Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E. P. Chew, T. Roeder, and P. Lendermann, eds.

ROBUSTNESS OF MIDDLEWARE COMMUNICATION IN CONTESTED AND DYNAMIC ENVIRONMENTS

Claudia Szabo
Dustin Craggs
Dumitru Alin Balasoiu

The University of Adelaide
North Terrace
Adelaide, SA 5000, AUSTRALIA

Vanja Radenovic

Consilium Technology
147 Pirie Street
Adelaide, SA 5000, AUSTRALIA

Benjamin Campbell

Defence Science and Technology Group
Third Avenue
Edinburgh, SA 5111, AUSTRALIA

ABSTRACT

Contested and dynamic environments with poor and unreliable network conditions are often encountered by military operations or crisis management systems. While it is a common occurrence for networks to behave poorly, or for system nodes to malfunction, most existing decision making algorithms have only been tested in perfect conditions. In this paper, we present an analysis of the robustness of reinforcement learning and evolutionary approaches employed in a communication middleware that operates in a contested environment. The SMARTNet middleware prioritizes and controls the messages sent by each node, with the aim of preserving network bandwidth. We evaluate the robustness of a reinforcement learning and an evolutionary computation implementations as SMARTNet executes in changing conditions, with nodes dropping out, and the network becoming congested both generally and for specific message types.

1 INTRODUCTION

Rescue and military operations often operate in contested and dynamic environments, with unavailable or jammed networks (Campbell, Angus LTGEN 2018). In these systems, ensuring timely messages arrive without massive network disruptions is critical for the successful outcome of the operations and to ensure mission effectiveness. Most contested environments have poor networks or have networks where the network performance might quickly deteriorate over time or as the network nodes move further apart geographically or are partitioned from the network, either intentionally or through forces of nature.

For example, operations in foreign and contested terrain use radio-based ad-hoc networks which cannot guarantee high bandwidth reliable communication. In most battle scenarios soldiers may not have a complete view of the position or status of friendly forces and thus critical information dissemination decisions that will affect system-wide properties have to be made with incomplete and unreliable information. Systems that operate in rescue or humanitarian operations do not have the luxury of stable network and computational infrastructures, must compete over limited power resources and can can be affected by natural or man-made disasters. It is critical that the entities in these systems are responsive to frequent network capability changes, and that there is an understanding of how robust the system is to various environment changes.

One of the most frequently sent type of messages in these type of mobile systems is a Position Location Information (PLI) message, which details the position information of the sending entity. Other message types include status information, text, video, as well as other non-critical messages. While all messages are assumed to be important when they are sent at each individual node, there is a need for a prioritization mechanism that ensures timely delivery, responsiveness and global graceful degradation mechanisms for the entire system. To achieve this, node-specific rules can be defined based on the partial knowledge a node might have about its environment. Nodes then reason about context and prioritize information sharing activities as it suits the mission plan and the current context. This complex systems design approach assumes that individual nodes will make decisions either in cooperation with other nodes or in isolation caused by various network partitions or specific attacks.

SMARTNet (Chan et al. 2018; Szabo et al. 2020) is a proposed experimental middleware that prioritizes, controls, and transforms any communication sent by agents deployed on nodes across any networked applications. Running on every network-connected soldier, vehicle and headquarters, SMARTNet controls the node's access to the network and uses information available from its battle management systems, networks, and sensors to build up a representation of the current state of its platform, mission, environment and network. SMARTNet uses this contextual knowledge to dynamically decide what priority each message should have, whether the message needs to be transformed (reduced, compressed or filtered) to fit current network capacity, and when the message should be sent.

SMARTNet operates in contested and dynamic environments, and as such each agent needs to send numerous messages according to the operational and mission plan. Since the network is often unstable or limited in capacity, the sending of messages potentially results in degrading network performance. In our past work (Szabo et al. 2020), we have explored the use of centralised evolutionary algorithms to solve this problem, with promising results, while still facing challenges. In particular, our approach had a significant runtime when deployed live and could not adapt well to a dynamic environment when deployed statically. To address this, we introduce a simple reinforcement learning approach that considers local context information when sending messages. To address gaps in existing work, in this paper we also evaluate the robustness of reinforcement learning and evolutionary computation approaches, using varied scenarios. The contributions of this paper are:

- A reinforcement learning algorithm to reduce network congestion in the SMARTNet middleware while achieving timely delivery of messages
- An extensive experimental analysis of the robustness of the reinforcement learning and evolutionary computation over a baseline in a variety of congested and disrupted scenarios

2 BACKGROUND AND RELATED WORK

A mobile ad-hoc network (MANET) is a decentralised wireless network composed from mobile devices that operate without support from existing static infrastructure, e.g., routers or servers (Ali and Kulkarni 2015). Similar to a peer-to-peer network, it is the MANET nodes that provide the routing and all other services to every node in the network. The inherent flexibility, scalability and fault tolerance given by their design ensures that MANETs are good candidates for contested, dynamic, and resource constrained environments such as military operations (Rajabhushanam and Kathirvel 2011; Rath and Pattanayak 2014) or crisis management (Reina et al. 2015).

