Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

A SIMULATION TESTBED FOR THE ANALYSIS OF DESIRED EMERGENT PROPERTIES IN SYSTEMS RUNNING IN DYNAMIC AND CONTESTED ENVIRONMENTS

Claudia Szabo
Dustin Craggs

School of Computer Science
The University of Adelaide
North Terrace
Adelaide, 5005, AUSTRALIA

Wayne Johnson
Gregory Judd
Keith French

Defence Science Technology Group
Third Avenue
Edinburgh, 5111, AUSTRALIA

ABSTRACT

Designing complex systems with a specific emergent property is challenging even in perfect environmental conditions, and several additional challenges are introduced in contested, dynamic environments such as ongoing battles or military exercises. An example of a desired emergent property is disseminating information in a timely manner without causing significant network disruptions. While military communication strategies exist, their evaluation is performed in costly real-life deployments with limited exploratory capability. In this paper, we propose to achieve timely information dissemination through a rule-based expert system that adapts to network and operational context changes by change the priority of messages sent to the network. To analyse this emergent behavior, we present an evaluation simulation testbed that explores various operational conditions for the evaluation of communication strategies in a land-exercise focused scenario.

1 INTRODUCTION

Complex systems exhibit behaviors that cannot be easily reduced to their individual components alone, and require in depth analysis at system and component level (Mogul 2006, Szabo and Teo 2016). The analysis of these *emergent properties* is crucial as systems grow in size, coupling, and geographic distribution (Mogul 2006, Bedau 1997, Mittal 2013, Szabo and Teo 2016) but becomes critical as systems are deployed in environments that are contested and with significant resource constraints. In a military context, beyond modeling complex physical and hardware environments, there is a need to understand interactions between people (e.g., friendly forces, adversaries, or civilians), business processes and organizations, as well as social groups and individual agents. Critical to military operations in this environment is the digital network (Campbell 2018), and a crucial aspect of its effectiveness are the Information Management and Dissemination (IMD) strategies.

An example of a desired emergent property in the IMD context is assuring the integration of Intelligence, Surveillance and Reconnaissance information in contested and dynamic environments, with poor network conditions due to the use of low powered radios that are potentially disrupted either by the enemy or by forces of nature. For example, soldiers in battle in a foreign and contested terrain are unlikely to have access to reliable high bandwidth communication networks and must rely on more austere military radio-based ad-hoc networks, on the move over the physical terrain. Due to the nature of battle, soldiers might not have a complete view of the position or status of their platoon and thus critical information dissemination decisions that will affect system-wide properties have to be made with incomplete and unreliable information. In these

scenarios, entities have complex operational behaviors dictated by the mission battle plan, but also need to be responsive to frequent network capability changes due to their operation in a disrupted environment with limited infrastructure. Beyond individual entity behavior that is sensitive to operational and network context changes, each entity sends in the network a wide variety of messages, such as Position Location Information (PLI) messages, which detail the position information of the sending entity, status information and other non-critical messages. There is a need for assurance of timely delivery, responsiveness as well as ensuring graceful degradation mechanisms that can manage the fragility of the communication networks.

To achieve this, quality of information (QoI) attributes (Suri et al. 2015) can be used as a means of prioritizing information flows. However, the creation of this QoI based prioritization is challenging as information priority needs to change dynamically as the mission unfolds and context changes. In addition, unforeseen but serious events, such as deliberate network disruption, can change the communication strategy in terms of destination and message priority. A potential solution is to design the system as a complex system where each node has an understanding (partial or full) of the mission and network context and is able to reason over that context to prioritize and control the information for dissemination. There are numerous approaches that design systems with emergent properties across the agent-based simulation (Bernon et al. 2003, Jacyno et al. 2009, Salazar et al. 2011, Kolen et al. 2018) and systems-of-systems design (Falkner et al. 2018, Kewley et al. 2008) and mechanism design and game theory communities (Park et al. 2000). These approaches are deployed in stable environments whose properties do not vary beyond well defined ranges and tend to focus more on the critical task of obtaining the desired property and less on analyzing its side-effects within the environment (Szabo and Teo 2016), such as the potential of firing on friendly forces due to reduced situation in our IMD example.

In addition to the challenges identified above, i.e., rudimentary or disrupted and potentially unsafe network and lack of complete system-wide information, any proposed solution needs to undergo a rigorous Verification, Validation, and Accreditation (VV&A) in order to be deployed (Sargent and Balci 2017). This requires significant experimental analysis of any proposed solution in existing, pre-defined military scenarios in newly created situations. In this paper, we propose a modeling and simulation testbed to analyze the parameters and individual node or entity behaviors that lead to the desired emergent property of timely message delivery. We define node specific behaviors and system wide communication protocols that have the potential to achieve the desired property. Our simulation and analysis testbed allows for the in-depth analysis of the side-effects of various node-specific communication strategies in a round-robin network wide communication protocol. The node-specific behavior represents the first step towards more complex behaviors where nodes collaborate to achieve the timely delivery of PLI messages and where network status and other context information are inferred. The main contributions of our work are:

