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.

USE OF REINFORCEMENT LEARNING FOR PRIORITIZING COMMUNICATIONS IN CONTESTED AND DYNAMIC ENVIRONMENTS

Dustin Craggs
Kin Leong Lee
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

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

ABSTRACT

Systems operating in military operations and crisis situations usually do so in contested and dynamic environments with poor and unreliable network conditions. Individual nodes within these systems usually have an incomplete, local and changing view of the system and its operating environment, and as such optimizing how nodes communicate in order to improve decision making is critical. In this paper, we propose the integration of reinforcement learning algorithms with the SMARTNet middleware, a middleware that prioritizes and controls messages sent by each node, allowing it to determine the best priority for each message type. We experiment with both direct and indirect prioritisation approaches, where the reinforcement learning dissemination system determines the specific priority of a message on its arrival or upon its sending respectively. Our experimental analysis show significant improvements over the baseline in some of the high congestion scenarios but also highlights several avenues for future work.

1 INTRODUCTION

Ensuring timely messages arrive without massive network disruptions is critical to rescue and military operations in contested and dynamic environments, where the network is unavailable or jammed (Campbell, Angus LTGEN 2018). 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 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 need to be responsive to frequent network capability changes due to their operation in a disrupted environment with limited infrastructure. One of the most frequently sent message type across these networks 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. Within these contested networks operating in dynamic environments, there is a need for assurance of timely delivery, responsiveness and graceful degradation mechanisms that can manage the fragility of the communication networks.

One way to achieve this is by designing node-specific rules or behaviors relying on the partial or complete understanding that a node has of the system context. The node would then be able to reason over the 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, and that the local node decisions will then lead to a complex system with a desired emergent behavior.

Several approaches that design complex systems with emergent properties have been proposed, including systems-of-systems design (Falkner et al. 2018; Kewley et al. 2008), mechanism design and game theory (Park et al. 2000), and agent-based simulation (Bernon et al. 2003; Mittal 2013),(Jacyno et al. 2009; Salazar et al. 2011; Kolen et al. 2018) approaches. While in general successful in achieving the desired emergent property, most of these approaches are evaluated in small scale scenarios and tend to be deployed in stable environments whose properties do not vary beyond well defined ranges. In addition, no side-effect analysis is present (Szabo and Teo 2016), such as the potential of firing on friendly forces due to reduced situation awareness due to missing delivery of critical messages.

SMARTNet (Chan et al. 2018) 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, communication 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 propose to use machine learning algorithms to ensure that the SMARTNet middleware adapts well to context changes, in particular those with medium and high network congestion. The use of machine learning algorithms to improve the performance and security of MANETs has recently been explored by a number of studies. In this paper, we report on the integration of two machine learning implementations into SMARTNet. We employ a reinforcement learning algorithm in two ways to explore the best moment when a machine learning algorithm can be deployed. Firstly, we employ a DQN algorithm to set the message priority when the message arrives at each agents' queue, before it is allocated to a packet. This approach works well in networks that are not congested, as the time between when the message arrives and when it is allocated to a packet is very short, and thus network conditions do not change. Secondly, we employ a soft actor-critic (SAC) approach to determine the ruleset by which a message is prioritised right before it is allocated to a packet. The contributions of this industry experience report are twofold:

- Two reinforcement learning algorithms to reduce network congestion in the SMARTNet middleware while achieving timely delivery of messages
- An extensive experimental analysis of the improvements of the two algorithms over a baseline in a variety of simulation scenarios

2 BACKGROUND AND RELATED WORK

A mobile ad-hoc network (MANET) is a decentralised wireless network with two key characteristics, namely, that it is composed of mobile devices and that it operates without support from static infrastructure