There has been significant research interest towards machine learning and artificial intelligence technologies, and their application in various network related problems (Talawar and Ashoka 2020; Li et al. 2018) such as routing protocols (Wu et al. 2008; Gudakahriz et al. 2011; Li et al. 2018; Ghouti et al. 2013) and security (Talawar and Ashoka 2020). An iterated local search that finds the minimum spanning tree to connect all nodes is proposed by Wu et al. (2008). Wolf and Merz (2009) focus on achieving minimum power broadcast. However, centralized, global knowledge is required by both approaches. To alleviate this, nature-inspired algorithms have been used by Günes et al. (2003), who use behavior similar to that of ant colonies to design a routing algorithm that requires only local knowledge. Other nature inspired approaches include ant, bee or swarm abstractions (Gudakahriz et al. 2011; Saleem et al. 2012) respectively.

Specifically focusing on MANET routing protocols, Li et al. (2018) use reinforcement learning in a vehicular MANET to improve the message delivery ratio. The geographical area is divided into smaller grids and the approach iteratively finds the next optimal grid toward the destination. Extreme learning machines and user mobility models are used by Ghouti et al. (2013) in the design and implementation of an enriched MANET, that benefits from the prediction of its mobility. The approach is limited by the prediction accuracy when the distance between neighbouring nodes is computed. Thangaramya et al. (2019) uses a convolutional neural network in a wireless sensor network to achieve energy aware message routing. Lastly, a shortest path approach is proposed by Ghasemnezhad and Ghaffari (2018), focusing on node cooperation, with secure routing and reduced energy consumption as the main design requirements.

Several other approaches exploit optimization and evolutionary computation algoirthms, such as the one proposed by Liu and Huang (2009), who developed a multi-objective Evolutionary Algorithmic (EA) approach to solving the Multicast Routing Problem in MANETs. The EA genome is the paths, and the approach employs the Prufer path encoding scheme. The objectives include delay, delay jitter and packet loss minimization and maximization of bandwidth utilisation. Abdou et al. (2011) applied evolutionary algorithms for solving adaptive flooding in MANETs. In flooding, information regarding route discovery and maintenance is spread around the network. Nodes are interested in spreading as much information about themselves in the network as possible, however this leads to bandwidth saturation. To address this, an adaptive model is proposed for each flood message broadcast. The model considers different network densities encoded in parameter configurations that are generated using an EA. Lastly, two multi-objective EAs are compared by Yetgin et al. (2012), with a focus on determining routing paths. Route energy use and end-to-end delay are considered in the analysis.

2.1 SMARTNet Overview

To address challenges of timely message sending in contested environments, the Semantically Managed Autonomous and Resilient Network (SMARTNet) (Chan et al. 2018) middleware prioritises, transforms, and controls messages within the network. Messages passed through each SMARTNet node benefit from enhanced situational and mission context awareness. The middleware aims to achieve timely delivery while maintaining critical network performance, and employing several approaches to message transformation and routing (Quin et al. 2019). SMARTNet integrates various information sources and can be deployed on different platforms. Figure 1 shows the main SMARTNet components. Several modules are of interest, including *Mission Context Inference*, which infers the current context from sensor information, *Network Context Inference*, which infers the global network state from current local information, and *Transform*, which transforms the messages into packets. Helper modules include *Logging*, *Sensor Adapters* allowing SMARTNet to connect to a variety of sensors, and *Tactical Network Adapters* allowing SMARTNet to run on top of varying kinds of networks. SMARTNet uses plugins to connect to four types of network, namely, a single-threaded in-memory network, a proprietary network simulator called FogNet, the EMANE network emulator (Ivanic et al. 2009; United States Naval Research Laboratory 1996), and real life radios.

The message types that can be sent include Position Location Information (PLI), enemy detection, text, video and operational context messages. The PLI messages contain node position and node status information, and they are the mechanism through which a node updates other nodes of its status. Other message types can also be sent, including critical information about adversary locations or detection (RedDetection) and status or less-critical text, voice, imagery or video information, as defined in the scenario. The SMARTNet communication strategy module determines the priorities of all message types. Context information is comprised from network and operational information.

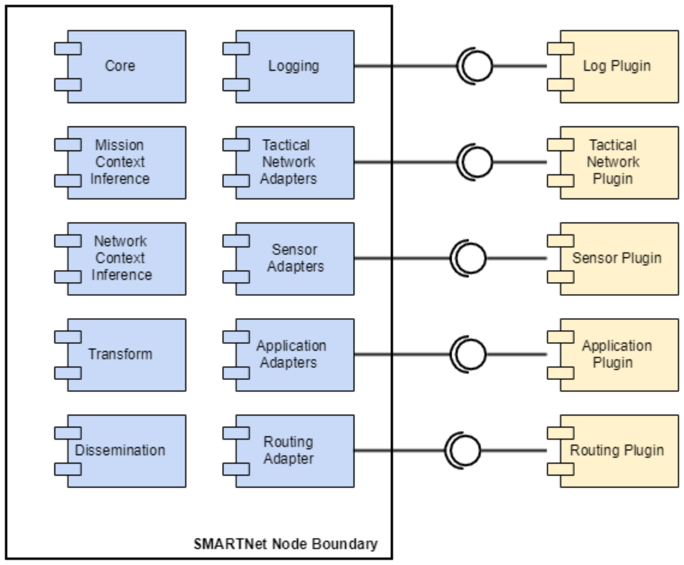

Figure 1: Main components of the SMARTNet architecture.

