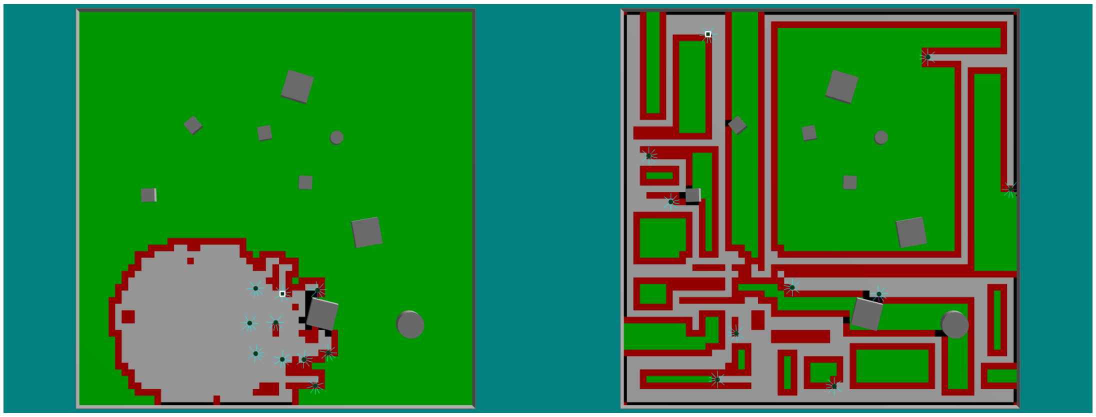

Figure 1: Visualisation of exploration state after 95s when applying constant cohesion (left) compared with no cohesion (right) under perfect network conditions. Explored cells are shown in grey if empty or black if containing/bordering an obstacle; unexplored cells are shown in red when bordering explored cells, else in green.

The effect of adding a cohesion element to the target cell selection depends on the environmental factors encountered by the robots. Under perfect communication, robot cohesion reduces the overall application performance. This can be intuitively understood by considering that robots that disperse maximally will not block each other and can greedily explore new cells. The closer the robots are pulled together, the more often robots may need to change direction (for example by revisiting already explored cells) in order to avoid violating the minimum separation constraint. As shown in Figure 1, this may result in robots towards the rear of a group being surrounded by already explored cells.

With a communication model that deteriorates with distance, the performance of a solution where robots disperse decreases, since robots are unable to receive the sensing information of other robots. In extreme cases, this could effectively result in each robot needing to explore the entire area. There is therefore a trade-off in which some amount cohesion is useful to preserve communication, but too much cohesion decreases performance by limiting the exploration ability. The optimal level of cohesion depends on the properties of the wireless medium, which in a real scenario may not be known a priori.

In this paper, we argue for application adaptation based on the state of the network, by applying cohesion weighting on target cell selection using a mixed-criticality approach. While the network layer is in *LO* criticality mode the robots can disperse to maximise their exploration potential, before being pulled back towards each other if/when the network changes to *HI* criticality mode. Once the network has recovered, the robots can then resume exploring. This results in an equilibrium that allows the robots to adapt their behaviour to the encountered conditions. We compare the effects of the following four ways of applying the cohesion weighting:

- No cohesion: The cohesion weight is completely disabled.
- Constant cohesion: A 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 weighting when the network protocol has been in *HI* criticality mode at any point within the last three seconds.