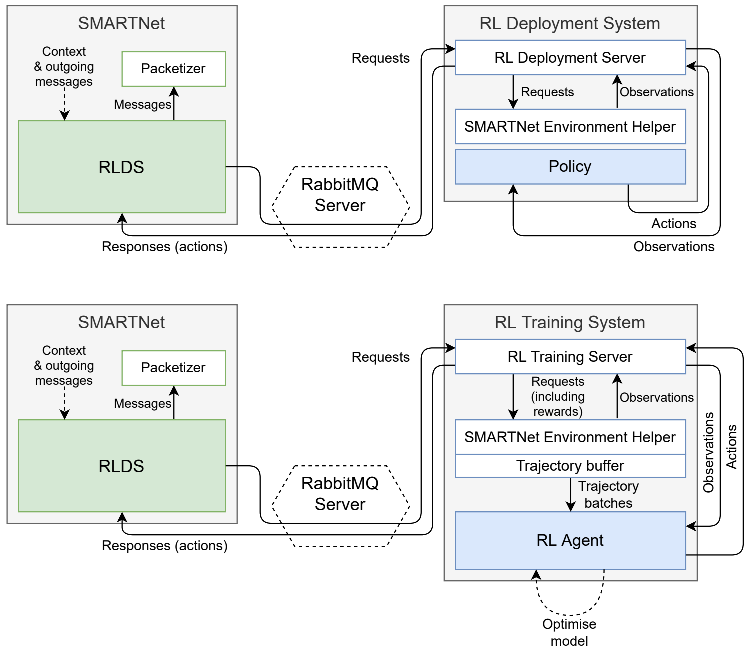

Figure 1: Architectural Overview of the RLDS and RL Training and Deployment Systems

such as routers or servers (Ali and Kulkarni 2015). Instead of using static infrastructure, MANET nodes provide routing and services to each other in a system similar to a peer-to-peer network. Due to their lack of reliance on fixed infrastructure, MANETs are utilised in contested and resource-constrained environments such as crisis areas (Reina et al. 2015) and battlespace (Rajabhushanam and Kathirvel 2011; Rath and Pattanayak 2014) scenarios, where they can be deployed quickly to address emergent situations.

2.1 SMARTNet Overview

The Semantically Managed Autonomous and Resilient Network (SMARTNet) (Chan et al. 2018) is a middleware developed to prioritise, transform, and control messages passed through each node to achieve timely delivery of operationally important information while maintaining critical network performance. SMARTNet allows its nodes to have enhanced situational awareness of mission context and network state through the integration of various information sources. This allows nodes to make more informed decisions regarding message transformation and routing (Quin et al. 2019).

Within each SMARTNet node, several modules interact in order to achieve the desired goals, 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. SMARTNet has access 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; EMANE Network Emulator 1996), and real life radios, which connect to SMARTNet through plugins.

The types of messages that can be sent in the SMARTNet middleware include Position Location Information (PLI), enemy detection, text, video and operational context messages. The main responsibility of the SMARTNet communication strategy module 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 or detection (RedDetection) 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 of the battle and other nodes respectively.

The application of machine learning and artificial intelligence algorithms has expanded in recent years (Talawar and Ashoka 2020; Li et al. 2018), with an increased focus on security (Talawar and Ashoka 2020) and routing protocols (Wu et al. 2008; Gudakahriz et al. 2011; Li et al. 2018; Ghouti et al. 2013). Wu et al. (2008) propose an iterated local search to find the minimum spanning tree to connect all nodes in the network, while Wolf and Merz (2009) attempts to achieve minimum power broadcast. Both approaches are centralized offline approaches and require global knowledge in order to converge. In contrast, . Günes et al. (2003) propose to use ant colonies in a routing algorithm that requires only local knowledge. Other examples of algorithms that employ local knowledge rely on ant, bee or swarm abstractions (Gudakahriz et al. 2011; Saleem et al. 2012). Several papers are focused on machine learning for MANET routing protocols and their evaluation in various simulated scenarios. Li et al. (2018) use reinforcement learning in a vehicular MANET to improve the message delivery ratio with minimum possible delay and hops. The protocol divides the geographical area into smaller grids and finds the next optimal grid toward the destination. Second, it discovers a vehicle inside or moving toward the next optimal grid for message relaying. Ghouti et al. (2013) employ user mobility models and extreme learning machines to design a mobile adhoc network that is enriched with the prediction of its mobility. The analysis finds that the approach is limited by the prediction accuracy while calculating the distance between the neighboring nodes. In wireless sensor networks, a CNN trained fuzzy network is used by Thangaramya et al. (2019) towards energy aware message routing. The approach is analysed in two simple scenarios.