- An engineered desired emergent property of timely message delivery
- A simulation experimentation architecture where the effectiveness of node-specific and environment specific parameters can be explored for a specific communication strategy
- A node-specific rule-based behavior that is easily extensible to include complex behaviors that are reactive both to other node information and to context changes

2 PROPOSED APPROACH

To ensure timely message delivery in contested and dynamic environments, there is a need for adaptive logic running on entities participating in a military operation. The individual behavior of the entities and their interaction in a volatile environment create a complex systems problem, and the desired timely message delivery becomes an engineered emergent property. There is also a critical need to ensure that the adaptive

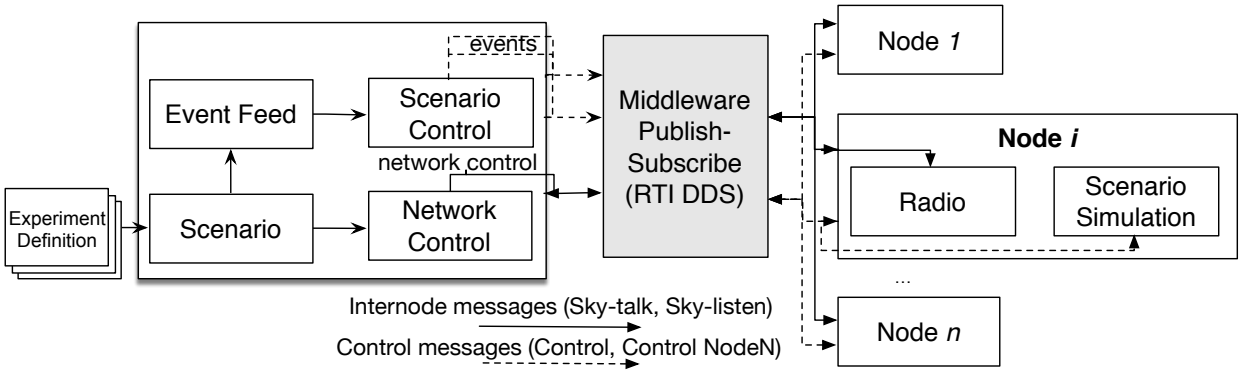


Figure 1: An Overview of the Simulation Testbed.

logic is tested and validated in a variety of scenarios before it is deployed. We propose to define the adaptive logic running on each entity using a rule-based expert system. Using our proposed simulation testbed, we explore the engineered emergent behavior and its side effects, both at a micro, individual node level, as well as at a macro, system-wide level.

The architecture of our simulation testbed is presented in Figure 1. Our testbed is designed as a plug-and-play architecture, allowing us to introduce both simple and complex communication strategies and to evaluate their performance under a variety of scenarios and input parameters. Individual nodes run node-specific communication strategy modules. Nodes are connected and communicate through the use of a publish/subscribe middleware, allowing for a fully distributed system where node logic is run on separate computers. Simulation experiments are created and subsequently driven through the use of an experiment control. A node encapsulates an individual platform/unit, performing both its communications and scenario decisions. The `Experiment Control` represents the central experiment control process and it determines and emulates network conditions such as scheduling and link bandwidth between nodes. The `Experiment Control` can also generate additional scenario events for nodes. The communication between nodes and experiment control happens over four channels, namely, `Control-nodeN` (node-specific control messages), `Control` (broadcast control messages, for network status information), `Sky-talk` (for messages to other Nodes, via the Network Control module) and `Sky-listen` (for messages from other Nodes, via the Network Control module). All experiment runs are driven by an experiment definition file, which defines the scenario used, the communication strategies employed by nodes, as well as the number of replications and various other input parameter ranges[1].

In a real-life deployment, the main functionality of the software system running on each node is to determine the priorities of Position Location Information (PLI) messages, which capture information about the position of a node as well as other status information. The PLI messages are the main avenue through which a node updates other nodes in the system about their status. A node will also send non-PLI messages, which could be, for example, critical information about adversary locations and status or less-critical text, voice, imagery or video information, as defined in the scenario. The node logic will change the priorities of PLI and non-PLI messages based on changes in context. *Context information* is comprised from network and operational information, representing the status of the network and the status of the battle and other nodes respectively. In our simulation testbed, the software architecture running on the node is enhanced with a radio simulator since access to a real operational radio is not possible within the simulation. The node architecture has three main modules, namely, `Context Controller`, `CommStrategy` and `Message`

[1] A version of the experiment definition file can be found online (CREST 2017).