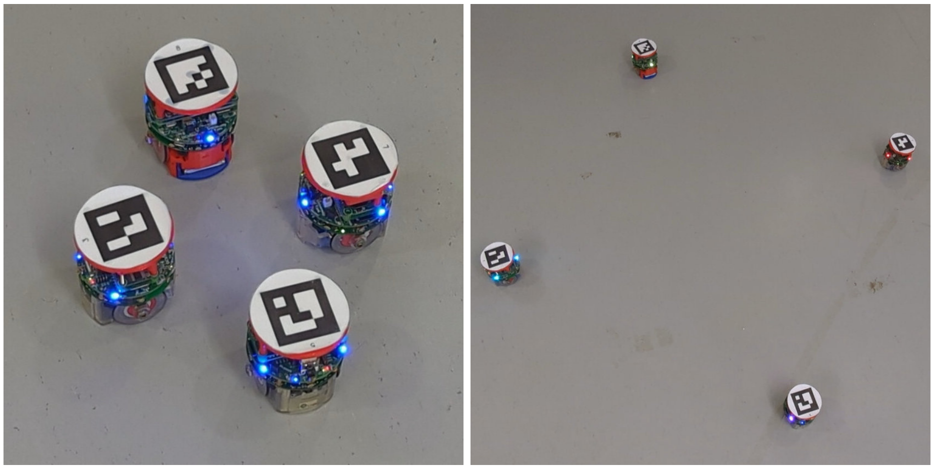

Figure 2: The running "physicalised simulation" of an earlier experiment on the Pi-Puck robots. The simulation shows the LED synchronisation breaking down as the robot separation increases.

7 Simulation Setup

7.1 Simulation Framework

Our simulation setup is based on the popular ARGoS robot simulator [11]. An extended version of a custom networking plugin [13] implements the network simulation capabilities. Each simulation step is assumed to correspond to a network level transmission slot in which a robot can attempt to send or receive a single message. A simulated transmission model controls which messages are successfully received.

In order to partially validate our simulation framework, we have implemented a "physicalised simulation" of our earlier experiments using real Pi-Puck robots (Figure 2). By this we mean an implementation on physical robots, but where some key components are still simulated. Specifically, we use Pi-Puck robots communicating over IEEE 802.15.4 provided by XBee modules, whilst simulating the packet loss by artifically discarding some messages according to the packet delivery rates provided by the transmission model. Since this still relies on a simulated transmission model, the results between the full simulation and physicalised simulation are very similar. Due to the logistical challenges of running larger scale experiments with physical robots, we have not yet implemented such a physicalised simulation for the current experiments.

7.2 Robot Configuration

Each robot is configured with two network buffers, for position messages and cell status message. A position message contains the robots target location, while a cell status message encodes a single cell to be either clear or containing an obstacle. Since the focus of these experiments is on the effect of application level behaviour changes based on the network criticality level, we configure both buffers as *HI* criticality to prevent criticality changes having an impact at the network level. Positional data is time sensitive to ensure the minimum robot separation is preserved, and so this buffer is configured with the higher priority. Positional messages are set to be retried for a maximum of 0.8s to prevent old position data filling up transmission buffers, whilst cell status messages are set not to expire since these are not time sensitive.

7.3 Arena Generation

We randomly generate 100 simulated arenas, comparing the exploration performance of each cohesion and transmission model configuration combination across these arenas. The robots are placed first, by distributing them within a 1.8m by 1.8m square centred around a random point in the area, whilst ensuring each robot placed at least 30cm away from the next closest robot. We then distribute up to 17 obstacles across the arena, again requiring a minimum separation from each other obstacle and each robot starting position.

7.4 Transmission Model

Packet delivery is determined by a simulated transmission model in which successful or unsuccessful delivery is determined independently for each transmission and each potential receiving node. The packet delivery rate is assumed to be fixed to a maximum of 95% for distance of less than 0.5m, after which the packet delivery rate is inversely proportional to the square distance between the nodes, subject to an additional constant scaling factor (k). By modifying this scaling factor, the effect of the packet delivery rate on the application behaviour can be observed.

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

Where:

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

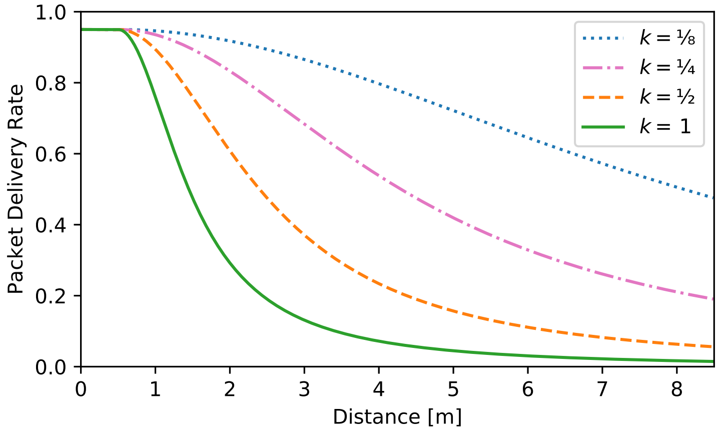

Figure 3: Transmission model defining packet reception probability in relation to distance, for scaling factors $k = \frac{1}{8}$, $k = \frac{1}{4}$, $k = \frac{1}{2}$, and $k = 1$.