3 REINFORCEMENT LEARNING IN THE SMARTNet MIDDLEWARE AND SIMULATION SYSTEM

We propose the integration of a reinforcement learning system within the SMARTNet middleware in order to improve the behavior of the middleware across a variety of network load scenarios. Our proposed reinforcement learning system consists of three main components: the Reinforcement Learning Dissemination System SMARTNet module (RLDS), the Reinforcement Learning Training System (RL Training system), and the Reinforcement Learning Deployment System (RL Deployment System) as shown in Figure 1.

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.

3.1 Reinforcement Learning Dissemination System (RLDS)

The RLDS module is responsible for providing a Message Ordering implementation within SMARTNet, determining the order in which outgoing messages are provided for packetisation. The RL Training and the RL deployment systems require information about the messages being prioritised as well as the current context. The RLDS is responsible for transforming this information into a normalised representation suitable for the RL training system. To address different types of scenarios with respect to network congestion, we propose and evaluate two RLDS implementations with different approaches to prioritisation of messages, namely, direct and ruleset-based.

Direct prioritisation RLDS This implementation enforces a priority-based ordering on messages based on the current context, where the RLDS delegates prioritisation decisions to the RL Training/Deployment system. A primary limitation of this approach is that it assigns priorities *on arrival* of messages, rather than *on creation of the packet*. In the time after the message is prioritised, the context changes and newer messages arrive, potentially rendering the priority values outdated.

Ruleset-based RLDS When using the ruleset-based RLDS, the RL agent is responsible for determining the parameters of a prioritisation ruleset. This ruleset is used to map each message to a priority based on the properties of that message. This is done right before each packet generation. The agent is able to change the priority values for each of the message property combinations to adapt to different node contexts. We explore basic and extended rulesets that capture different context settings.

The **RL Training System** is responsible for providing actions in the form of priority values for a message to the RLDS, while simultaneously training a model based on feedback from the RLDS in the form of rewards. The RLDS and RL Training System communicate using the RabbitMQ message broker. A Remote Procedure Call pattern has been implemented over the RabbitMQ message broker to simplify the process of acquiring the next action. Requests from the RLDS are encoded as JSON objects. Replies are currently encoded as comma separated lists of numbers.

The RL Training system supports one type of request, called GET_PRIORITY. The request contains information about the observation and the reward. The response, sent back to the RLDS from the RL Training System, is either a single integer in the range [0, 9] representing the priority value that should be assigned to the message in the request (direct prioritisation) or a comma separated list of floating point numbers in the range [0, 9], with one value per unique combination of the message properties defined in the RLDS (ruleset-based). These values represent the actions for each RLDS implementation, which are discussed in more detail in Section 3.2. There are two main components to the RL Training system, namely, the RL Training Server and the RL Agent. The RL Training Server component sets up a RabbitMQ connection and services GET_PRIORITY requests from a SMARTNet instance that use the RLDS. It provides the Observation, Reward, and Step Type from each request to the SMARTNet Environment Helper to acquire an object representing the current time step. It queries the RL Agent using this object to obtain the action (priority value) to send back to the RLDS in the response. Finally, it passes the time step object and action to the SMARTNet Environment Helper to be converted to a Trajectory object suitable for training, and added to the buffer. The model is evaluated periodically during training in order to assess its performance. This evaluation consists of one or more complete runs over a SMARTNet scenario. It is necessary to use a separate instance of SMARTNet for this evaluation, as evaluating on the same instance would interrupt the training process by resetting the run.

