Learning an Adaptive Forwarding Strategy for Mobile Wireless Networks: Resource Usage vs. Latency

Victoria Manfredi
Dept. of Math and Computer Science
Wesleyan University
Middletown, CT, 065459 USA
vumanfredi@wesleyan.edu

Alica P. Wolfe
Dept. of Math and Computer Science
Wesleyan University
Middletown, CT, 06459 USA
pwolfe@wesleyan.edu

Xiaolan Zhang
Dept. of Computer and Information Sciences
Fordham University
Bronx, NY, 10458 USA
xzhang@fordham.edu

Bing Wang
Dept. of Computer Science
University of Connecticut
Storrs, CT, USA 06269
bing@uconn.edu

Abstract

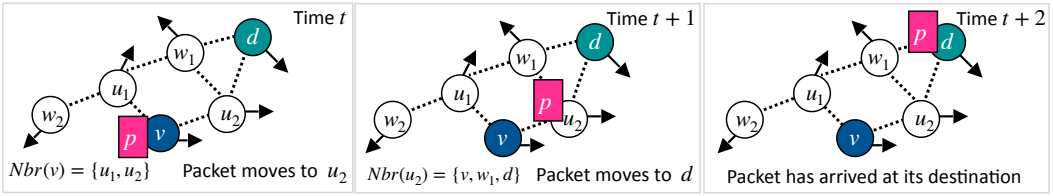
Mobile wireless networks present several challenges for any learning system, due to uncertain and variable device movement, a decentralized network architecture, and constraints on network resources. In this work, we use deep reinforcement learning (DRL) to learn a scalable and generalizable forwarding strategy for such networks. We make the following contributions: i) we use hierarchical RL to design DRL packet agents rather than device agents, to capture the packet forwarding decisions that are made over time and improve training efficiency; ii) we use relational features to ensure generalizability of the learned forwarding strategy to a wide range of network dynamics and enable offline training; and iii) we incorporate both forwarding goals and network resource considerations into packet decision-making by designing a weighted DRL reward function. Our results show that our DRL agent often achieves a similar delay per packet delivered as the optimal forwarding strategy and outperforms all other strategies including state-of-the-art strategies, even on scenarios on which the DRL agent was not trained.

1 Introduction

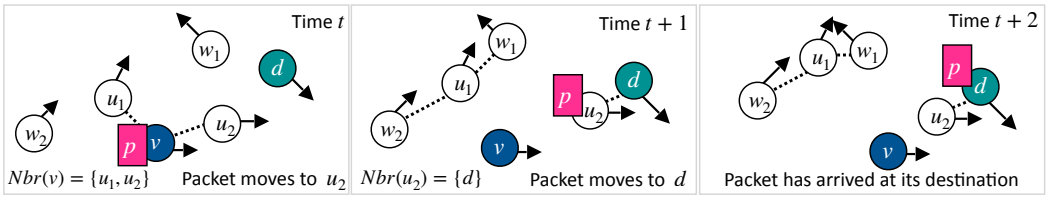
Mobile wireless networks have been used for a wide range of real-world applications, from vehicular safety [62, 55, 16] to animal tracking [25, 71] to environment monitoring [41, 2] to search-and-rescue [20, 21, 47] to military deployments [46, 44, 26] to the mobile Internet of Things [6]. These networks present several challenges, however, for learning systems. Devices are moving due to their association with, for instance, vehicles, UAVs, robots, animals, or people, which causes changes in the network connectivity. As a consequence, devices are often only able to communicate with each other during limited windows of time. Furthermore, it may be difficult to predict when opportunities for communication may occur as these depend on device movement. Depending on the specific network problem to address, there may also be competing goals to trade-off, such as delay vs. resource usage. Finally, the network architecture is decentralized, complicating training of any learning systems as exchange of training information is limited by the available communication opportunities.

In this work, we focus on the problem of packet forwarding in a mobile wireless network. Traditionally, forwarding strategies are hand-crafted to target specific kinds of network connectivity. For

Reinforcement Learning for Real Life (RL4RealLife) Workshop in the 36th Conference on Neural Information Processing Systems (NeurIPS 2022).



(a) *Routing*: v has a contemporaneous path to d, and so p is forwarded along the path.



(b) *DTN forwarding*. Only temporal paths exist from v to d, so devices independently choose p's next hop.

Figure 1: Packet p destined to device d travels from device v to device u_2 to finally arrive at its destination device d, but whether routing or DTN forwarding is used depends on whether a contemporaneous path is present.

instance, in some mobile networks, see Fig. 1(a), contemporaneous end-to-end paths may exist due to dense connectivity or slow device movement, and so routing [23, 43, 11, 42] is used to discover and forward packets over these paths. In other mobile networks, see Fig. 1(b), devices only occasionally meet due to sparse connectivity or fast device movement, and so contemporaneous paths rarely exist. Consequently, delay tolerant network (DTN) forwarding [56, 57, 63] is used and the best next hop for a packet is chosen based on criteria such as expected delay to meet the packet's destination.

In many real mobile wireless networks, however, a mix of connectivity is often found, see [34], motivating the need for an adaptive forwarding strategy. To learn such a strategy, we use deep reinforcement learning (DRL) [60], using deep neural networks (DNNs) [7] to approximate the RL policy. By choosing next hops locally at devices, both contemporaneous and temporal paths can be implicitly constructed by the DRL agent. Importantly, devices located in different parts of a network can independently run the same DRL agent as the agent will react appropriately to the local state at each device by using the parts of its policy relevant for that state. Consequently, a single policy can be applied to a state space that includes both well-connected and poorly connected parts of a network.