3 Artificial Intelligence Modules in SMARTNet

A reinforcement learning and an evolutionary optimization system were integrated within the SMARTNet middleware to improve the behavior of the middleware across a variety of network load scenarios. Reinforcement learning was used to ensure that the middleware reacts promptly to sudden environmental changes. In addition, reinforcement learning modules can be employed autonomously.

3.1 Reinforcement Learning Dissemination System (RLDS)

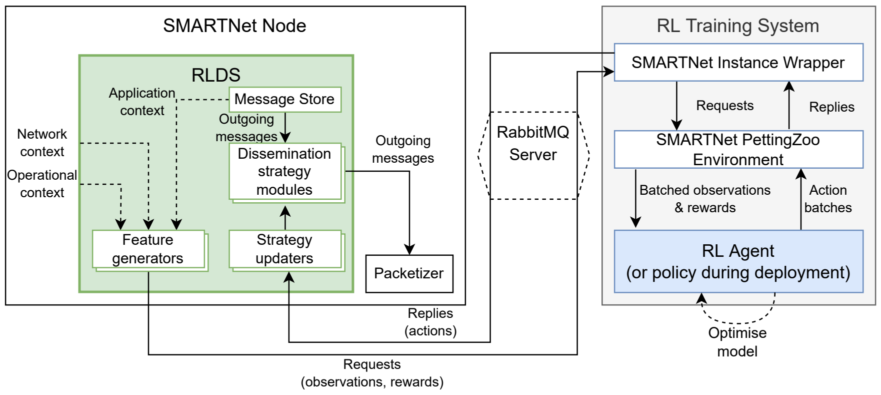

Figure 2: Architecture overview for the RLDS.

Our reinforcement learning system consists of three main components: the Reinforcement Learning Dissemination System SMARTNet (RLDS), the Reinforcement Learning Training System (RL Training system), and the Reinforcement Learning Deployment System (RL Deployment System) (Figure 2).

The main functionality of the RLDS module is to order outgoing messages when they are provided to the packetiser. The RL Training and Deployment systems are implemented in Python and communicate with the RLDS via the RabbitMQ message broker. The RLDS module is responsible for providing a Message Ordering implementation within SMARTNet which then determines the order in which outgoing messages are provided for packetisation. The RL Training system and the RL deployment system require contextual information about the specific messages being prioritised. The RLDS transforms this information into a normalised representation suitable for the RL Training system.

Each Node in SMARTNet has its own Reinforcement Learning Dissemination System (RLDS) module, which is responsible for executing dissemination system decisions within SMARTNet. The RLDS obtains dissemination rulesets from an external process (the RL agent) via RabbitMQ. Requests to the external RL agent are generated by a number of Feature Generators, which take raw context information and transform it into a representation suitable for the agent to use during training or deployment. For each request/observation, the RL agent returns, via RabbitMQ, a set of ruleset parameter values that is then applied to the relevant dissemination strategy modules by the Strategy Updaters. The dissemination strategies prioritise and transform outgoing messages before providing them to the packetizer to be sent on the network.

The RL Training System is responsible for servicing requests for dissemination strategy ruleset parameter updates from the RLDS, while simultaneously training a model based on feedback from the RLDS in the form of rewards. The requests consist of an observation from a particular SMARTNet node summarising the context of that node, as well as a reward. The observation summarises the network and operational context of the node. The responses consist of an action, which is an update to the SMARTNet dissemination system parameters. The RLDS and RL Training System communicate using the RabbitMQ message broker. Requests and responses are encoded as JSON objects. All nodes periodically request parameter updates from the RL agent without immediately waiting for a response, giving the RL agent time to service the requests asynchronously. A fixed amount of simulation time later, each node updates its strategy. This introduces a fixed, deterministic amount of latency into the decision making of the Nodes, but increases the speed at which the simulation runs.

The **SMARTNet Instance Wrapper** provides a Python interface for managing the configuration, output, and lifecycle of a SMARTNet simulation process. It also sets up a RabbitMQ connection with SMARTNet and provides an interface to communicate with the SMARTNet nodes during the run.

The **SMARTNet PettingZoo Environment** provides access to SMARTNet via an API suitable for reinforcement learning (the PettingZoo API (Terry, Black, Grammel, Jayakumar, Hari, Sulivan, Santos, Perez, Horsch, Dieffendahl, Williams, Lokesh, Sullivan, and Ravi 2020)). It manages an underlying SMARTNet instance using the SMARTNet Instance Wrapper, allowing an RL agent to obtain observations and rewards from Nodes, and provide corresponding actions in return. The environment automatically cycles to a new SMARTNet instance upon reset. During training, the environment is wrapped to provide batched observations and rewards and accept batched actions, where each element of the batch corresponds to one of the SMARTNet nodes in the simulation.

3.1.1 RL Agent

The RL Agent is responsible for predicting the optimal action given a provided observation, as well as for training this model based on incremental reward values. A Soft Actor-Critic (SAC) agent trained using self-play, where the same policy is applied at each Node in the system, is used in this experimentation. The SAC agent uses a continuous, stochastic policy and is trained using entropy-regularisation to facilitate exploration. The SAC implementation comes from the TensorFlow Agents library. The result of this training is a *policy*, which is a function that maps observations to actions. When training concludes, the RL Agent saves this policy for later use (such as for re-evaluation and comparison to other strategies).