The **RL Agent** is responsible for predicting the optimal action given a provided observation using a model (such as a Deep Q-Network), as well as for training this model based on incremental reward values. The RL Agent's training process runs in a separate thread, waiting for Trajectories to be added to the Trajectory Buffer in the SMARTNet Environment Helper. When it receives a trajectory, it adds it to its own training buffer and performs a training step. 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 comparisons to other strategies.

The RL Agent uses a collect policy for training. This policy is used to ensure that the agent explores the space of possible actions, rather than always choosing an action that is currently estimated to be optimal. The collect policy that is currently used by the agent is epsilon-greedy, which takes a single parameter, epsilon representing the probability that the agent will pick a random action rather than using the action that is currently estimated to be optimal by the underlying model.

3.2 OBSERVATIONS AND ACTIONS

Direct Prioritisation with DQN Table 1 captures the observations needed for the direct prioritisation implementation. Each of the context properties is the context that is local to the node that is prioritising the message. The contexts are constrained by a maximum value and then normalised. The action for the direct prioritisation RLDS is setting a single priority p, where $p \in [0,9], p \in \mathbb{Z}$.

Basic ruleset-based action space One of the major limitations of the priority-based action-space is the

Table 1: Description of observation properties in direct prioritisation.

Observation property name	**Description**
Message Properties	
is_pli	1 if the current message is a PLI, else 0
is_red	1 if the current message is a Red Spot, else 0
is_tg	1 if the current message is a Tactical Graphic (non-Red), else 0
is_text	1 if the current message is a Text Message, else 0
is_sos	1 if the current message is a SOS Message, else 0
age	Age of the message. Constrained to 60 seconds and then normalised to [0, 1].
is_routed	0 if the message originated at the sender, 1 if the message is being routed at a gateway.
queue_size	The number of messages in the message store of the node. Constrained to 1000 and then normalised between 0 and 1.
Context Properties	
avg_distance_to_others	The average current known distance (as in physical distance) to all other nodes. Constrained to 10 km and then normalised between 0 and 1.
last_SOS_seconds	The time, in seconds, since the last known SOS message received from the sender was created. Constrained to 10 seconds, then normalised between 0 and 1. Defaults to 1 if no SOS has been sent.
time_since_last_packet_sent	The number of seconds since the last packet was sent. Constrained to 1000 seconds, then normalised between 0 and 1. If no packet has been sent, defaults to 1.
number_of_nodes	Number of nodes in the system. Maximum 150 and normalised to [0, 1].
distance_since_last_pli	Distance between current position and position of last sent PLI. Constrained to 10 km and then normalised to [0, 1].

assignment of priorities at the time that the message is added to the outgoing queue, rather than when a packet is generated. This leads to situations where new messages or changes to the context render previously assigned priorities sub-optimal. Ideally, all messages would be prioritised (or re-prioritised) just prior to packet generation. In addition, prioritising every message individually upon its arrival at the message store could also make training an effective RLDS model more difficult. The effects of each action are delayed in this case, as messages will not be sent for some time after prioritisation. When congestion is present, many messages will never be sent, meaning that many actions have only a very minimal effect on the environment. The training of the model is easier when the effect of actions has a more immediate impact on the environment and consequently the rewards.

In the ruleset-based approach, the agent uses the current node context to determine the priority values of a ruleset that is applied to the outgoing messages prior to packet generation. Rather than an action being the assignment of a single priority to a message, the action is to determine a ruleset that will then be used by SMARTNet to assign the individual priorities prior to packet generation. The ruleset is represented as a matrix of priority values. Priorities can be integers in some fixed range, namely, $p \in [0,9], p \in \mathbb{Z}$. Table 2 presents the observation properties used by the agent to determine ruleset priorities.