Most works on DRL-based forwarding in wireless networks focus on stationary devices; those that do consider device mobility either target specific kinds of network connectivity or have other limitations (see §2). In this work, we focus on designing a forwarding strategy for mobile wireless networks able to adapt to very different kinds of network connectivity. We make the following contributions.

i) **Packet-centric decision-making to simplify learning (§3.1).** We use *packet agents*, making packets, rather than devices, the decision-making agent, to accurately assign credit to those actions that affect packet success. Because packets can wait for several time steps before making a decision, we use hierarchical RL [13, 61, 4] with *fixed policy options* [61] to more efficiently back up from one decision point to another.

ii) **Relational features to ensure generalizability despite device mobility (§3.2)** We use *relational features* to represent the states and actions used by the DRL agent. Relational features model the relationship between network devices instead of describing a specific device, and so support generalizability to scenarios on which the DRL agent was not trained. This allows us to *train offline* thereby avoiding the need for decentralized communication. Relational features also allow us to structure the DNN representing the DRL forwarding policy to consider *one action at a time*, producing a single Q-value per state, action pair. The number of times the DNN is used to predict Q-values then corresponds to the number of actions available, allowing the trained DRL agent to handle varying numbers of neighbors (see Fig. 1) and so be network independent.

iii) **A weighted reward function to trade-off competing goals (§3.3).** To incorporate packet forwarding goals and network resource considerations into packet decision-making, we design a *weighted reward function* for the DRL agent. Using our reward function, more weight can be placed on higher priority forwarding considerations, such as to not waste resources unnecessarily.

iv) **Extensive evaluation (§4).** We evaluate our approach, which combines the above key design decisions, on three different networks sizes and six different transmission ranges spanning the continuum from disconnected to well-connected networks. Our results show that our DRL agent trained on only one of these scenarios generalizes well to the other scenarios. Specifically, our DRL agent achieves delay similar to an optimal strategy and outperforms all other strategies in terms of delay including the state-of-the-art seek-and-focus strategy [56], even on scenarios on which the DRL agent was not trained. While the optimal strategy requires global knowledge of future device movement, our DRL agent uses only locally obtained feature information.

2 Related Work

Many recent works on DRL based forwarding strategies focus on stationary wireless networks, see [68, 64, 67, 12, 38, 59, 69, 70, 9, 35]. Fewer works consider mobile wireless networks, and those that do often optimize for specific kinds of network connectivity, such as focusing primarily on vehicular networks [31, 51, 32, 33] or UAV networks [15, 54, 49, 52, 45]. Works that consider mobile networks more broadly have limitations: [48] uses per-destination Q-values; [14, 53] consider simplified state and action spaces; [18] focuses specifically on forwarding messages to network communities; [22, 26] focus on relatively limited network scenarios with a few fixed flows and up to 50 devices; [24] extends early work [8, 29] to tactical network environments but uses RL to estimate the shortest path to the destination rather than to select next hops. In comparison, we specifically focus on designing an adaptive forwarding strategy that can span the continuum from sparsely connected to well-connected mobile networks, and consider networks with 100 mobile devices. While our work builds off of ideas in [35], we consider the much harder network scenario of mobile devices, rather than the stationary devices considered in [35], and we design novel features to capture temporal and spatial network connectivity as well as propose a reward function to reflect competing network goals.

An important takeaway of our work is the need to tailor existing machine learning techniques to handle the decentralized communication and resource constraints of mobile wireless networks. For instance, while we have some features that represent actual relationships between devices as in relational learning [17, 28, 58], we primarily use relational techniques to treat devices as inter-changeable objects described by their attributes rather than their identity, to build a single model that works across all devices. While our aggregated neighborhood features are similar in spirit to graph neural networks [50, 27, 5], our features are simpler to compute and can more easily handle changing neighbors. While we use DRL to learn a policy, we handle actions differently than in a DQN [37] as the number of actions available at a device changes over time and varies across devices. Finally, while our weighted reward function could also be converted to use multi-objective reward techniques [65, 19], which find policies for a range of reward factor weightings, doing so would make training the DRL agent more computationally expensive.

3 Learning an Adaptive Forwarding Strategy

3.1 Packet Forwarding Using DRL

RL focuses on the design of intelligent agents: an RL agent interacts with its environment to learn a policy, i.e., which actions to take in different environmental states. The environment is modeled using a Markov decision process (MDP). An MDP comprises a set of states (S), a per state set of actions ($\mathcal{A}(s)$), a reward function, and a Markovian state transition function in which the probability of the next state $s' \in S$ depends only on the current state $s \in S$ and action $a \in \mathcal{A}(s)$. RL assumes that these state transition probabilities are not known, but that samples of transitions of the form (s, a, r, s') can be generated. From these samples, the algorithm learns a Q-value for each (s, a) pair. A Q-value estimates the expected future reward for an agent, when starting in state s and taking action a. Once learned, the optimal action in state s is the one with the highest Q-value. When the MDP has a small number of states and actions, an RL agent can learn a Q-value function using Q-learning (see [66]). When the state space is too large for exact computation of the Q-values, function approximation may be used to find approximate Q-values. Here, we use DNNs for function approximation, see Fig. 2. Each state s and action a is translated into a set of features via the functions $f_s(\cdot)$ and $f_a(\cdot)$. These features are input to the DNN, to produce as output an approximate Q-function $\hat{Q}(f_s(\cdot), f_a(\cdot))$.

In a mobile network, it is natural to think of devices as the DRL agents choosing next hops for packets, but doing so confuses the decision-making dependencies that are involved in forwarding a packet.

That is, the steps from a packet's source device to its destination device (or drop) are constructed by the behavior of all devices through which the packet passes, rather than only by the device at which the packet is currently located. Thus, we use *packet agents*, making packets, rather than devices, the decision-making agent, to more accurately assign credit to the actions that affect packet success. A packet's decision-making, however, can also involve multiple time steps, such as waiting in a queue for its turn to make a forwarding decision, or waiting for another device to come within transmission range. Thus, we use hierarchical RL [13, 61, 4], specifically *fixed policy options* [61], to allow reward signals to be backed up over multiple time steps in a single update. The use of options additionally supports faster learning while using less training data.