During training, trajectories are sampled from 24 parallel SMARTNet instances, each running a 13-Node scenario. Trajectories from each node are batched together, resulting in a training batch size of $13 * 24 = 312$. The learner and SMARTNet environments run in separate processes. Running the SMARTNet environments in parallel drastically increases the speed of data collection. Batched samples from SMARTNet are placed into the replay buffer, from which the learner samples trajectories for training uniformly at random.

The RL Agent is trained using observations and rewards generated by the Feature Generators, and must provide actions to the Strategy Updaters that can be used to update the dissemination ruleset parameters. Observations consist of local node context from several sources. Rewards are taken directly from the MLM score, which is SMARTNet's global situational awareness metric. Actions consist of a priority value for each message category, as well as a time and distance threshold to apply as PLI Message rate-limiters.

The observations generated by a node comprise of local information available to that node that summarise its context in the mission (operational context), the context of its networks (network context), and the context of the dissemination system itself (application context). Operational context includes the average geographical distance to other Nodes, the Node's own level in the operational hierarchy, the number of other nodes in the network, and whether the Node is acting as a gateway. Network context includes information that can be used to estimate the outgoing capacity of the Node's networks. Application context includes a summary of the composition of the outgoing Message Store, including information about its size, the message categories it contains, the ages of these categories, and the fraction of routed outgoing traffic.

The mid-level metric (MLM) captures the overall performance of SMARTNet in disseminating messages to their required destination based on message type, timeliness, identity of the receiving node, and the current state of a simulated operational environment. The message generation and scoring is derived from unclassified subject matter expert (SME) advice, with a higher score being better and the best score being 0, corresponding to perfect and instant message dissemination.

The MLM has four main scoring components, namely, Blue Spots, Enemy Detection, Text Messages, and Tactical Graphics. Each of these score components is calculated as an accumulation of penalties as shown in Table 1, which are computed based on several modifiers that depend on each node's specific type. The penalty for a message is calculated as the product of the Score Measurement, Time Modifier, and Receiver modifier for the specific message type. In order to balance their contribution to the total score, a weighted sum of penalty components is computed for each node:

$$P = 2.0P_b + 4.0P_e + 0.025P_{tx} + 0.025P_{tg}$$

where P_b refers to the information type associated with friendly nodes (blue), P_e is associated with the enemy (red) detection, P_{tx} is associated with text messages and P_{tg} is associated with tactical graphics messages. Each information type has an associated time and receiver modifier depending on the node type. For example, for an enemy detection message, if the node is part of a section or platoon, the enemy detection message is less important the further the enemy is from the receiving node, but more important if the node is part of a battlegroup. The system provides updates of the score once per second as the scenario is progressing. The final penalty sums are averaged across all nodes in a specific scenario.

3.1.2 Agent & Learner Hyperparameter Summary

Replay buffer capacity: The replay buffer is the size of the trajectory buffer that is used to store the observation, reward, and action history from the SMARTNet instances. The learner samples from this buffer uniformly at random in order to reduce gradient variance and stabilise training. The buffer capacity is in number of batched samples, each of which is 312 elements long, meaning that the maximum capacity of the buffer in terms of individual samples is the buffer capacity multiplied by 312.
Actor & critic fully connected network layer parameters The actor and critic networks use a simple, fully-connected neural network architecture to approximate their targets. These networks are parameterised by the number of layers, and the number of nodes in each layer.
Training steps per iteration On each iteration, the RL system collects one batched set of observations, actions, and rewards from the 24 parallel SMARTNet instances. Because such a large amount of data is collected on each step, it is an advantage to run multiple iterations of the learner between each collection, to ensure that the learner is likely to utilise every collected sample to increase its speed of convergence.
Reward scale factor The rewards produced by SMARTNet are large, negative MLM penalties which must be scaled to prevent overadjustments of the agent early on during training.

Table 1: Modifiers used in MLM calculation.

Information Type	Score Measurement	Time modifier	Receiver Modifier
Movement (Blue)	Distance error: 0.1 point per meter	NA	Penalty reduced by 20% for each 100 meters (ground truth distance to node); Penalty reduced by 20% for each node distance in operational hierarchy
Enemy Detection	Time since generation: 10 points per second	Penalty decays by 20% every 1 minute	Penalty reduced by factor PR for each 100 meters (ground truth distance to node)/ PR starts at 20% for section level nodes, and is reduced by 4 percentage points for hierarchical level above Section level; (PR = 0.20 for sections, 0.16 for platoons, 0.12 for company, 0.08 for battlegroup)
Text Messages	Time since generation: 5 points per second	Decays by 0.5 points/minute	NA
Tactical Graphics	Time since generation: 5 points per second	Decays by 0.1 points/minute, reduced to 0 at 30 minutes.	NA

3.1.3 Actions & Rewards

The action space for the RL System consists of the five priorities for each message category, and the time and distance threshold. The agents are provided with rewards directly from the MLM score. The single-step reward for a transition is the change in score between the initial and final observations in the transition. The rewards are provided to the Agent via the Remote Updater using the MLM Score Feature Generator.