This is a simple model that cannot capture the true complexity of real-world wireless communications. Prior research [1, 4] 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. describes this as the “connected” region [1], which is followed by “transitional” and “disconnected” regions where packet delivery first becomes intermittent and then mostly unsuccessful. This fits well with the mixed criticality system model, which inherently assumes that the system will operate under its optimistic assumptions most of the time, before encountering some kind of state change that requires rectification.

The implementation here does not rely on specific details of the transmission model, and the criticality response merely assumes that moving closer to other nodes is likely to improve packet reception rates. We therefore believe the simulation results should be broadly applicable regardless of the selected transmission model.

8 Results

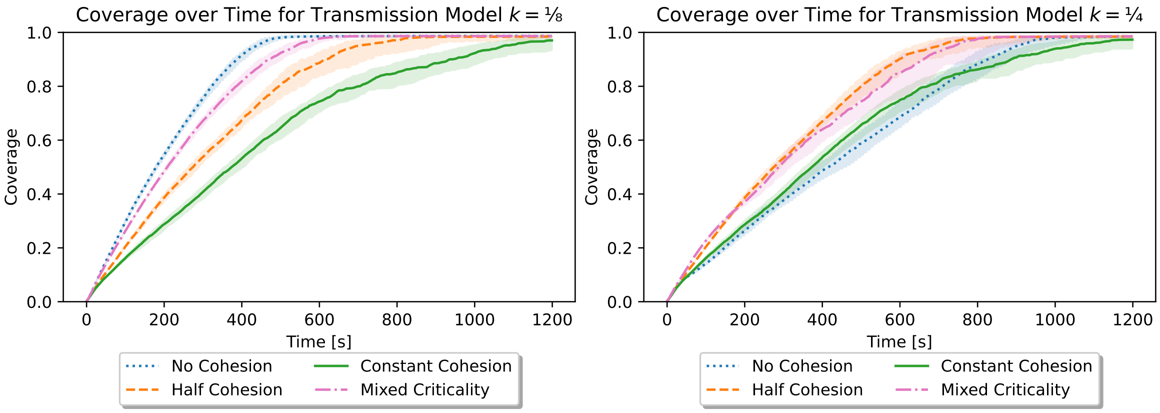

Figure 4: Coverage over time across the four cohesion modes for transmission model scaling factors $k = \frac{1}{8}$ and $k = \frac{1}{4}$. Line shows median value; shaded region shows interquartile range across 100 simulation runs.

With the transmission model scaling factor configured for the most gradual dropoff in packet delivery rates, $k = \frac{1}{8}$, the simulation results in Figure 4 shows that not applying a cohesion force results in the highest performance. At this low dropoff rate the robots can adequately communicate across the entire arena such that there is no advantage to moving as a cohesive group. The mixed criticality configuration can spend a significant proportion of its runtime in *LO* criticality mode (where no cohesion is applied) and thus displays better performance than the half cohesion or constant cohesion configurations.

When the transmission model scaling factor is increased to $k = \frac{1}{4}$ a maximal dispersion of the nodes begins to impede communication between the nodes, creating an advantage to applying some level of cohesion. The mixed criticality and half cohesion configurations perform similarly to each other. The no cohesion configuration starts to suffer from the aforementioned communication issues, while the performance of the constant cohesion configuration is still reduced from applying a stronger cohesive force than necessary.