3.2 Mobile Network Features

Our goal is to be able to use the same learned forwarding strategy at different devices and in different mobile networks (e.g., dense or sparse networks with or without contemporaneous end-to-end paths) with unknown or unseen device mobility. To achieve this, we use *relational features* to represent the states and actions used by the DRL agent. Relational features, such as delay to destination, model the relationship between network devices instead of describing a specific device and so are not tied to a specific network topology. We propose five classes of relational features specifically tailored to model mobile wireless networks. These classes give a general framework for organizing the features needed for forwarding. The features we propose for each class, however, are not exhaustive but rather only the specific features our DeepRL agent uses in our simulations in §4. We expect many other features could be proposed for each class. We next describe the features we use, from the point of view of a packet p when choosing its next hop. We assume that p's destination is device d and that p is currently located at device v which has neighbors $u \in Nbr(v)$ when describing these features.

Packet features, $f_{packet}(p)$, are a function of information about packet p. Our DRL agent currently uses only one packet feature: packet p's time-to-live (TTL), which is used in real networks to prevent undeliverable packets from looping forever in the network. The TTL field is decremented when a packet is forwarded to another device. A packet is dropped when its TTL reaches 0.

Device features, $f_{device}(v, d)$, are a function of device v's information and destination device d's ID. Our DRL agent uses four device features computed at the current timestep t: i) device v's queue length, ii) device v's queue length considering only packets for destination d, iii) device v's node degree, and iv) device v's node density, computed as the fraction of neighbors that v has out of the N devices in the network. Even when a network has little congestion, queue length information can still be useful when choosing next hops. For instance, next hop devices that do not have any packets in the same flow as p may be preferred to better distribute traffic. Our DRL agent additionally keeps track of two device features, the x and y location coordinates of device v, which are only used to compute the Euclidean distance, a path feature described later in this section. While we assume every device knows its own location, locations for other devices are obtained only when two devices meet and exchange features. Consequently, location (and thus distance) features can be out-of-date.

Path features, $f_{path}(v, d)$, describe the time-varying path from a device v to a destination device d. Unlike the other features discussed so far, path features may use not just current device information but also historical information. Consequently, these features may have some associated uncertainty. Our DRL agent currently uses only one path feature: the Euclidean distance from device v to destination d. This is calculated using v's current x and y location coordinates, and device v's recorded (and possibly out-of-date) location coordinates of destination d (see description of device features).

Neighborhood features, $f_{nbrhood}(v, d)$, are computed over the current neighbors $Nbr(v)$ of a device v. Our DRL agent uses the following neighborhood features: for each device and path feature, $f_i \in f_{device}(v, d) \cup f_{path}(v, d)$, we compute the minimum, maximum, and average over device v's current neighborhood, $Nbr(v)$. These features compress the information obtained from a variable number of neighbors into a fixed size vector to input to the DNN in Fig. 2.

Context features, $f_{context}(p, u)$, provide context for other features. For a packet p at a device v considering a next hop $u \in Nbr(v) \cup v$, our DRL agent uses context features that indicate whether p has recently visited u or not. Each packet p stores in its packet header the last $N_{history}$ device IDs that it visited, where $N_{history}$ is a predetermined constant. Let $\mathcal{H}(p, i)$ be the ID of the device that packet p visited i hops ago, for $0 \leq i < N_{history}$. When $i = 0$, then $\mathcal{H}(p, 0)$ is the device at which p is currently located. To make the context features relational, rather than use device IDs, we use a

Table 1: For each feature f_i, normalization is done using $(f_i + 1)/(D + 1)$, where 1s are added to avoid zero values for features. Neighborhood features are not normalized as they are a function of other normalized features; context features also are not normalized as they are Boolean valued.

Packet Features	D	**Device Features**	D	**Path Features**	D
Packet TTL	300	Queue length	30	x-coordinate location	500m
		Per-destination queue length	30	y-coordinate location	500m
		Node degree	10	Euclidean distance	2
		Node density	N		

sequence of Boolean features, b_i, defined as:

$$b_i = \begin{cases} 1, & \text{if } u = \mathcal{H}(p, i), \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The use of packet history reduces unnecessary packet transmissions. For instance, even if a possible next hop device u has promising features for reaching the destination, if packet p recently visited u, then u may be a less good next hop than it seems based solely on u's other feature values.

Feature estimation. A device discovers its neighbors through the use of "heartbeat" control messages. All features are then estimated using local exchange of information between neighboring devices. Device v obtains the following information from each neighbor u: i) the features $f_{device}(u, d) \cup f_{path}(u, d) \cup f_{nbrhood}(u, d)$, ii) u's current x and y location coordinates, timestamped with u's current clock, and iii) the x and y location coordinates for every other device w, which u has either recorded directly from w or received indirectly from another device, along with the recording's timestamp, i.e., the time on w's clock of when the coordinates were recorded. Device v then uses ii) and iii) to update its recording of the x and y location coordinates for every other device, overwriting older recordings with more recent recordings for a device w, comparing the timestamps associated with the recordings. Because these timestamp comparisons always compare timestamps received from the same device w, no clock synchronization is needed.

Feature normalization. Mobility makes normalization challenging as the ranges of the raw feature values may be very different in different mobile networks. To address this, we make the normalization a function of network properties when possible, such as the (approximate) number of devices in the network or the (approximate) size of the area in which devices are moving, see Table 1. In this way, the normalization can better adapt to new network environments.

3.3 MDP Formulation

Let $Nbr(v)$ be the current neighbors of device v. Each individual packet agent p currently located at device v can choose between moving to one of these neighbors or staying at device v. Packet p's actions therefore correspond to the set $Nbr(v) \cup \{v\}$. We next define the states, actions, and reward function for our packet-centric DRL agent from p's point of view.

States. The state features that p uses to make a decision are a function of features derived from p, d, and v: $f_s(v, p, d) = f_{packet}(p) \cup f_{device}(v, d) \cup f_{path}(v, d) \cup f_{nbrhood}(v, d)$.

Actions. The action features for each action u in p's action set $Nbr(v) \cup \{v\}$ are defined by $f_a(u, p, d) = f_{device}(u, d) \cup f_{path}(u, d) \cup f_{nbrhood}(u, d) \cup f_{context}(p, u)$. This action description reuses many of the same features in the state description, but is defined in terms of a device $u \in Nbr(v) \cup \{v\}$ rather than just p's current location at device v, and includes the context features.