3.2 Evolutionary Computation

The Evolutionary Algorithm (EA) approach applies a method analogous to biological evolution to the SMARTNet dissemination ruleset. It starts by generating an initial population of rulesets with random parameters (individuals). The chromosome is composed of static message priority parameters for each message category (Position Location Information, Tactical Graphic, Text, SOS, Enemy position) and position transform parameters and has the following structure: $PLI\,priority\,|\,Tactical\,Graphic\,priority\,|\,Text\,priority\,|\,SOS priority\,|\,Red\,priority\,|\,Update\,Rate\,(seconds)\,|\,Update\,Distance\,(meters)$ Priorities are integers in the range [0, 4], and Update Rate and Update Distance both are integers in the range [0, 1000]. The next generation of individuals is then created by applying the *selection*, *crossover* and *mutation* operations to the initial population. The approach evaluates each individual by running a number of SMARTNet runs and obtaining the average MLM score. The fittest (highest MLM) 50% of the individuals from the population are retained. In the *crossover*, new individuals are generated to replace the discarded ones by either applying the crossover operation, or copying an individual from the previous generation. *Mutation* randomly changes some of the new individuals. This process is repeated on each successive generation of individuals until a termination condition is met, either a generation limit or a maximum number of SMARTNet runs.

For these experiments, we employ a static dissemination ruleset which assigns predefined priorities to each message type (PLI, Red, Text, Tactical Graphics, SOS). Each priority is restricted to be an integer in the range [0, 4]. In this context, **mutation** replaces a single element of the individual (priority or threshold) with a uniformly randomly sampled value. **Crossover** splices the priorities of the two individuals at a random index; 50% of the time swap either the time or distance threshold of the two individuals.

4 EXPERIMENTAL ANALYSIS

We perform robustness experiments on both the RL and EA implementations within the SMARTNet middleware, as described in the below. Our robustness scenarios focus on events that are typical of contested and dynamic environments, where the network performance might deteriorate or nodes might become unavailable. As such, our scenarios test gateway and node disconnection, increased rate of messages of different types, and decreased network performance captured through increased send rate. We employ a Python API integrated in the PettingZoo library in order to facilitate experimentation. The API enables the creation, termination, and configuration of SMARTNet instances. It also enables communication between SMARTNet and Python via RabbitMQ, which is used to enable a remote update to the SMARTNet dissemination ruleset from an external Python process.

The SMARTNet Python Environment wraps a single Interactive SMARTNet instance, and provides multiple versions of its API for different types of RL models/agents. The Nodes in the SMARTNet instance are provided to the RL agent as a batch of samples, enabling a single-agent RL system that supports batched samples to be used in a multi-agent context. Multiple environments can also be batched together to scale the system to many CPUs relatively easily.

For the EA setup, we employ a varying mutation rate of (5%, 10%, 20%), a crossover rate of (40%, 50%, 60%) and a population size of (10, 20, 30) individuals respectively. *Mutation rate* refers to the probability that each new individual will undergo mutation when creating a new generation . The *crossover rate* refers to the probability that an individual will be created by applying the crossover operation to two individuals versus copying an individual from the previous generation. The EA is implemented in Distributed Evolutionary Algorithms in Python (DEAP). The SMARTNet runs used for the EA consider a high congestion setting, similar to the last row in Tables 2, 3, 4, and 5.

Our robustness experiments evaluate the values of the MLM in varied changing network and system conditions. First, we evaluate the performance of the two algorithms against a baseline when a gateway is disconnected. In a less critical scenario, we assume lower level nodes become disconnected from the network. We then evaluate the behavior of the two algorithms when specific message types (text and SOS) suddenly increase in their sending rate. Lastly, we explore a situation where the send interval increases, emulating a situation where the network performance deteriorates during the simulation run.

4.1 Scenario Setup and Baseline

We evaluate the two implementations and their generated SMARTNet dissemination strategies across a range of scenario parameters, including packet send intervals (400ms, 2000ms, 15000ms), network size (13 and 40 nodes) and random seeds (0-99 for 13-node networks, 0-49 for 40-node networks). We use an in-memory network model with a range of fixed packet send intervals to evaluate strategies over a wide range of congestion conditions. The scenarios use a hierarchical network topology where nodes are connected in a ternary tree (see Figure 3). The baseline implementation assumes FIFO message prioritization.

4.2 Single Gateway Disconnection Scenarios

In these scenarios, a single gateway node is disconnected from the rest of the network (node 2) at the mid-point of the scenario. This not only disconnects the gateway node, but also isolates its hierarchical subordinates from the rest of the network. Its subordinates are still able to communicate with each other, however. The effect of this change is most apparent on the 40-node scenarios, as the gateway node has 12 children in this case (compared with only 3 in the 13-node scenarios).