Several message properties could be used to enable the RLDS to learn to perform more complex behaviours, however many are continuous values, so one or more breakpoints will need to be defined to determine when their value should impact prioritisation. Alternatively, the breakpoints for each message context property could also be assigned as a learnable parameter. There are a number of possible message properties that may be worth considering, such as whether the message is newest/oldest of its category, the age of the message, the distance moved since creation, the origin and destination of the message, message size, or the status of the node of origin, e.g., an SOS node or a high network congestion. Table 3 presents the basic ruleset used in this implementation. In future work, non-priority parameters could also be added to the ruleset, such as an update rate and update distance for messages, a data rate for the admission controller, or a situation or threshold at which to expire messages early.

Table 2: Description of observation properties in ruleset-based prioritisation.

Observation property name	**Description**
Context properties	
queue_size	The number of messages in the message store of the node. Constrained to 1,000 and then normalised between 0 and 1.
avg_distance_to_others	The average current known distance (as in physical distance) to all other nodes. Constrained to 10 km and then normalised between 0 and 1.
number_of_nodes	Number of nodes in the ORBAT. Constrained to 150 and normalised to [0, 1].

Table 3: Basic ruleset-based action space.

Ruleset property name	**Property values per message type**		
	PLI	**Red**	**Text**
age	$\leq 10s$	between 10 and 30 s	$\geq 30s$
is_routed	Current sender originated message	Message routed at gateway	
sender_has_recent_sos	No SOS received in the last 5m	SOS received in the last 5m	

Table 3 outlines the message properties that are used by the basic ruleset-based action implementation to assign each message to a priority. The action must contain a priority for each unique combination of the possible property values (e.g. a Red position message that is 25 seconds old, is being routed at a gateway, and was sent from a node that has recently sent an SOS message). Thus, in this case, the action space of the agent is a set of 60 priorities (as there are 60 unique combinations of values of the 5 message categories, 3 age categories, 2 is_routed categories, and 2 sender_has_recent_sos categories. Every message is assigned, according to its properties, the corresponding priority from this set of 60). By default, the Soft Actor-Critic agent used in this implementation works with continuous action spaces, so rather than constraining these priorities to integral values, floats in the range [0, 9] are used.

Extended ruleset-based action space We extend the basic ruleset-based approach by growing the observation space and by using a larger ruleset, with an additional property related to the message's position: `newest/oldest`. For PLI, this means the oldest message of category from the sender. For Red, this is neither olders nor newest, while for Text this is the newest from the sender. The observation space is extended as shown in Table 4. There are a total of 180 different combinations of unique property values (with newest/oldest, age, is_routed, and sender_has_recent_sos) in the extended ruleset).

Rewards The rewards are based on the value of a calculated Mid Level Metric (MLM). Every observation is passed to the RL Training System alongside a reward. Each reward is the sum of MLM values generated since the last observation. If multiple observations are passed to the RL Training System with no penalties generated by the MLM in the intervening period, the reward accompanying those observations is 0.

The MLM represents a snapshot of the global information accessibility for nodes in the system at a given time. It is computed as an accumulated sum of penalties for each piece of information in the system that has not yet been delivered to its intended destination. The magnitude of each penalty is based on how relevant/stale the missing information is, and is modified by operational context. Each of the five broad information categories in the system, namely, PLIs, Enemy Detections, Text Messages, Tactical Graphics, and SOS messages, is treated differently when calculating penalties, and the specific treatment is determined

Table 4: Extended observation properties.

Observation Property	**Description**
Context properties	
queue_size	The number of messages in the message store of the node. Constrained to 1000 and then normalised between 0 and 1.
avg_distance_to_others	The average current known distance (as in physical distance) to all other nodes. Constrained to 10 km and then normalised between 0 and 1.
number_of_nodes	Number of nodes in the ORBAT. Constrained to 150 and normalised to [0, 1].
time_since_last_packet	Time since the last packet was sent by the requesting node. Constrained to 60 seconds and normalised to [0, 1].
average_hierarchical_distance	The average hierarchical distance between the requesting node and other nodes in the scenario.

based on subject matter expert advice. The goal of the SMARTNet dissemination system is to minimise the sum of these penalties across a range of different scenarios.