Rewards. Forwarding strategies for mobile wireless networks must trade-off competing goals for packet delivery, such as minimizing delay until delivery while also minimizing resource usage like energy and link bandwidth. As a packet travels to its destination, it must also assign credit or blame to the devices it passes through, depending on the success or failure of the packet to reach its destination. To incorporate these considerations into packet decision-making, we design a *weighted reward function*. We define separate rewards for the action of a packet choosing to stay at its current device, r_{stay}, vs. moving to a neighbor device that is not the destination, $r_{transmit}$. The ratio of r_{stay} to $r_{transmit}$ determines the trade-off the forwarding strategy learns between minimizing packet delivery delay vs. number of transmissions. For mobile networks, where forwarding loops can easily arise, explicitly penalizing transmissions is important. For instance, by setting $r_{stay} = r_{transmit}$,

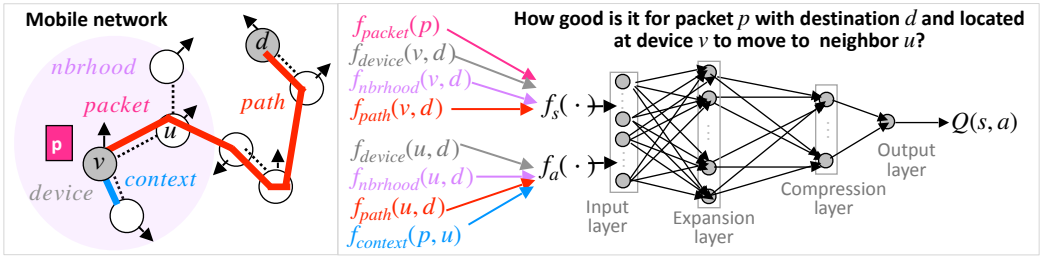


Figure 2: Packet p at device v makes a forwarding decision by activating the DNN at v once for each action a available in p's current state s. The DNN inputs are the features $f_s(\cdot)$ and $f_a(\cdot)$, which describe the state from p's point of view and the action under consideration. Packet p chooses the action with the best Q-value.

the DRL agent would minimize delay, but ignore the number of transmissions made. We define two other rewards, for actions that lead to a packet being delivered to its destination, $r_{delivery}$, or dropped, $r_{drop} = r_{transmit}/(1-\gamma)$, where $\gamma \in [0, 1]$ is the RL discount factor. The drop reward is equivalent to receiving a reward of $r_{transmit}$ for infinite timesteps. Our reward settings are given in Table 2.

Decision-making. The DNN architecture we use to approximate the Q-value function is shown in Fig. 2. Our DNN has four layers: input, expansion, compression, and output. Let $F = |f_s(\cdot)| + |f_a(\cdot)|$ be the number of input features and thus the size of the input layer. The expansion layer has $10F$ neurons and the compression layer has $F/2$ neurons. The DNN then outputs a Q-value for each state, action pair, represented by the feature vectors $f_s(\cdot)$ and $f_a(\cdot)$. Each device has a copy of the same trained DNN that encodes the forwarding strategy, which allows decision-making to be done independently at each device. Fig. 2 also overviews how our DRL packet agent uses the trained DNN to make a forwarding decision. To obtain the features $f_s(\cdot)$ and $f_a(\cdot)$ to input into the neural network in Fig. 2, the DRL agent computes the following features for packet p with destination d currently at device v: $f_{packet}(p)$, $f_{device}(v, d)$, and $f_{path}(v, d)$ using local information at device v; $f_{nbrhood}(v, d)$, $f_{device}(u, d)$, $f_{path}(u, d)$, and $f_{nbrhood}(u, d)$ using the features received by v from $u \in Nbr(v)$; and finally, $f_{context}(p, u)$ using only u's ID in addition to the information carried in packet p's header fields. The number of times the DNN is used to predict Q-values corresponds to the number of actions available to the packet. Using the DNN to separately make predictions for each action, rather than making predictions for all actions at once, allows the DRL agent to handle varying numbers of actions, and, correspondingly, varying numbers of neighbors. During training, ϵ-greedy action selection is used, with ϵ set as in Table 2.

Training. In a mobile wireless networks, devices can only exchange information (such as whether a particular packet reached its destination or was dropped) using distributed communication. Consequently, it is typically not feasible to gather the information needed for training online due to limited network bandwidth. Instead, we use *offline training*. During training, for each packet decision, regardless of the packet's device location, we record the state features, the action features for each action considered, the action selected, and the received reward in one file. Because our relational features are device and network independent, we are able to train a single DRL agent using this file using experience replay. The learned forwarding strategy is then copied to each device and used *independently* at each device by packets to choose next hops. This approach to training is particularly important as even a single mobile network scenario includes network connectivity changing over time and space, making it essential to be able to include such diversity during training.

4 Simulation Results

Our simulations are done using a custom discrete-time packet-level network simulator that we have implemented in Python3. The DRL agents are trained and all strategies are tested using this simulator. We use Keras v.2.5.0 [10] and Tensorflow v.2.50 [1] to implement the DNN. Table 2 gives our simulation parameters.

4.1 Methodology

Device mobility. We use BonnMotion [3] to generate mobility traces for devices moving under the steady-state random waypoint mobility model [39, 40, 30]. For training, we use traces of $N_{train} = 25$ devices moving in a 500m $\times$ 500m area at an average speed of 3 m/s with a speed delta of 2 m/s and

Table 2: Simulation parameters.

Symbol	*Meaning*	*Value*
N_{train}; N	# of devices during training; testing	25; 25, 64, 100
X_{train}; X_{test}	Transmission range for training; testing	50m; 30m to 80m
ϵ_{train}, ϵ_{test}	RL exploration rate for training; testing	0.1; 0
γ	RL discount rate	0.99
$r_{delivery}$; r_{stay}; $r_{transmit}$	RL reward: delivery; stay; transmit	0; -1; -1, -2, -10
r_{drop}	RL reward: drop	$r_{transmit}/(1-\gamma)$
$N_{history}$	Length of device visit history	0, 5
T_{train}; T_{test}	# of timesteps for training; testing	90,000; 100,000
T_{model}	# of training timesteps used by testing model	60,000
$T_{cooldown}$	# of timesteps at simulation end with no traffic	10,000
T_{round}	# of timesteps per round	1000
-	DNN training dropout rate	0.2

a transmission range of X_{train}=50m. For testing, we generate separate traces for $N = 25, 64, 100$ devices using the same size area and range of speeds, but vary the transmission range, X_{test} from 30m to 80m to obtain both poorly connected and very well connected mobile scenarios, see Appendix A.