In the low and medium congestion scenarios, the EA and RL strategies perform marginally worse across the board on the gateway-disconnection scenarios, with as small exception in the medium 40 nodes scenario,

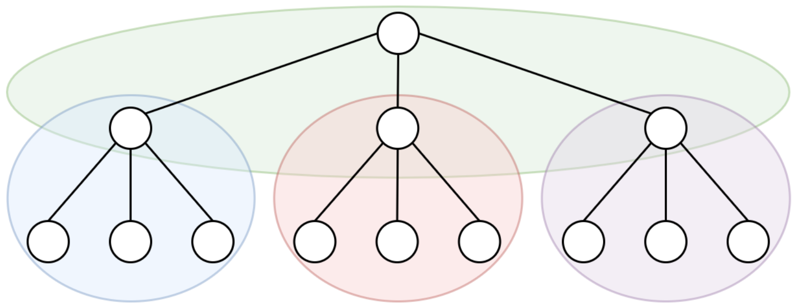

Figure 3: A 13-node hierarchical network topology. Nodes are connected to their operational superiors and ellipses represent fully connected subnets.

Table 2: Final MLM scores in single gateway disconnection.

(Congestion level, # nodes, send interval)	**Baseline**	**Best RL**	**Best EA**
(Low, 13, 0.4s)	-22.22	-22.6	-22.54
(Medium, 40, 2s)	-167.2	**-166**	-174.5
(Low, 40, 0.4s)	-128.2	-128.5	-133.4
(High, 13, 15s)	-109.5	**-86.29**	**-81.34**

where the RL performs slightly better. In the high congestion scenario, both the RL and EA outperform the baseline, with the EA showing a 26% improvement over the baseline, versus a 22% improvement of the RL. However, their margin of difference is significantly narrower for the standard scenarios. This may be a result of improved generalisability of the learned strategies, or the fact that these scenarios represents a more challenging dissemination context than their standard equivalents, meaning that the potential margin for improvement on the strategy's part is much smaller. The RL strategy outperforms the EA strategy, indicating that it can adapt dynamically to system context and is better able to generalise to these scenarios.

4.3 Lower Level 3-node Disconnection Scenarios

These scenarios are similar to the gateway disconnection scenarios, except that several lower-level nodes (nodes 5, 6, and 7 in the blue subnet in Figure 3) are disconnected from the network rather than a single gateway. In the case of the 13-node scenarios, these nodes are leaf nodes, and in the case of the 40-node scenarios they are low-level gateway nodes.

Table 3: Final MLM scores in 3-node disconnection.

(Congestion level, # nodes, send interval)	**Baseline**	**Best RL**	**Best EA**
(Low, 13, 0.4s)	-22.52	-22.91	-22.82
(Medium, 40, 2s)	-199.1	**-197.3**	-204.1
(Low, 40, 0.4s)	-160.5	-160.7	-166.6
(High, 13, 15s)	-109.5	**-86.00**	**-81.80**

In the single gateway disconnection scenarios the performance difference between the strategies is relatively small. The EA still outperforms both other strategies on the high congestion scenario (27% improvement versus 22% improvement of the RL versus the baseline). This may be due to improved generalisation or reduced scope for improvement on this more challenging scenario set. On average, the RL strategy performs better than the EA strategy, indicating better generalisability of the dynamic strategy.

4.4 Text Message Rate Increase Scenarios

In these scenarios, the Text Message generation rate is increased from 10 seconds mean time to happen (MTTH) to 1 second MTTH. Thus, it starts at a lower rate than the standard scenario (3 seconds MTTH) and ends higher. Text messages are relatively large messages (500 bytes), so this increase could test the ability of a strategy to avoid starving other categories when an influx of large messages occurs. Table 4 presents our results. The gap between the strategies is wider as compared with both node-disconnection scenario modifications. However, it is still narrower than for the standard scenarios, indicating again either somewhat improved generalisability of the learned/optimised strategies, or merely a more challenging dissemination context with less scope for improvement. The RL strategy performs significantly better than the EA strategy in this experiment, showing improvements both in the best and in the average MLM score. This indicates that it has improved robustness compared with the static EA strategy.

Table 4: Final MLM scores as message rate increase (Text and SOS).

(Congestion level, # nodes,	**Text Message**			**SOS Message**		
send interval)	**Baseline**	**Best RL**	**Best EA**	**Baseline**	**Best RL**	**Best EA**
(Low, 13, 0.4s)	-0.92	-1.33	-1.23	-0.44	-0.80	-0.74
(Medium, 40, 2s)	-338.6	**-218.5**	**-233.4**	-11.05	-16.11	-34.99
(Low, 40, 0.4s)	-7.10	-7.93	-23.63	-1.77	-2.43	-2.64
(High, 13, 15s)	-160.6	**-144.9**	**-139.9**	-55.89	**-50.80**	**-50.05**

4.5 SOS Rate Increase Scenarios

These scenarios are similar to the Text Message rate increase scenarios, but instead increase the rate of SOS message generation from 5 minutes mean time to happen (MTTH) to 30 seconds MTTH. This represents a lower rate both before and after the change than in the standard scenarios (6 seconds MTTH). SOS messages are small (1 byte) but generally important messages to send quickly.

When subject to increased SOS generation rates, a similar pattern emerges to that of text message increase scenarios. Both strategies still underperform compared with the baseline in the low and medium congestion scenarios, but by a smaller margin than in the standard scenarios, as shown in Table 4. The RL agent performs much better than the EA strategy, again indicating that its adaptive behaviour leads to better generalisation and thus robustness to a range of scenarios. This is in spite of the fact that it was trained on a limited set of scenarios, and that it can only change rulesets based on context and is unable to individually prioritise messages based on their content. Both the RL and EA strategies outperform the baseline on the high congestion SOS rate increase scenarios.