We employ two types of agent implementations within the RL Agent, namely, Deep Q-Network (DQN), and Soft Actor-Critic (SAC). The DQN agents are suitable for use with relatively simple, discrete action spaces, such as the direct prioritisation action space. The SAC agent is useful when optimising the basic and extended ruleset-based action space, as it supports continuous, multidimensional action spaces. The SAC model has also been extended to sample trajectories from multiple SMARTNet environments in a round-robin order. This speeds up training, allowing multiple SMARTNet instances to run in parallel, without introducing synchronisation issues. However, this has limitations, such as when one or more environments are running slower than others (e.g. a 13-node and 40 node scenario running in parallel). Since the agent cycles through each environment during, blocking on a trajectory request, the agent frequently has to wait for the slowest environment to progress.

The RL Training system can be run by starting an instance of the Training Server and a SMARTNet run. The Training Server creates a number of SMARTNet environment wrappers and an RL Agent, and starts the RL Agent's training procedure. The experiments run a large number of SMARTNet runs so that the RL Training system will constantly be provided with observations/rewards until its training is complete. General training parameters include the number of iterations and trajectory collection steps per iterations, epsilon, gamma, learning rate, and network parameters for the DQN agents, as well as reward scale, actor and critic network parameters, and actor and critic learning rates for the SAC agents.

4 EVALUATION

We evaluate the two machine learning 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 different fixed packet send intervals in order 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 2).

Results from each combination (network size, send interval) are averaged over a number of random seed values, where each random seed represents a unique scenario with varying node movement and traffic generation patterns. 13-node network hierarchies were evaluated over 100 different random seed values. 40-node hierarchies were experimentally confirmed to exhibit lower variance, and thus a reduced set of 50 random seeds was sufficient when evaluating these scenarios. We select a sample combination of number of nodes, congestion level and send interval for these experiments, as shown in Table 5.

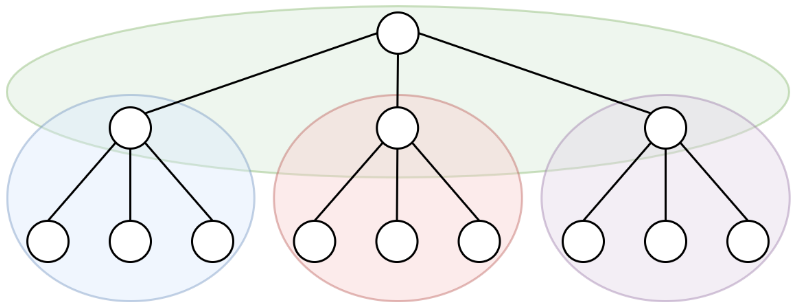

Figure 2: A 13-node hierarchical network topology. Nodes are connected to their operational superiors and ellipses represent fully connected subnets.

Agent policies are compared to a preexisting set of baseline dissemination strategies in order to evaluate their performance. These baseline strategies use a simple, static ruleset composed of fixed priorities for each message category as well as a time and distance based rate limiter for Position Location Information (PLI) messages. This rate limiter is parameterised by fixed time elapsed and distance travelled thresholds that must be satisfied before sending a PLI message.

Table 5: Percentage improvement of DQN, Direct Basic and Advanced Soft Actor-Critic experiment.

(Congestion level, # nodes, send interval)	$pi_{DQN\_baseline}$	$pi_{SAC1\_baseline}$	$pi_{SAC2\_baseline}$
(Low, 13, 0.4s)	72.5%	71.0%	73.8%
(Medium, 40, 2s)	-1400%	-166%	-50%
(Low, 40, 0.4s)	49.0%	45.7%	47.1%
(High, 13, 15s)	-59%	10.06%	-56.9%

The parameters for these baseline strategies were optimised using a grid-search approach to arrive at a set of static baseline strategies. Based on the search, the best overall baseline was selected across all scenarios, as well as the best strategy for each specific combination of network size and send interval. The strategies were selected based on their performance over a reduced set of random seeds. They were then re-evaluated on the full set of evaluation scenarios to obtain their final performance measurements. They represent highly suitable static strategies for this set of scenarios.