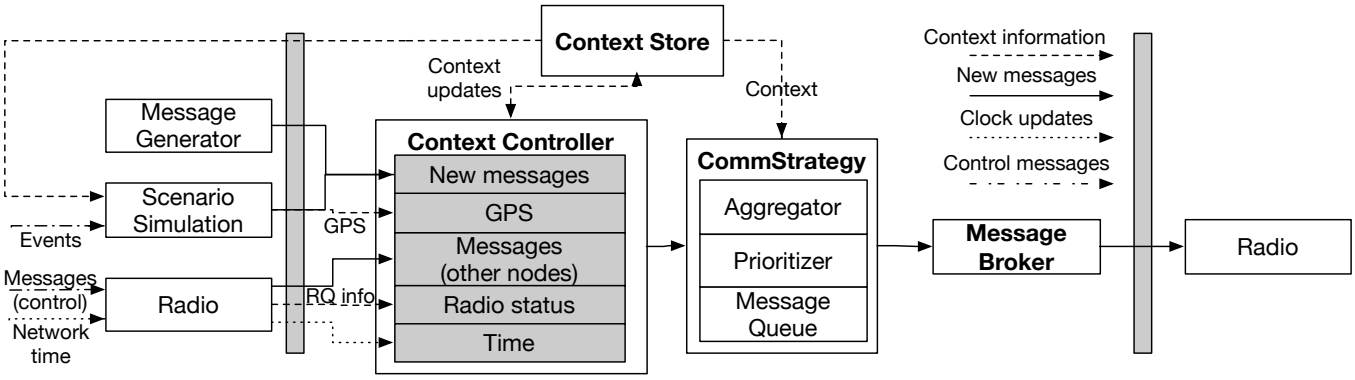


Figure 2: Detailed Node Architecture.

`Broker` as shown in Figure 2. These components interact with each other as well as with the *stimulation* components shown on the left side of the gray line as discussed in the following.

The `Context Controller` monitors all context inputs such as state changes from scenario (e.g. GPS sensor updates), network changes (network control messages), changes to other nodes updated via messages from other nodes, and radio internal queue state changes (if available) among others. Once a context change is detected, the `Context Controller` updates the local context in the `Context Store`, notifies the `CommStrategy` (Communication Strategy), and sends update messages to the `Scenario Simulation` module. New messages generated by the `Message Generator` or `Scenario Simulation` are forwarded to the `Communication Strategy` to be prioritized and sent.

The `Radio` receives messages from the Message Broker and adds them to an internal queue. When it is the Node's turn to broadcast, the Radio publishes messages on one of the channels of the communication middleware, called Sky-talk. Turns are coordinated based on messages from the Network Control module via the Control channel, and currently follow a round-robin scheduling. The Radio also subscribes to Sky-listen, and relays messages from other Nodes to the Context Controller.

The `Scenario Simulation` module receives input from one or more scenario definition files and updates the state of the Node at each time-step. The Scenario Simulation may also use current context information to make decisions that modify or expand upon the ones outlined by the scenario definition, such as deciding to interpret the positions as a plan, and periodically going ''off plan".

The `Message Generator` generates superficial messages to send to other Nodes, providing a wider variety of message types to help assess the effectiveness of the communications logic. Messages have a sourceId (the ID of the node that generated the message, or control), a creation time stamp, a priority, and a typeId that represents the message type, e.g PLI, InContact (with the enemy) among others.

The `Scenario Simulation` file specifies a time tick followed by NetworkControl, which sends a tick message to each Radio component on each Node. All Node submodules respect network time for time-critical tasks. The `Experiment Control` module controls the distribution of the scenario information (Scenario Control) and the network conditions (Network Control)as shown in Figure 1.

2.1 Baseline Communication Strategy

The pseudocode for the baseline communication strategy is shown in Algorithm 1, with prioritization decisions based on the state of battle (e.g., attack, moving forward, deep in contact). When a new Message is generated, it is passed (via the ChangeMonitor) to the Communication Strategy (CommStrategy). The CommStrategy delegates aggregation, prioritization, and ordering of output messages to its three sub-modules, namely, Aggregator, Prioritizer, and Message Queue. The CommStrategy forwards messages

its Aggregator, which may decide to hold a message (to drop, aggregate with other messages, or send later) or forward the message immediately. The DefaultAggregator forwards all messages to the Prioritizer immediately. The DynamicAggregator forwards all non-PLI messages immediately, but keeps PLIs until a time or distance threshold is met, at which point the latest PLI is forwarded. The Prioritizer assigns a priority to each message based on the current context. It then places the message in the MessageQueue.

The MessageQueue is a priority queue and we refer to it in the following as the application message queue (AMQ) to distinguish it from the radio queue (RQ). If two or more messages in the queue have the highest priority value, the newest is taken first; if they share the highest priority and creation time, one is taken arbitrarily. The MessageBroker is responsible for moving messages from the AMQ to RQ. In the baseline strategy, the MessageBroker indiscriminately moves messages from the AMQ to the RQ. The Radio monitors messages from NetworkControl to determine when it is its turn to send. It then sends any messages in its internal queue on a First-In-First-Out basis, up to a maximum of MPS messages per turn.