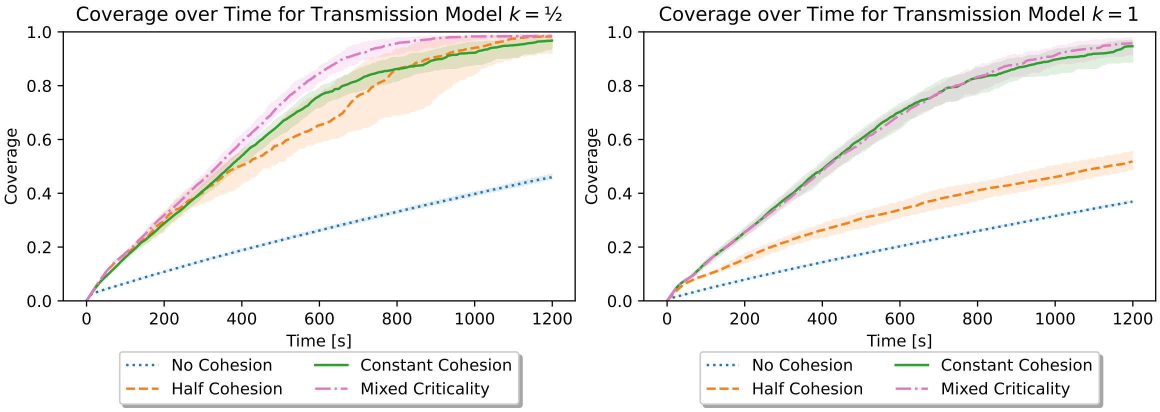

Figure 5: Coverage over time across the four cohesion modes for transmission model scaling factors $k = \frac{1}{2}$ and $k = 1$. Line shows median value; shaded region shows interquartile range across 100 simulation runs.

A further increase of the transmission model scaling factor to $k = \frac{1}{2}$, shown in Figure 5, causes the no cohesion configuration to break down. The performance of the half cohesion configuration is also decreased, showing a fairly wide spread depending on runtime circumstances. The mixed criticality configuration is able to adapt to the conditions, yielding the highest performance for this transmission model.

At the maximum tested dropoff rate, with the transmission model scaling factor set to $k = 1$, the mixed criticality mode and constant cohesion modes perform very similarly, with the low packet delivery rates causing the mixed criticality configuration to spend large proportions of its time in *HI* criticality mode where cohesion is applied. The half cohesion and no cohesion modes both show poor performance, with the half cohesion mode no longer resulting in a sufficiently tight formation to maintain communication between the nodes.

The constant cohesion configuration is mostly unaffected by the transmission model's scaling factor. While this results in comparatively good performance when communications are limited to short ranges, it cannot take advantage of long communication ranges under good radio conditions. The half cohesion configuration provides comparatively good performance for the two middling transmission models, but neither takes full advantage of long communication ranges nor does it cope with very short communication ranges.

The mixed criticality mode is better able to adjust to different communication dropoff rates, and can do so in a way that does not require the application layer to understand the specific issues that are happening at the networking layer. While a static cohesion factor can provide similar levels of performance for any given transmission model, this requires the transmission characteristics to be known beforehand and to remain static. For a real-world application this is unlikely to be the case. A static cohesion factor therefore requires the application designer to make a tradeoff between performance under good networking conditions and reliability under network degradation. A mixed criticality approach avoids this tradeoff by allowing the application to observe the true conditions at runtime.

9 Limitations and Future Work

The communication protocol presented in this paper is developed to study the effect of adjusting application behaviour based on a network layer criticality level. Compared to a complete protocol it is lacking in several aspects, most importantly the absence of formal timing analysis that can provide guarantees on the network performance. In future work we intend to develop a real-time mixed-criticality protocol that supports network layer and application integration while being suitable for swarm robotics and providing such timing guarantees.

We also intend to further study the two-way relationship between application behaviour and network MAC layer. In the implementation provided in this paper, the application is aware of the current network layer criticality level, and has been programmed that cohesion should be applied only when the network layer is in *HI* criticality mode. Beyond this simple rule, however, it could be imagined a more intelligent application could attempt to predict the effect of its future behaviour on the network. This could allow the application to either modify its planned behaviour to avoid network issues, or warn the network layer such that it could prepare for a drop in packet delivery rates.

10 Conclusion

In this paper we have presented a mixed-criticality approach to swarm robotics application behaviour. Mixed criticality is widely applied in real-time CPU scheduling and network protocols, but has so far not been widely applied to swarm robotics. The parameters controlling a robot's behaviour may have different optimal values depending on the conditions encountered by the robot. Communication conditions at runtime can vary from those expected by the application designer in ways that are often opaque to the application, placing stress on the network and leading to a loss of real-time performance. A mixed-criticality approach allows a systematic way for the application to both define what is considered of higher and lower importance, and then respond in a way that prioritises resources appropriately.

Our simulation results show that a swarm robotics application using a mixed-criticality approach to adjust its behaviour to match the encountered conditions can be made more robust than one that uses a static configuration. In future work we intend to develop mixed-criticality network protocols targeted at swarm robotics applications and further study the integration of network and application behaviour.