We define a high-level evaluation metric used to rank SMARTNet dissemination strategies across all scenario parameters. Since the magnitude of the total MLM scores varies drastically across different scenario parameter values, each strategy's performance on relatively high-penalty scenarios would dominate the final metric, making the contribution of some of the other scenarios insignificant. To address this, we focus on a strategy's percentage improvement over a baseline, and use the mean across all percentage improvements as the main evaluation metric. For a strategy s we define the percentage improvement pi over a baseline as follows:

$$pi_{s\_baseline} = (1 - \frac{MLM(s)}{MLM(baseline)}) * 100$$

The percentage improvement may be negative, indicating that the performance of the strategy under examination was worse than the baseline for that scenario. This value is calculated for each unique combination of scenario parameters, and the mean across all percentage improvements for a particular strategy across scenarios will be used as the final metric with which to compare strategies. It is important to note that this metric treats each scenario as equally important. However, scenarios could be weighted by importance in this metric and a weighted average be computed across all scenarios.

Direct prioritisation with DQN A sample model was trained using the current system for 150,000 iterations, $\varepsilon = 0.1$, $\gamma = 0.01$ and 10 trajectory collection steps per iteration. The model was evaluated during training on a single shortened 13-node scenario at the beginning and every 40000 training iterations. The trained model was evaluated using the evaluation system on a wider range of scenarios, as shown in Table 5 ($pi_{DQN\_baseline}$).

The model performs well for the low congestion scenarios (with 13 and 40 nodes respectively), and otherwise underperforms the baseline, in some cases severely. The policy performs well for low congestion scenarios, and poorly for medium and high congestion scenarios.

Basic Ruleset-based prioritisation with SAC In this experiment, the ruleset-based approach is used with the Soft Actor-Critic agent. This experiment was trained using a single training SMARTNet instance, with 13-node, medium and high congestion scenarios with 10 different random seed values, and evaluated on networks of size 13 and 40 nodes. A summary of the hyperparameters used in this experiment are given in Table 6. Similarly to the previous experiment, in order to reduce training time, only a single scenario is

Table 6: Hyperparameters for the basic Soft Actor-Critic experiment.

Parameter	**Value**
Number of iterations	500,000
Actor & critic network layer widths	256, 256
Reward scale	1.0
Tau	0.05
Replay buffer capacity	100,000
Batch size	256
Learning rates	3e-4
Collect steps per iteration	1

used to evaluate the model during training, so this plot may not be representative of average performance across the training data. However, its performance on this single scenario throughout training indicates that, while it tended to improve overall over the random policy, its performance was quite variable.

Table 5 presents the evaluation of the basic and advanced soft actor-critic approach against the baseline ($pi_{SAC1\_baseline}$).

The policy trained in this experiment performs significantly better for first scenario than the direct prioritisation DQN model did, but still fails to outperform the baselines for this scenario. It also outperforms the previous model in high congestion scenario, as well as the overall baseline.

Advanced Ruleset-based prioritisation with SAC This experiment uses five simultaneous training instances that are sampled from cyclically during training. This increased the rate at which training progressed, but presents a possibly more varied and more challenging set of samples in the replay buffer at any given time. In addition, in order to allow the SMARTNet instance to progress while it is awaiting a reply from the RL system, the request for the next ruleset is made immediately after sending a packet rather than at the time of packet generation. Thus, the observation provided to the agent is always exactly one send interval behind the actual observation. The policy learned in this experiment shown in Table 5 ($pi_{SAC2\_baseline}$) performs poorly in the high congestion scenario when compared both with the baselines and with the policy from the basic ruleset-based SAC implementation, with significant improvements only in the low congestion scenarios. This may be due to the training on a range of different scenarios simultaneously, which has an effect on the way the training of the agent compared with training on those same scenarios sequentially. More investigation will be required to identify how the variability in samples may improve or degrade learning performance.