Algorithm 1 Overview of the Baseline Communication Strategy.

```
 1: function PROCESSMESSAGE(message, context)
 2:     if message.typeId is not PLI then
 3:         assign_priority(message, context)
 4:         priority_queue.add(message)
 5:     else if time_of_last_sent_PLI is null or context.current_time - time_of_last_sent_PLI >= 0 then
 6:         time_of_last_sent_PLI = context.current_time
 7:         position_of_last_sent_PLI = message.position
 8:         assign_priority(message, context)
 9:         priority_queue.add(message)
10:     end if
11: end function
12: function ASSIGN_PRIORITY(message, context)
13:     typeId = message.typeId
14:     if typeId is PLI then
15:         if context.operational_location is deep then
16:             message.priority = 70
17:         else if context.operational_task is attack then
18:             message.priority = 50
19:         else if context.operational_location is forward then
20:             message.priority = 40
21:         else
22:             message.priority = 30
23:         end if                                  ▷ non PLI messages prioritized here
24:     end if
25: end function
```

Improving the Baseline Communication Strategy Variations to the baseline communication strategy are driven by changes in context. Currently, we explore three main variations, namely, (i) strategies that explore additional information, (ii) strategies that explore time locality, and (iii) strategies that explore spatial locality, which are implemented as changes in line 5 and in the ASSIGN_PRIORITY function. The main additional information explored currently is information about the radio queue length. In this strategy, we propose to delay the messages sent to the radio until the radio queue can receive them. The *time locality strategy* defers sending messages from the AMQ to the RQ until a time threshold has elapsed. The time threshold refers to the number of time units the Aggregator waits between dispatching PLI messages to the AMQ. This wait time is based on the time when the last PLI was dispatched by the Aggregator and does not take into consideration the creation time of the PLI. The Aggregator always dispatches the most recent PLI that it has received. The *spatial locality strategy* exploits the situation where the position of the nodes deviates from the initial operational plan by having nodes send PLI messages at an increased rate.

3 EXPERIMENTAL ANALYSIS

The prototype adaptation logic is implemented in Prolog, with the rest of the node logic components and simulation testbed implemented in Java. A series of scripts and Matlab visualisation tools ensure that experiments in our testbed are fully automated. To show the benefits of using our experimental framework but also gain an in-depth understanding of the conditions under which the desired emergent property can be obtained, we aim to understand what influences the effectiveness of a communication strategy to generate the desired emergent property and what the side effects of the employed communication strategies are.

In these experiments, we employ a battle scenario where 20 nodes are moving through enemy terrain. The nodes go through an initial planning stage, after which they spread out through the terrain, until they reach their destination around 1,790 ticks later. For each node we compute the average delay between the Position Location Information (PLI) updates. This metric measures how frequently nodes send PLI messages. At a time moment t, the delay is calculated as the average of the following function $out\_of\_dateness(t) = (t - message\_creation\_time)$.

The delay is the number of time steps since the creation time of the last PLI that was sent, averaged over all time steps. The data point is the mean value for a node across all trials. For each of the experiments detailed below, each data point represents an average of ten runs. All experiments were run on a 3.1GHz Core i7, 16 GB RAM laptop computer running MacOS 10.12. In the following, we explore the behavior of the individual nodes and the systems under varying conditions of node throughput values (measured in the Messages Per Step, MPS, that is, the number of messages that a node sends during one round robin round), under the assumption that the choice of MPS is influenced by network conditions.

3.1 Exploring Radio Queue Information

We expand the baseline communication strategy with a delay logic implemented in the `MessageBroker`. The MessageBroker forwards messages from the application's message queue to the Radio's message queue, and it will make use of context information about the radio current queue length if it is available. In a *moderated* send strategy, the MessageBroker will moderate the number of messages it forwards to the Radio when the Radio's queue length reaches a threshold value, τ. When the radio queue reaches a length that is equal to τ, the MessageBroker will stop sending messages to the radio queue. Sending messages resumes once the radio queue length drops under τ. In the meantime, other messages would have accumulated in the application queue, and those with the highest priority are dispatched to the radio. This virtually delays and/or drops stale messages. When *unmoderated*, the MessageBroker has no information about the Radio's queue length, and may continue to fill its internal queue, leading to PLI delivery delays.

Table 1: Average PLI delays.

Throughput (msg/step)	**Average PLI delay (unmoderated)**	**Average PLI delay (moderated)**
10	619.8	687.5
15	485	51.83
25	237.6	47.26
35	**43.59**	**27**

In this experiment, we set the value of τ to be double the throughput value, $\tau = 2 * MPS$. We chose this value as a heuristic. Figure 3 shows a comparison between the moderated and unmoderated strategies when the throughput on each node increases between 10 and 40 messages per step. The moderated send strategy performs considerably better than the unmoderated send strategy in terms of delay from MPS 12

until MPS 40 where network conditions are such that almost all messages can be broadcast on the next send turn regardless of strategy. At MPS 12, the unmoderated strategy is able to send more PLIs, as it does not perform much prioritisation of messages because messages are quickly moved to the radio queue, which is un-prioritized. After MPS 21, the unmoderated strategy also sends more non-PLI messages and fewer PLIs than the moderated strategy, likely due to it not prioritizing messages.