4.6 Send Interval Increase Scenarios

The scenario settings override the send interval, and as such there is only one scenario per network size. The send interval is initially set to 1 second, and is changed to 10 seconds at the mid-point of the run, capturing the situation where the network deteriorates during a run. Table 5 shows our results.

Table 5: Final MLM scores when the send interval increases.

Number of nodes	**Baseline**	**Best RL**	**Best EA**
40	-544.0	**-130.1**	**-252.9**
13	-64.91	**-25.86**	**-27.44**

As it can be seen, the learned/optimised strategies outperform the baseline by a significant margin. The EA agent marginally underperforms the baseline, while the RL agent improves upon it by 24.84%. This is possibly because it is able to generalise to the two send interval values in this experiment that were not present in the original scenario set used to evaluate the baseline.

Our results shows that both reinforcement learning (RL) solutions and evolutionary computation (EA) solutions have the potential to increase the robustness of the SMARTNet middleware when it is running in contested and dynamic environments. In low congestion scenarios, while the RL and EA solutions come close to the baseline, they do not offer any improvements. However, in medium to high congestion scenario, it is the dynamic learning ability of the RL and the optimized nature of the EA that show significant improvements over the static baseline. The average improvements of the solutions range from 13% to 25%.

Despite the EA optimizing a high congestion scenario, the RL solutions significantly outperform the EA solutions, showing better generalisability of the dynamic strategy.

5 CONCLUSION

The SMARTNet middleware optimizes the sending of various message types of different importance across a MANET, with an aim to obtain timely delivery of important messages. We perform an extensive experimental analysis of the robustness of reinforcement learning (RL) and evolutionary computation (EA) solutions in various scenarios typical of a contested and dynamic environment, including gateway and individual node failure, and deterioration of network performance.

Our analysis shows that while the RL and EA show no improvements in low congestion scenarios, we see some improvements (up to 36%) in some medium congestion scenarios, and consistent significant improvements over the baseline (up to 25%) in high congestion scenarios. In the case where the network performance deteriorates, the RL strategy offers an improvement of 61% over the baseline, in the 40 node scenario. These results are very promising, showing the benefits of artificial intelligence approaches in improving middelware performance in MANETs, while considering contested and dynamic environments. In addition, an AI approach has significant promise to be more generalisable and thus applicable to varied scenarios. Despite its improvements, the current reinforcement learning implementation is single-agent, assuming all agents are identical and observe the environment in the same way. This is not the case when there are significant network partitions or when adversarial agents are present. Our future work includes the implementation and evaluation of a multi-agent reinforcement learning algorithm in these environments.

REFERENCES

Abdou, W., C. Bloch, D. Charlet, D. Dhoutaut, and F. Spies. 2011. “Designing Smart Adaptive Flooding in MANET using Evolutionary Algorithm”. In *International Conference on Mobile Wireless Middleware, Operating Systems, and Applications*, edited by N. Venkatasubramanian, V. Getov, and S. Steglich, 71–84. Berlin, Heidelberg: Springer.

Ali, A. K. S., and U. Kulkarni. 2015. “Characteristics, Applications and Challenges in Mobile Ad-Hoc Networks (MANET): Overview”. *IPASJ International Journal of Electronics & Communication* 3(12):006–012.

Campbell, Angus LTGEN 2018. “Australia’s Joint Force Land Capability – Address by Chief of Army, Lieutenant General Angus Campbell, to Australian Defence Magazine Congress”. February 14th, Canberra.

Chan, K., K. Marcus, G. Judd, and P. Boyd. EasyChair, 2018. “Semantically Managed Autonomous and Resilient Tactical Networking (SMARTNET) and Hybrid C2 operations”. EasyChair Preprint no. 670.

Ghasemnezhad, S., and A. Ghaffari. 2018. “Fuzzy Logic Based Reliable and Real-time Routing Protocol for Mobile Ad Hoc Networks”. *Wireless Personal Communications* 98(1):593–611.

Ghouti, L., T. R. Sheltami, and K. S. Alutaibi. 2013. “Mobility Prediction in Mobile Ad Hoc Networks Using Extreme Learning Machines”. *Procedia Computer Science* 19:305–312.

Gudakahriz, S. J., S. Jamali, and E. Zeinali. 2011. “NISR: A Nature Inspired Scalable Routing Protocol for Mobile Ad Hoc Networks”. *Journal of Computer Science Engineering and Technology* 1(4):180–184.

Günes, M., M. Kähmer, and I. Bouazizi. 2003. “Ant-Routing-Algorithm (ARA) for Mobile Multi-hop Ad-hoc Networks - New Features and Results”. In *The Second Mediterranean Workshop on Ad-hoc Networks*. June 25th-27th, Mahdia, Tunisia.

Ivanic, N., B. Rivera, and B. Adamson. 2009. “Mobile Ad Hoc Network Emulation Environment”. In *MILCOM 2009-2009 IEEE Military Communications Conference*, 1–6. Manhattan, New York: Institute of Electrical and Electronics Engineers.