Our results show that the use of machine learning approaches offers a significant improvement over the baseline, in particular for low congestion scenarios with a large number of nodes (73% improvement). High congestion scenarios show small improvements in some cases, and no improvements were shown in medium congestion scenarios. This may be due to a number of reasons, including the lack of extensive context information and the need for more extensive training, in particular for medium congestion, 40 nodes scenarios. Secondly, expanding the range of contexts incorporated into the observations may be extremely important improving the performance of the RLDS. This will ensure the contexts will be useful in determining the optimal priority, but also for predicting future rewards to assist with training. This could include, for example, operational context information such as the number of enemies in the vicinity, or whether the node is a gateway node. Another critical improvement refers to computing a reward metric that is less scenario specific. Currently, the reward function is highly variable when training on different scenarios, with larger scenarios (with a higher number of nodes or higher congestion) resulting in much higher MLM penalties. This may affect the ability of the RLDS to generalise across a range of scenarios.

5 CONCLUSION

We explore the use of reinforcement learning to optimize the message sending behavior of the SMARTNet middleware. The 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 employ two soft actor-critic approaches that set the message priority right before the message is sent. The two approaches employ a basic and an advanced ruleset respectively to determine the message priority based on the current context. Our analysis shows that the reinforcement learning approach is promising, showing significant improvements over a baseline in low congestion scenarios with a large number of nodes. We have been able to obtain improvements only in a few high congestion scenarios, where potentially more complex rulesets for the SAC aproach could be used. In addition to the current parameters, these could consider network context, an estimate of the number of enemies in the vicinity, and information about the sender node, such as whether it is a gateway node.

REFERENCES

Ali, A. K. S., and U. Kulkarni. 2015. "Characteristics, Applications and Challenges in Mobile Ad-Hoc Networks (MANET): Overview". *Wireless Networks* 3(12).

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".

Chan, K., K. Marcus, G. Judd, and P. Boyd. 2018. "Semantically Managed Autonomous and Resilient Tactical Networking (SMARTNET) and Hybrid C2 operations". Technical report.

EMANE Network Emulator 1996. "https://www.nrl.navy.mil/itd/ncs/products/emane".

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.

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, 9-20.

Ivanic, N., B. Rivera, and B. Adamson. 2009. "Mobile Ad Hoc Network Emulation Environment". In *Proceedings of 2009 IEEE Military Communications Conference (MILCOM)*, 1–6. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

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: Association for Computing Machinery.

Kewley, R., J. Cook, N. Goerger, D. Henderson, and E. Teague. 2008. “Federated Simulations for Systems of Systems Integration”. In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Mönch, T. J. O. Rose, and J. W. Fowler, 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*.

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.

Mittal, S. 2013. “Emergence in Stigmergic and Complex Adaptive Systems: A Formal Discrete Event Systems Perspective”. *Cognitive Systems Research* 21:22–39.

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.

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 *Proceedings of 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 1–12. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

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

Rath, M., and B. K. Pattanayak. 2014. “A Methodical Survey on Real Time Applications in Manets: Focussing on Key Issues”. In *Proceedings of 2014 International Conference on High Performance Computing and Applications (ICHPCA)*, 1–5. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

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.

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: Association for Computing Machinery.

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 *25th International Conference on Engineering of Complex Computer Systems*, 197–205. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

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

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.

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.

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*, 192–203. Springer.

Wu, X., X. Wang, and R. Liu. 2008. “Solving Minimum Power Broadcast Problem in Wireless Ad-Hoc Networks using Genetic Algorithm”. In *Annual Communication Networks and Services Research Conference*, 203–207. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

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

Kin Leong Lee is a software developer in the Complex Systems Research Group at the University of Adelaide. His email address is kinleong.lee@adelaide.edu.au.

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.

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 is the DSTG technical lead for the Semantically Managed and Resilient Tactical Networking (SMARTNet) research activity. He has research interests in distributed control of land vehicle mission systems, distributed autonomy and RF signature management. His email address is benjamin.campbell5@defence.gov.au.