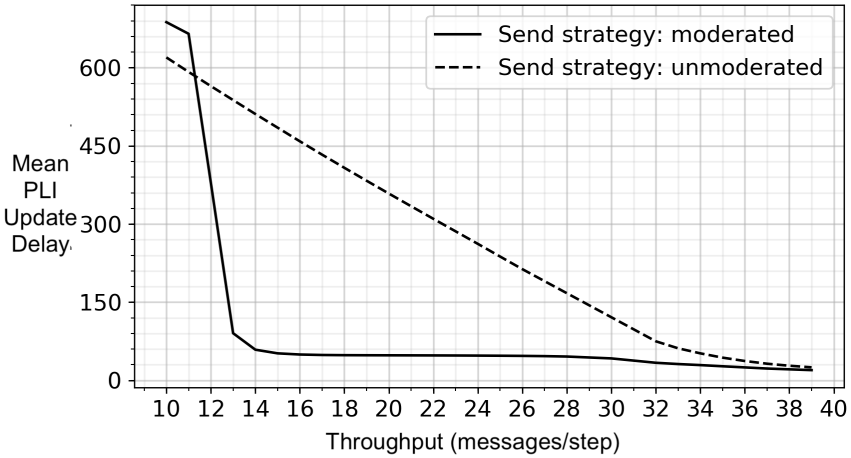


Figure 3: Effects of Throughput on Mean Delay Between PLI Updates.

A summary of the results is presented in Table 1, showing that the moderated send strategy outperforms the unmoderated strategy for all but one value of MPS, with close performance for the outlier.

Beyond system metrics analysis and visualization, our architecture allows for in-depth analysis of node-related behaviors. To illustrate this, we show in Figure 4 and 5 an overview of the status of a single node throughout the simulation for MPS=35. We analyze a single simulation run instead of presenting the averages because we aim to also understand when the PLI messages are sent, as network and operational contexts might mean that PLI messages are not sent at every turn.

At MPS=15, with an unmoderated send strategy, messages move quickly from the Application Message Queue (AMQ) to the Radio's internal queue (RQ), with little chance for prioritization. The unmoderated strategy results in sending a PLI update on each turn, but the PLIs that are sent are much older, since they come from an un-prioritized (first in, first out) queue.

At MPS 35, the two send strategies both begin to converge as nearly all messages can be sent, as shown in Figure 4 and Figure 5. For both strategies, increases in PLI generation correlate with spikes in queue length and delay, pushing the system into a state where PLI updates experience higher latency (due to the increase in radio queue length). The generation rate of PLIs is shown in green on the positive axis. The PLI generation rate is different for each Node, and the rate is not constant throughout the scenario.

3.2 Exploring Time Locality Strategies

In these experiments, we aim to determine the effect of throttling the sending of PLI messages based on a time threshold. The time threshold refers to the number of time units the Aggregator waits between dispatching PLI messages to the outgoing AMQ. This wait time is based on the time at which the last PLI was dispatched by the Aggregator and does not take into consideration the creation time of the PLI. The Aggregator always dispatches the most recent PLI received. To explore this, we run our simulations with different values of the time threshold, in a range between 0 and 16. The results are shown in Figure 6.

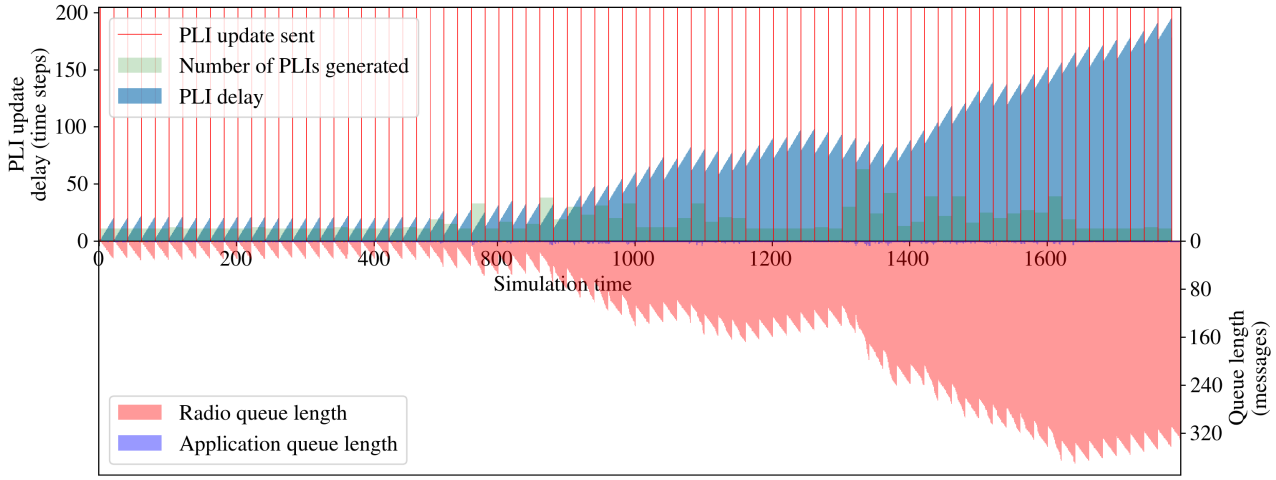


Figure 4: Node View of the Unmoderated Communication Strategy at MPS=35.