Network traffic. We model flow arrivals using a Poisson distribution with parameter $\lambda_F = .001N/25$, scaling the number of flows as a function of the number of devices N in the network. We model flow durations using an exponential distribution with parameter $\lambda_D = 5000$ and packet arrivals on flows using a Poisson distribution with parameter λ_P = 0.01. A simulation run starts with $\lambda_F \lambda_D$ initial flows. Each device has a queue with a maximum size of $B = 200$ packets, beyond which additional packets are dropped. A packet's TTL field is initialized to $TTL_{train} = 300$ during training and $TTL_{test} = 3000$ during testing. We use a large testing TTL to ensure no packet is dropped.

MAC protocol. At each timestep, every device in the network is given an opportunity to transmit up to $k = 100$ packets in its queue. Given our network traffic settings, this k is sufficient for a device to transmit all of its packets, avoiding the need for device decision-making.

4.2 Forwarding Strategies Compared

In our simulations, we compare the performance of five forwarding strategies, including a delay minimizing strategy (*optimal*) and a transmission minimizing strategy (*direct transmission*) to give bounds on the performance of the DRL agent. We also compare with the Utility and Seek-and-Focus strategies from [56] as state-of-the-art strategies.

Optimal forwarding uses complete information about current and future network connectivity to calculate the minimal hop path that achieves the minimal delivery delay for each packet. To find these forwarding paths, we make use of epidemic routing. Epidemic routing creates many copies for a packet and distributes them to the network. For those packet copies that reach the destination with the minimum latency, we further find the packet copy that reached the destination with the minimum number of hops. This strategy minimizes delay, while maintaining a good trade-off in terms of network resources, but is not practical to implement in real networks.

Direct transmission forwarding only forwards a packet one hop, directly from the source to the destination. This is optimal when the goal is to minimize the number of transmissions per packet.

Utility-based forwarding [56] maintains a timer at each device v for every other device d in the network, denoted as $\tau_v(d)$, which is the time elapsed since device v last met device d. We implemented the timer transitivity as defined in [56]. For many mobility models, a smaller timer value on average implies a shorter distance to the device, so the timer evaluates the "utility" of a device in delivering a packet to another device. A device v chooses a neighbor u as the next hop for a packet if u has the smallest timer to the packet's destination d (among all neighbors of v) and u's timer to d is smaller than v's timer to d by more than the utility threshold, U_{th}. If no such neighbor exists, device v does not forward the packet. We optimize U_{th} for $N_{train} = 25$ and $X_{train} = 50$m, which are the same settings on which the DRL agent used in testing was trained. The details of utility-based forwarding and the parameter values we use can be found in Appendix B.

Seek-and-focus forwarding [56] combines the utility-based strategy with a random forwarding strategy. If the packet is perceived to be far from its destination, the packet is in seek phase (random

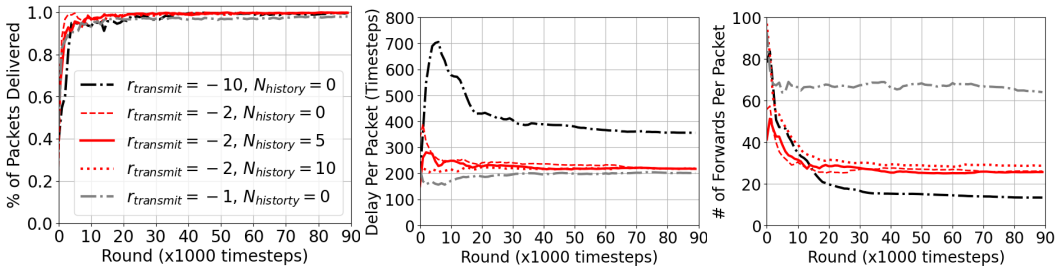


Figure 3: DRL agent training performance on $N_{train} = 25$ devices and transmission range $X_{train} = 50m$. Varying $r_{transmit}$ varies the trade-off between delay and number of forwards per packet.

forwarding); otherwise, the packet is in the focus phase, and utility-based forwarding is used to choose the next hop. Two additional parameters are used to avoid a packet stuck in a phase for a long time: T_{focus} controls the maximum duration to stay in focus phase, and T_{seek} controls the maximum duration to stay in re-seek phase before going to seek phase. All together, six parameters are used in seek-and-focus forwarding. Again, we optimize these parameters for $N_{train} = 25$ and $X_{train} = 50m$. Further details and the parameter values we use are given in Appendix B.