Li, F., X. Song, H. Chen, X. Li, and Y. Wang. 2018. “Hierarchical Routing for Vehicular Ad Hoc Networks via Reinforcement Learning”. *IEEE Transactions on Vehicular Technology* 68(2):1852–1865.

Liu, Y., and J. Huang. 2009. "A Novel Fast Multi-objective Evolutionary Algorithm for QoS Multicast Routing in MANET". *International Journal of Computational Intelligence Systems* 2(3):288–297.

Quin, F., T. Bamelis, S. B. Sarpreet, and S. Michiels. 2019. "Efficient Analysis of Large Adaptation Spaces in Self-adaptive Systems Using Machine Learning". In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 1–12. Manhattan, New York: Institute of Electrical and Electronics Engineers.

Rajabhushanam, C., and A. Kathirvel. 2011. "Survey of Wireless MANET Application in Battlefield Operations". *International Journal of Advanced Computer Science and Applications (IJACSA)* 2(1).

Rath, M., and B. K. Pattanayak. 2014. "A Methodical Survey on Real Time Applications in MANETS: Focussing on Key Issues". In *2014 International Conference on High Performance Computing and Applications (ICHPCA)*, 1–5. Manhattan, New York: Institute of Electrical and Electronics Engineers.

Reina, D., M. Askalani, S. Toral, F. Barrero, E. Asimakopoulou, and N. Bessis. 2015. "A Survey on Multihop Ad Hoc Networks for Disaster Response Scenarios". *International Journal of Distributed Sensor Networks* 11(10):647037.

Saleem, M., I. Ullah, and M. Farooq. 2012. "BeeSensor: An Energy-efficient and Scalable Routing Protocol for Wireless Sensor Networks". *Information Sciences* 200:38–56.

Szabo, C., V. Radenovic, G. Judd, D. Craggs, K. L. Lee, X. Chen, and K. Chan. 2020. "Optimizing Communication Strategies in Contested and Dynamic Environments". In *2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS)*, 197–205. Manhattan, New York: Institute of Electrical and Electronics Engineers.

Talawar, M. B., and D. Ashoka. 2020. "Link Failure Detection in MANET: A Survey". *Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough*:169–182.

Terry, J. K., B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sulivan, L. Santos, R. Perez, C. Horsch, C. Dieffendahl, N. L. Williams, Y. Lokesh, R. Sullivan, and P. Ravi. 2020. "PettingZoo: Gym for Multi-Agent Reinforcement Learning". *arXiv preprint arXiv:2009.14471*.

Thangaramya, K., K. Kulothungan, R. Logambigai, M. Selvi, S. Ganapathy, and A. Kannan. 2019. "Energy Aware Cluster and Neuro-fuzzy Based Routing Algorithm for Wireless Sensor Networks in IoT". *Computer Networks* 151:211–223.

United States Naval Research Laboratory 1996. *EMANE Network Emulator*. https://www.nrl.navy.mil/itd/ncs/products/emane, accessed: April 12$^{\text{th}}$ 2022.

Wolf, S., and P. Merz. 2009. "Iterated Local Search for Minimum Power Symmetric Connectivity in Wireless Networks". In *European Conference on Evolutionary Computation in Combinatorial Optimization*, edited by C. Cotta and P. Cowling, 192–203. Berlin, Heidelberg: Springer.

Wu, X., X. Wang, and R. Liu. 2008. "Solving Minimum Power Broadcast Problem in Wireless Ad-hoc Networks Using Genetic Algorithm". In *6th Annual Communication Networks and Services Research Conference (cnsr 2008)*, 203–207. Manhattan, New York: Institute of Electrical and Electronics Engineers.

Yetgin, H., K. T. K. Cheung, and L. Hanzo. 2012. "Multi-objective Routing Optimization Using Evolutionary Algorithms". In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, 3030–3034. Manhattan, New York: Institute of Electrical and Electronics Engineers.

AUTHOR BIOGRAPHIES

CLAUDIA SZABO is an Associate Professor in the School of Computer Science and the Lead of the Complex Systems Research Group at the University of Adelaide. Her research interests lie in the area of complex systems and on using simulation to identify and validate their emergent properties. Her email address is claudia.szabo@adelaide.edu.au.

DUSTIN CRAGGS is a researcher and software developer in the Complex Systems Research Group at the University of Adelaide. His email address is dustin.craggs@adelaide.edu.au.

DUMITRU ALIN BALASOIU is a software engineer working in the Complex Systems Research Group at the University of Adelaide. His interests include data science, security and microservices. dumitru.balasoiu@adelaide.edu.au

VANJA RADENOVIC is a Software Chapter Lead at Consilium Technology and has over a decade of experience in defence research and software engineering projects. His research interests are in command and control (C2) systems and tactical networking. His email address is vanja.radenovic@consilium.technology.

BENJAMIN CAMPBELL is the Discipline Leader autonomous C5ISREW in Land Division at DSTG and the DSTG technical lead for the Semantically Managed and Resilient Tactical Networking (SMARTNet) research activity. His research interests are in distributed control of land vehicle mission systems, distributed autonomy and RF signature management. His email address is benjamin.campbell5@defence.gov.au.