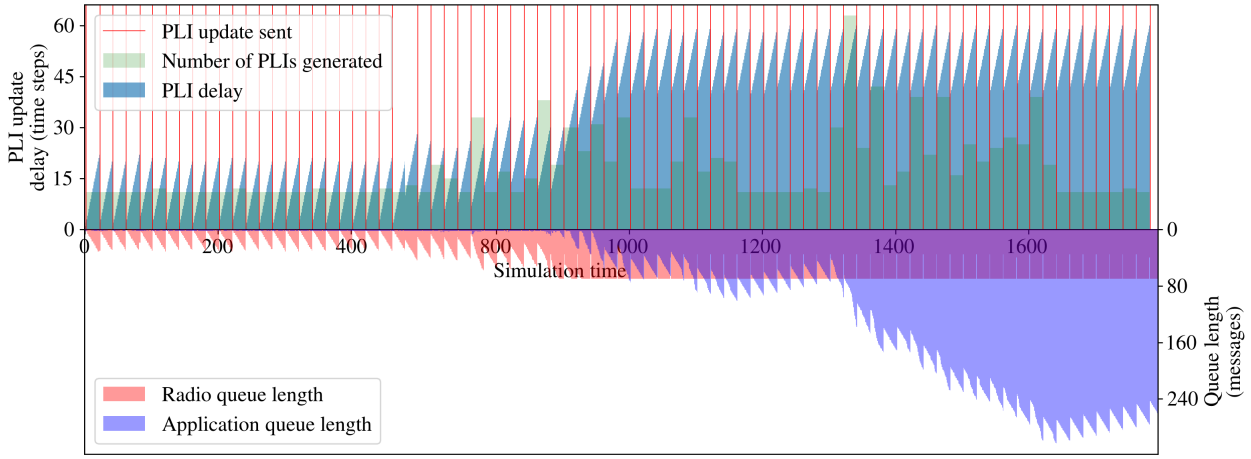


Figure 5: Node View of the Moderated Communication Strategy at MPS=35.

We can identify three regions in the plot across the range of throughput values in MPS, which represent changing network conditions, under the assumption here is that network conditions will influence the choice of MPS. If the network conditions are bad (MPS in the range of 14 to 19), there are many more messages generated than can be sent. In this region of the graph, higher aggregation time thresholds have a slightly higher mean update delay. In moderately good network conditions (MPS in the range of 20 to 24), higher time threshold strategies reach their optimal values more quickly than lower ones. This is because with fewer PLI messages being sent at higher thresholds, the internal queue of the radio reaches a point where it does not fill up immediately at lower MPS values. If the radio queue is not full, the time between passing a message to the radio and the radio sending that message is lower. In good network conditions (MPS above 25), we can see that while higher thresholds reach an optimal value more quickly, their optimal delay is still higher than that of lower time thresholds. This is because, once that optimal value is reached, network contention is less of an issue. When network contention is not an issue, the optimal strategy is to send more PLI messages, as this increases the probability that a newer PLI will be included in the radio's next

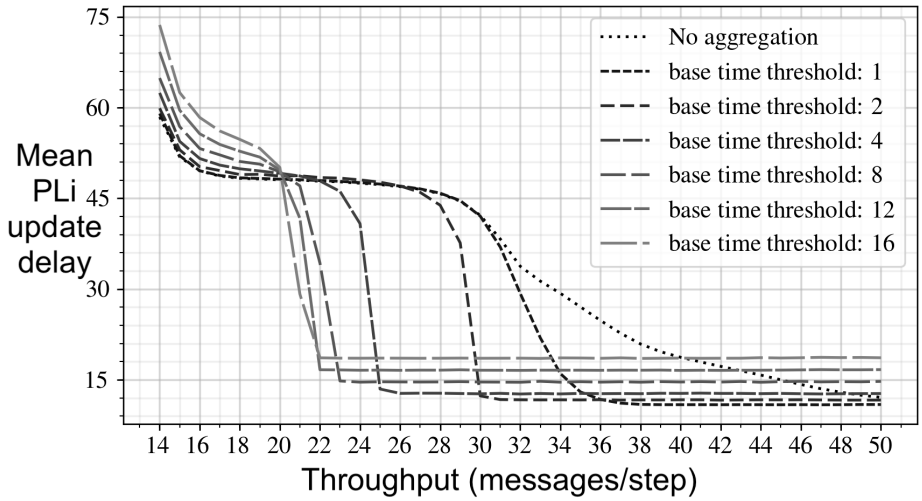


Figure 6: Effects of Throughput on Time Threshold Strategies.

send turn. Thus, as network conditions improve, less restrictive strategies become favorable. The average PLI delays values are shown in Table 2.

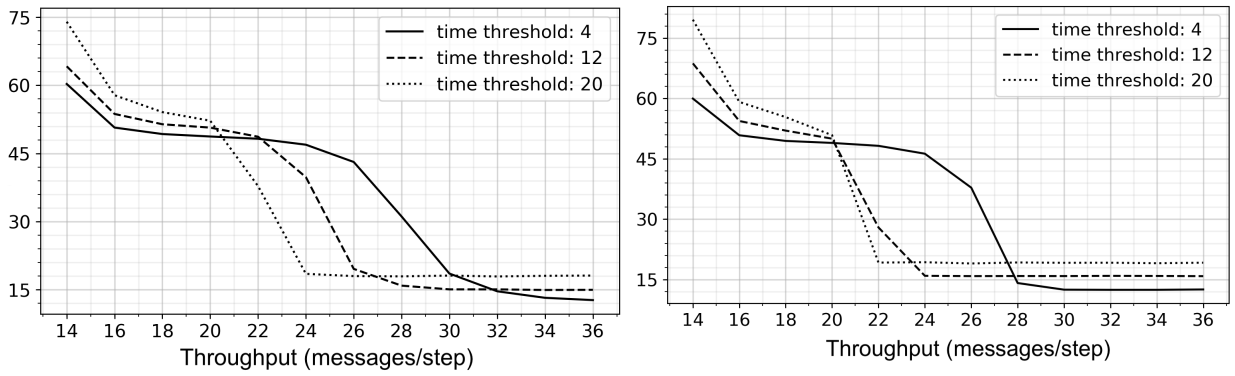
3.3 Exploring Spatial Locality Strategies