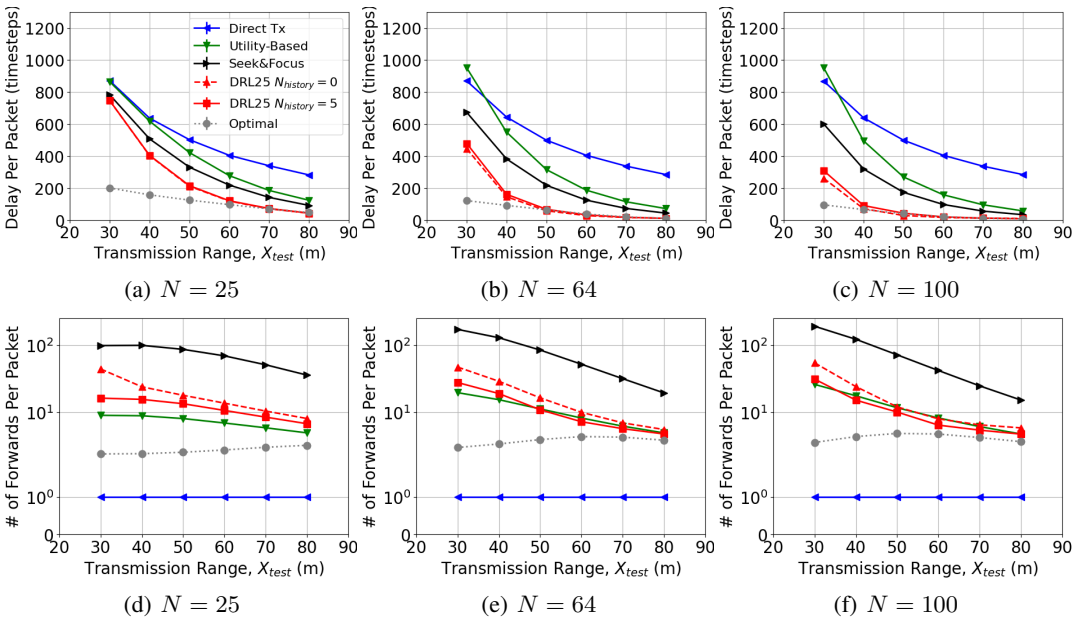
DRL25 forwarding uses a DRL agent to make forwarding decisions, trained on a network with $N_{train} = 25$ devices. Fig. 3 plots training performance, showing the DRL25 strategies converging. Each training simulation is run for T_{train} timesteps, where each timestep corresponds to one second. Training is done every T_{round} timesteps using all data received up to that timestep but using a randomly initialized DNN model. At each timestep, the cumulative performance over all packets delivered up to the timestep is shown. As shown in Fig. 3, varying the transmission reward, $r_{transmit}$, varies the learned trade-off between delay and transmissions. When $r_{transmit} = -1$, there is no penalty for transmission as a packet receives the same reward for transmitting as for staying at a device, since $r_{stay} = -1$. Correspondingly, we see that when $r_{transmit} = -1$, delay per packet delivered is the lowest of the strategies, but the number of forwards per packet is the highest. Conversely, for $r_{transmit} = -10$, there is a large penalty for making a transmission, and so, while delay per packet delivered is the highest for this strategy, the number of forwards per packet is now the lowest. In our testing results in Fig. 4, the DRL25 agents used were trained with $r_{transmit} = -2$ as this provides a good trade-off between delay and number of transmissions.

4.3 Evaluation Results

Overall results. In Fig. 4, we plot the the testing performance of the DRL25 agents. Both agents were trained on a mobile network scenario with $N_{train} = 25$ devices and a transmission range of $X_{train} = 50$m, with a transmission reward of $r_{transmit} = -2$. The two DRL25 agents differ only in their choices of $N_{history}$. The testing scenarios include 25, 64 or 100 devices (the largest network sizes we investigated) with transmission ranges varying from 30m to 80m, leading to various levels of connectivity levels (see Appendix A for more details). Most of the testing scenarios differ from the training scenario in terms of the number of devices and/or the transmission range. Fig. 4 shows the performance of the DRL25 agents on the various testing scenarios. We see that our DRL25 agents are able to generalize well from the training scenario to the various testing scenarios, including those on which they were not trained. We also observe that our DRL25 agents often achieve packet delivery delays similar to the optimal strategy, and outperform all other strategies in terms of delay.

Impact of network connectivity. As the network topology becomes more well connected (i.e., due to increasing N or X_{test}), the DRL25 agents start to have delay per packet delivered similar to that of the optimal strategy, with not too many more forwards per packet delivered. As the network topology becomes more sparse (i.e., due to decreasing N or X_{test}), the DRL25 agents start to have delay that is approaching the delay of the other strategies. The DRL25 agents have their highest delay per packet delivered for the $N = 25$ and X_{test} = 30m scenario which is the most disconnected scenario considered in our simulations (see Fig. 5). Due to the few neighbors and relatively long inter-meeting times between pairs of devices for this scenario, including temporal neighborhood features would be beneficial, as would be considering predicted future neighbors when choosing next hop actions.

Impact of context features. While $r_{transmit}$ penalizes packet transmissions, the context history features themselves do not directly penalize transmissions. Instead, the context history features



(a) $N = 25$ (b) $N = 64$ (c) $N = 100$

(d) $N = 25$ (e) $N = 64$ (f) $N = 100$

Figure 4: Testing performance, varying the number of devices N and the transmission range X_{test}. The DRL strategies were each trained on a $N_{train} = 25$ and $X_{train} = 50$m network scenario, for $r_{transmit} = -2$ and two different amounts of context history. Each point is the average of 50 simulation runs; 95% confidence intervals are shown but are very small. We observed all packets delivered by the end of each run for all strategies.

augment the state space to add context when actions are taken and ensure that actions that lead to unnecessary looping can be more easily identified. Fig. 4 shows that the DRL25 agent with no context history ($N_{history} = 0$) has a consistently higher number of forwards per packet delivered than the DRL25 agent with context history ($N_{history} = 5$) despite having a similar delay per packet.

5 Conclusions and Future Work

In this work, we have shown that it is possible to use DRL to learn a scalable and generalizable forwarding strategy for mobile wireless networks. We leverage three key ideas: i) packet agents, ii) relational features, and iii) a weighted reward function. Our results show that our DRL agent generalizes well to scenarios on which it was not trained, often achieving delay similar to the optimal strategy and outperforming all other strategies in terms of delay including the state-of-the-art seek-and-focus strategy [56]. The key ideas of our approach are generally applicable to other decision-making tasks in mobile wireless networks.

There are a number of research directions we would like to explore in future work. While we considered many features in this work, we would like to use feature ablation to understand which features most impact forwarding performance. We expect that as the kinds of mobility the DRL agent sees become more diverse, more features will also be needed to characterize the key differences in mobility and enable the DRL agent to generalize to a wide variety of mobile networks. More generally, we would like to expand our feature classes to include other features including alternatives to the Euclidean distance feature. We would also like to explore device decision-making to complement our packet agents. For instances, devices could learn which subsets of packets should get to make a forwarding decision when a transmission opportunity arises. Finally, we would like to update our reward function to incorporate additional network considerations such as fairness.