References

[1] Nouha Baccour, Anis Koubâa, Maissa Ben Jamâa, Denis do Rosário, Habib Youssef, Mário Alves & Leandro B. Becker (2011): *RadiaLE: A framework for designing and assessing link quality estimators in wireless sensor networks*. *Ad Hoc Networks* 9(7), pp. 1165–1185, doi:10.1016/j.adhoc.2011.01.006.

[2] Jan Dyre Bjerknes & Alan FT Winfield (2013): *On fault tolerance and scalability of swarm robotic systems*. In: *Distributed Autonomous Robotic Systems: The 10th International Symposium*, Springer, pp. 431–444, doi:10.1007/978-3-642-32723-0_31.

[3] Alan Burns, James Harbin, Leandro Indrusiak, Iain Bate, Rob Davis & David Griffin (2018): *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, doi:10.1109/RTCSA.2018.00017.

[4] Alberto Cerpa, Naim Busek & Deborah Estrin (2003): *SCALE: A tool for Simple Connectivity Assessment in Lossy Environments*. Technical Report. Available at `https://escholarship.org/uc/item/2g49z78g`.

[5] International Electrotechnical Commission et al. (2016): *Industrial Networks—Wireless Communication Network and Communication Profiles—WirelessHART (IEC 62591: 2016)*. *IEC: Geneva, Switzerland* 3, pp. 1–1043.

[6] Federico Ferrari, Marco Zimmerling, Luca Mottola & Lothar Thiele (2012): *Low-Power Wireless Bus*. In: *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, Association for Computing Machinery, New York, NY, USA, p. 1–14, doi:10.1145/2426656.2426658.

[7] Federico Ferrari, Marco Zimmerling, Lothar Thiele & Olga Saukh (2011): *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.

[8] J. Harbin, A. Burns, R. I. Davis, L. S. Indrusiak, I. Bate & D. Griffin (2019): *The AirTight Protocol for Mixed Criticality Wireless CPS*. *ACM Trans. Cyber-Phys. Syst.* 4(2), doi:10.1145/3362987.

[9] Alan G. Millard, Russell Joyce, James A. Hilder, Cristian Fleşeriu, Leonard Newbrook, Wei Li, Liam J. McDaid & David M. Halliday (2017): *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)*, pp. 741–748, doi:10.1109/IROS.2017.8202233.

[10] Carlo Pinciroli & Giovanni Beltrame (2016): *Buzz: An extensible programming language for heterogeneous swarm robotics*. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 3794–3800, doi:10.1109/IROS.2016.7759558.

[11] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella & Marco Dorigo (2012): *ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems*. *Swarm Intelligence* 6(4), pp. 271–295, doi:10.1007/s11721-012-0072-5.

[12] Mark Selden, Jason Zhou, Felipe Campos, Nathan Lambert, Daniel Drew & Kristofer S. J. Pister (2021): *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, doi:10.1109/MRS50823.2021.9620611.

[13] Sven Signer, Alan G. Millard & Ian Gray (2022): *Mixed-Criticality Wireless Communication for Robot Swarms*. In: *Proceedings of the 9th International Workshop on Mixed Criticality Systems*, pp. 20–25. Available at `https://wmc2022.github.io/assets/WMC_2022_Proceedings.pdf`.

[14] Vu Phi Tran, Matthew A. Garratt, Kathryn Kasmarik, Sreenatha G. Anavatti & Shadi Abpeikar (2022): *Frontier-led swarming: Robust multi-robot coverage of unknown environments*. *Swarm and Evolutionary Computation* 75, p. 101171, doi:10.1016/j.swevo.2022.101171.

[15] Steve Vestal (2007): *Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance*. In: *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, pp. 239–243, doi:10.1109/RTSS.2007.47.

[16] Changqing Xia, Xi Jin, Linghe Kong & Peng Zeng (2017): *Bounding the Demand of Mixed-Criticality Industrial Wireless Sensor Networks*. *IEEE Access* 5, pp. 7505–7516, doi:10.1109/ACCESS.2017.2654483.

[17] Marco Zimmerling, Luca Mottola, Pratyush Kumar, Federico Ferrari & Lothar Thiele (2017): *Adaptive Real-Time Communication for Wireless Cyber-Physical Systems*. *ACM Trans. Cyber-Phys. Syst.* 1(2), doi:10.1145/3012005.