In these experiments, we aim to determine how the difference between the real geographical position of the node and that prescribed in the mission plan influences the performance of the communication strategy. In this scenario, we interpret the input positional data as a plan. Each node deviates from it randomly for a period of time, changing the node's context to off-plan. This context change is picked up by the `Context Controller`, causing the Aggregator to increase the rate at which it sends PLIs by multiplying the time threshold by a multiplier, Δ, with $\Delta = 0.2$ or $\Delta = 0.5$. We explore this scenario in combination with a time locality threshold, and employ a `moderated` strategy, over various throughput values. The results for

Table 2: Average PLI delays for time locality.

Throughput	**Average PLI delay for Time Threshold**						
(msg/step)	**0**	**1**	**2**	**4**	**8**	**12**	**16**
15	51.86	52.05	53.03	54.43	56.87	59.58	62.5
25	47.28	47.37	47.69	13.48	14.67	16.61	**18.59**
35	**27.06**	**12.99**	**11.68**	**12.75**	14.69	16.59	18.61
45	14.99	10.88	11.69	12.75	14.7	16.61	18.6

various base time thresholds (the time threshold multiplied with Δ) are summarized in Figure 7 (a) and (b) for $\Delta = 0.2$ and $\Delta = 0.5$ respectively. As throughput increases, each time threshold reaches a stable, optimal mean PLI update delay. When this is reached, the Radio is no longer maximally utilized. Higher time thresholds forward fewer PLI messages, and so reach their optimal values at lower network throughput values. However, when network conditions are less restrictive at higher MPS, the increased delay between PLIs inherent in the use of higher time thresholds is apparent, and lower thresholds begin to perform better. Each time threshold series for $\Delta = 0.5$ plateaus earlier, but at a higher mean PLI delay value compared with the corresponding time threshold series for $\Delta = 0.2$. This is similar to how higher time thresholds compare with lower ones for a single multiplier as shown in Table 3.



(a) $\Delta = 0.2$ (b) $\Delta = 0.5$

Figure 7: Effects of Spatial Locality Multipliers on Various Time Threshold Aggregations Δ for Mean PLI Update Delay.

Table 3: Average PLI delays for time and spatial locality.

Multiplier	**Throughput (msg/step)**	**Average PLI delay for Time Threshold**		
		4	**12**	**20**
1	14	62.2	69.47	94.62
	22	47.52	16.98	20.43
	30	12.84	16.83	20.51
0.2	14	60.31	64.2	74.02
	22	48.25	48.69	37.81
	30	18.56	15.08	18.11
0.5	14	59.97	68.72	79.58
	22	48.23	27.93	19.22
	30	12.49	15.87	19.17

Discussion The communications strategy attempts to optimize the number of messages in each of the two queues in order to minimize delay between PLI updates and maximize the number of messages that are sent. Increases in radio queue size result in higher PLI delay, since a PLI entering the radio's queue must wait for all other currently enqueued message to be sent. The size of the application queue does not necessarily correlate with PLI delay, since the application queue prioritizes newer messages, moving them forward in the queue. However, if too many messages are left in the application queue, message types with higher prioritization for a given context state may build up, preventing lower priority message types from being sent at all. Leaving messages in the application queue is also detrimental to performance in the case where the radio's queue becomes empty when it would have been possible to send messages.

At a throughput of 15 MPS, the moderated send strategy offers an 89.3% improvement of PLI delay over an unmoderated strategy, and a 19.8% improvement of the number of non-PLI messages sent. With less restrictive network conditions at a throughput of 25 MPS, the moderated send strategy offers an 80.1% improvement in PLI delay as well as a 10.4% improvement in the number of non-PLI messages sent.

The communication strategy attempts to optimize the number of messages in each of the two queues (AMQ and RQ) in order to minimize delay between PLI updates while maximizing the number of messages that are sent. Increases in radio queue size result in higher PLI delay, as a PLI in RQ must wait for all other

currently enqueued message to be sent. The size of AMQ does not necessarily correlate with PLI delay, since the application queue prioritizes newer messages, moving them forward in the queue. However, if too many messages are left in the application queue, message types with higher prioritization for a given context state may build up, preventing lower priority message types from being sent at all.

4 CONCLUSION

Using our simulation testbed we were able to explore the benefits and side effects of a simple communication strategy logic running on individual nodes as part of a battle scenario. Our analysis has shown the engineering of a desired emergent behavior and has identified the communication strategies that were most likely to achieve it, namely, a communication strategy based on knowing the status of the radio queue (for an 89.3% improvement over baseline) and one on delaying and dropping stale messages. Our fully automated testbed has also allowed us to explore the side effects of the chosen strategies on the internal node mechanics.