Acknowledgements

This material is based upon work supported by the National Science Foundation (NSF) under award #2154190. Results presented in this paper were obtained in part using CloudBank, which is partially supported by the NSF under award #2154190. The authors also acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing HPC and consultation resources that have contributed to the research results reported within this paper.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Cristina Albaladejo, Pedro Sánchez, Andrés Iborra, Fulgencio Soto, Juan A. López, and Roque Torres. Wireless sensor networks for oceanographic monitoring: A systematic review. *Sensors*, 10(7), 2010.

[3] Nils Aschenbruck, Raphael Ernst, Elmar Gerhards-Padilla, and Matthias Schwamborn. Bonnmotion: a mobility scenario generation and analysis tool. In *Proceedings of the 3rd international ICST conference on simulation tools and techniques*, pages 1–10, 2010.

[4] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1):41–77, 2003.

[5] Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andy Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Jayne Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv*, 2018.

[6] Oladayo Bello and Sherali Zeadally. Intelligent device-to-device communication in the internet of things. *IEEE Systems Journal*, 10(3):1172–1182, 2014.

[7] Yoshua Bengio. *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[8] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in neural information processing systems*, pages 671–678, 1994.

[9] Long Chen, Bin Hu, Zhi-Hong Guan, Lian Zhao, and Xuemin Shen. Multiagent meta-reinforcement learning for adaptive multipath routing optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[10] Francois Chollet et al. Keras, 2015.

[11] Thomas Clausen, Philippe Jacquet, Cédric Adjih, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot. Optimized link state routing protocol (olsr). 2003.

[12] Valerio Di Valerio, Francesco Lo Presti, Chiara Petrioli, Luigi Picari, Daniele Spaccini, and Stefano Basagni. Carma: Channel-aware reinforcement learning-based multi-path adaptive routing for underwater wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 37(11):2634–2647, 2019.

[13] Thomas G Dietterich. The MAXQ Method for Hierarchical Reinforcement Learning. In *ICML*, volume 98, pages 118–126. Citeseer, 1998.

[14] Ahmed Elwhishi, Pin-Han Ho, Kshirasagar Naik, and Basem Shihada. Arbr: Adaptive reinforcement-based routing for dtn. In *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 376–385. IEEE, 2010.

[15] Ming Feng, Lijun Qian, and Hao Xu. Multi-robot enhanced manet intelligent routing at uncertain and vulnerable tactical edge. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pages 1–9. IEEE, 2018.

[16] M. Gerla, E. Lee, G. Pau, and U. Lee. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In *Proc. IEEE World Forum on Internet of Things*, 2014.

[17] Lise Getoor and Ben Taskar. Introduction to statistical relational learning, vol. 1. *MIT press Cambridge*, 10:1296231, 2007.

[18] Chenchen Han, Haipeng Yao, Tianle Mai, Ni Zhang, and Mohsen Guizani. Qmix aided routing in social-based delay-tolerant networks. *IEEE Transactions on Vehicular Technology*, 71(2):1952–1963, 2021.

[19] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):1–59, 2022.

[20] J.-H. Huang, S. Amjad, and S. Mishra. CenWits: a sensor-based loosely coupled search and rescue system using witnesses. 2005.

[21] Lun Jiang, Jyh-How Huang, Ankur Kamthe, Tao Liu, Ian Freeman, John Ledbetter, Shivakant Mishra, Richard Han, and Alberto Cerpa. SenSearch: GPS and witness assisted tracking for delay tolerant sensor networks. In *Proc. of International Conference on Ad-hoc and Wireless Networks (ADHOC-NOW)*, 2009.

[22] LIU Jianmin, WANG Qi, HE Chentao, and XU Yongjun. Ardeep: Adaptive and reliable routing protocol for mobile robotic networks with deep reinforcement learning. In *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, pages 465–468. IEEE, 2020.

[23] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[24] Matthew Johnston, Claudiu Danilov, and Kevin Larson. A reinforcement learning approach to adaptive redundancy for routing in tactical networks. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pages 267–272. IEEE, 2018.

[25] P. Juang, H. Oki, Y. Wang, Margaret Martonosi, LiShiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proc. of International Conference on Architectural Support for Programming Languages and Operating Systems*, 2002.

[26] Saeed Kaviani, Bo Ryu, Ejaz Ahmed, Kevin A Larson, Anh Le, Alex Yahja, and Jae H Kim. Robust and Scalable Routing with Multi-Agent Deep Reinforcement Learning for MANETs. *arXiv preprint arXiv:2101.03273*, 2021.

[27] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Representation Learning (ICLR)*, 2017.

[28] Daphne Koller, Nir Friedman, Sašo Džeroski, Charles Sutton, Andrew McCallum, Avi Pfeffer, Pieter Abbeel, Ming-Fai Wong, Chris Meek, Jennifer Neville, et al. *Introduction to statistical relational learning*. MIT press, 2007.

[29] S. Kumar and R. Miikkulainen. Confidence-based Q-routing: an on-line adaptive network routing algorithm. In *Proc. of Artificial Neural Networks in Engineering*, 1998.

[30] J-Y Le Boudec and Milan Vojnovic. Perfect simulation and stationarity of a class of mobility models. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 4, pages 2743–2754. IEEE, 2005.

[31] Fan Li, Xiaoyu Song, Huijie Chen, Xin Li, and Yu Wang. Hierarchical routing for vehicular ad hoc networks via reinforcement learning. *IEEE Transactions on Vehicular Technology*, 68(2):1852–1865, 2018.

[32] Arbelo Lolai, Xingfu Wang, Ammar Hawbani, Fayaz Ali Dharejo, Taiyaba Qureshi, Muhammad Umar Farooq, Muhammad Mujahid, and Abdul Hafeez Babar. Reinforcement learning based on routing with infrastructure nodes for data dissemination in vehicular networks (rrin). *Wireless Networks*, pages 1–16, 2022.

[33] Long Luo, Li Sheng, Hongfang Yu, and Gang Sun. Intersection-based v2x routing via reinforcement learning in vehicular ad hoc networks. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[34] Victoria Manfredi, Mark Crovella, and Jim Kurose. Understanding stateful vs stateless communication strategies for ad hoc networks. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 313–324, 2011.