This work represents a promising first step towards the in-depth analysis of individual node behaviors that achieve the desired emergent property of resilient communication in dynamic and contested environments. The testbed allows important military *what-if* questions to be examined, in particular with respect to access to radio queues. While the queue modeling here is rudimentary, it gives preliminary evidence as to the benefits of an open architecture specification of new radio capabilities in defence acquisition projects.

Despite its promise, several limitations exist. The simulation runs on a publish-subscribe communication middleware, with the network completely abstracted and removing all network traffic informaton from the Context Controller. This is a direct consequence of one of the requirements of our work, namely that of rapid prototyping of the communication logic running on each node. In the future, once several communication strategies are thoroughly analyzed, we aim to connect the node logic to a network emulator to better experiment with network context changes. Next, the current logic implementation does not consider more complex operational contexts (e.g., a soldier being shot) that would require critical changes to priorities nor any information about the status of other nodes in the system, and their observed operational and network contexts.

REFERENCES

Bedau, M. 1997. "Weak Emergence". *Nous* 31(11):375–399.

Bernon, C., M.-P. Gleizes, S. Peyruqueou, and G. Picard. 2003. "Adelfe: A Methodology for Adaptive Multi-Agent Systems Engineering". In *Engineering Societies in the Agents World III*, 156–169. Springer.

Campbell, Angus LTGEN 2018. "Australia's Joint Force Land Capability – Address by Chief of Army, Lieutenant General Angus Campbell, to Australian Defence Magazine Congress, Canberra,14 Feb 18".

CREST 2017. "https://github.com/CRESTPublic/CRESTPublic". Last accessed Jun. 2018.

Falkner, K., C. Szabo, V. Chiprianov, G. Puddy, M. Rieckmann, D. Fraser, and C. Aston. 2018. "Model-driven Performance Prediction of Systems of Systems". *Software & Systems Modeling* 17(2):415–441.

Jacyno, M., S. Bullock, M. Luck, and T. R. Payne. 2009. "Emergent Service Provisioning and Demand Estimation through Self-organizing Agent Communities". In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 481–488: ACM.

Kewley, R., J. Cook, N. Goerger, D. Henderson, and E. Teague. 2008. "Federated Simulations for Systems of Systems Integration". In *Proceedings of the 40th Winter Simulation Conference*, edited by S. M. et al., 1121–1129. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kolen, S., S. Dähling, T. Isermann, and A. Monti. 2018. "Enabling the Analysis of Emergent Behavior in Future Electrical Distribution Systems Using Agent-Based Modeling and Simulation". *Complexity*.

Mittal, S. 2013. "Emergence in stigmergic and complex adaptive systems: A formal discrete event systems perspective". *Cognitive Systems Research* 21:22–39.

Mogul, J. C. 2006. "Emergent (mis)behavior vs. Complex Software Systems". In *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems*, 293–304. New York, USA.

Park, S., E. H. Durfee, and W. P. Birmingham. 2000. "Emergent Properties of a Market-based Digital Library with Strategic Agents". *Autonomous Agents and Multi-Agent Systems* 3(1):33–51.

Salazar, N., J. A. Rodriguez-Aguilar, J. L. Arcos, A. Peleteiro, and J. C. Burguillo-Rial. 2011. "Emerging Cooperation on Complex Networks". In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 669–676: ACM.

Sargent, R. G., and O. Balci. 2017. "History of Verification and Validation of Simulation models". In *Proceedings of the 2017 Winer Simulation Conference*, edited by W. K. V. C. et al., 292–307. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Suri, N., G. Benincasa, R. Lenzi, M. Tortonesi, C. Stefanelli, and L. Sadler. 2015, October. "Exploring Value of Information-based Approaches to Support Effective Communications in Tactical Networks". *IEEE Communications Magazine* 53(10):39–45.

Szabo, C., and Y. M. Teo. 2016. "Formalization of Weak Emergence in Multiagent Systems". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 26(1):6.

AUTHOR BIOGRAPHIES

CLAUDIA SZABO is a Senior Lecturer at the School of Computer Science and the Head of the Complex Systems Research Group at The University of Adelaide. Her research focuses on modeling and analysis of complex systems and on practical applications of complexity theory to various application domains. Her email address is claudia.szabo@adelaide.edu.au.

DUSTIN CRAGGS is a Software Developer in the Complex Systems Research Group at The University of Adelaide. His email address is dustin.craggs@adelaide.edu.au.

WAYNE JOHNSON is a Senior Research Specialist at the Australian Defence Science and Technology Group. His research interest focuses on Artificial Intelligence techniques applied to data fusion problems in various Department of Defence application domains in general, and Army Command and Control systems in particular. His email address is wayne.johnson@dst.defence.gov.au.

GREGORY JUDD is a Senior Research Specialist at the Australian Defence Science and Technology Group. His email address is greg.judd@dst.defence.gov.au.

KEITH FRENCH is a Senior Research Specialist at the Australian Defence Science and Technology Group. email address is keith.french@dst.defence.gov.au.