[35] Victoria Manfredi, Alicia Wolfe, Bing Wang, and Xiaolan Zhang. Relational deep reinforcement learning for routing in wireless networks. In *Proceedings of the 22nd IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2021.

[36] Victoria Manfredi, Alicia P Wolfe, Xiaolan Zhang, and Bing Wang. Learning to route in mobile wireless networks. *arXiv preprint arXiv:2207.11386*, 2022.

[37] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[38] D. Mukhutdinov, A. Filchenkov, A. Shalyto, and V. Vyatkin. Multi-agent deep learning for simultaneous optimization for time and energy in distributed routing system. *Future Generation Computer Systems*, 94:587–600, 2019.

[39] William Navidi and Tracy Camp. Stationary distributions for the random waypoint mobility model. *IEEE transactions on Mobile Computing*, 3(1):99–108, 2004.

[40] William Navidi, Tracy Camp, and Nick Bauer. Improving the accuracy of random waypoint simulations through steady-state initialization. In *Proceedings of the 15th International Conference on Modeling and Simulation*, pages 319–326, 2004.

[41] Luiz F. P. Oliveira, António P. Moreira, and Manuel F. Silva. Advances in forest robotics: A state-of-the-art survey. *Robotics*, 10(2), 2021.

[42] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing for mobile computers. In *ACM SIGCOMM*, 1994.

[43] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Rfc3561: Ad hoc on-demand distance vector (aodv) routing, 2003.

[44] Konstantinos Poularakis, Qiaofeng Qin, Erich M Nahum, Miguel Rio, and Leandros Tassiulas. Flexible sdn control in tactical ad hoc networks. *Ad Hoc Networks*, 85:71–80, 2019.

[45] Xiulin Qiu, Lei Xu, Ping Wang, Yuwang Yang, and Zhenqiang Liao. A data-driven packet routing algorithm for an unmanned aerial vehicle swarm: A multi-agent reinforcement learning approach. *IEEE Wireless Communications Letters*, 11(10):2160–2164, 2022.

[46] R Ramanathan, R Allan, P Basu, J Feinberg, G Jakllari, V Kawadia, S Loos, J Redi, C Santivanez, and J Freebersyser. Scalability of mobile ad hoc networks: Theory vs practice. In *MILCOM*, 2010.

[47] William H Robinson and Adrian P Lauf. Resilient and efficient manet aerial communications for search and rescue applications. In *2013 International conference on computing, networking and communications (ICNC)*, pages 845–849. IEEE, 2013.

[48] Vitor G Rolla and Marilia Curado. A reinforcement learning-based routing for delay tolerant networks. *Engineering Applications of Artificial Intelligence*, 26(10):2243–2250, 2013.

[49] Arnau Rovira-Sugranes, Fatemeh Afghah, Junsuo Qu, and Abolfazl Razi. Fully-echoed q-routing with simulated annealing inference for flying adhoc networks. *IEEE Transactions on Network Science and Engineering*, 8(3):2223–2234, 2021.

[50] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[51] Cedrik Schüler, Manuel Patchou, Benjamin Sliwa, and Christian Wietfeld. Robust machine learning-enabled routing for highly mobile vehicular networks with parrot in ns-3. In *Proceedings of the Workshop on ns-3*, pages 88–94, 2021.

[52] Cedrik Schüler, Benjamin Sliwa, and Christian Wietfeld. Towards machine learning-enabled context adaption for reliable aerial mesh routing. In *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, pages 1–7. IEEE, 2021.

[53] Deepak Kumar Sharma, Joel JPC Rodrigues, Vidushi Vashishth, Anirudh Khanna, and Anshuman Chhabra. Rlproph: a dynamic programming based reinforcement learning approach for optimal routing in opportunistic iot networks. *Wireless Networks*, 26(6):4319–4338, 2020.

[54] Benjamin Sliwa, Cedrik Schüler, Manuel Patchou, and Christian Wietfeld. Parrot: Predictive ad-hoc routing fueled by reinforcement learning and trajectory knowledge. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–7. IEEE, 2021.

[55] Christoph Sommer and Falko Dressler. *Vehicular Networking*. Cambridge University Press, 2014.

[56] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S Raghavendra. Single-copy routing in intermittently connected mobile networks. In *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.*, pages 235–244. IEEE, 2004.

[57] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S Raghavendra. Efficient routing in intermittently connected mobile networks: The multiple-copy case. *IEEE/ACM transactions on networking*, 16(1):77–90, 2008.

[58] Jan Struyf and Hendrik Blockeel. Relational learning. 2010.

[59] J. Suarez-Varela, A. Mestres, J. Yu, L. Kuang, H. Feng, P. Barlet-Ros, and A. Cabellos-Aparicio. Feature engineering for deep reinforcement learning based routing. In *Proc. of IEEE International Conference on Communications (ICC)*, pages 1–6, 2019.

[60] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[61] Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[62] Chai K. Toh. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice Hall, 2001.

[63] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. *Duke Univ., Durham, NC, Tech. Report, CS-200006*, April 2000.

[64] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar. Learning to route with deep RL. In *NIPS Deep Reinforcement Learning Symposium*, 2017.

[65] Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

[66] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[67] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang. Experience-driven networking: A deep reinforcement learning based approach. In *IEEE INFOCOM*, pages 1871–1879, 2018.

[68] Dayong Ye, Minjie Zhang, and Yun Yang. A multi-agent framework for packet routing in wireless sensor networks. *Sensors*, 15(5):10026–10047, 2015.

[69] Xinyu You, Xuanjie Li, Yuedong Xu, Hui Feng, and Jin Zhao. Toward packet routing with fully-distributed multi-agent deep reinforcement learning. In *2019 International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, pages 1–8. IEEE, 2019.

[70] Xinyu You, Xuanjie Li, Yuedong Xu, Hui Feng, Jin Zhao, and Huaicheng Yan. Toward packet routing with fully distributed multiagent deep reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[71] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware design experiences in ZebraNet. In *Proc. of ACM SenSys*, 2004.

Appendix

A Dynamics of the Mobility Traces Used in Simulations

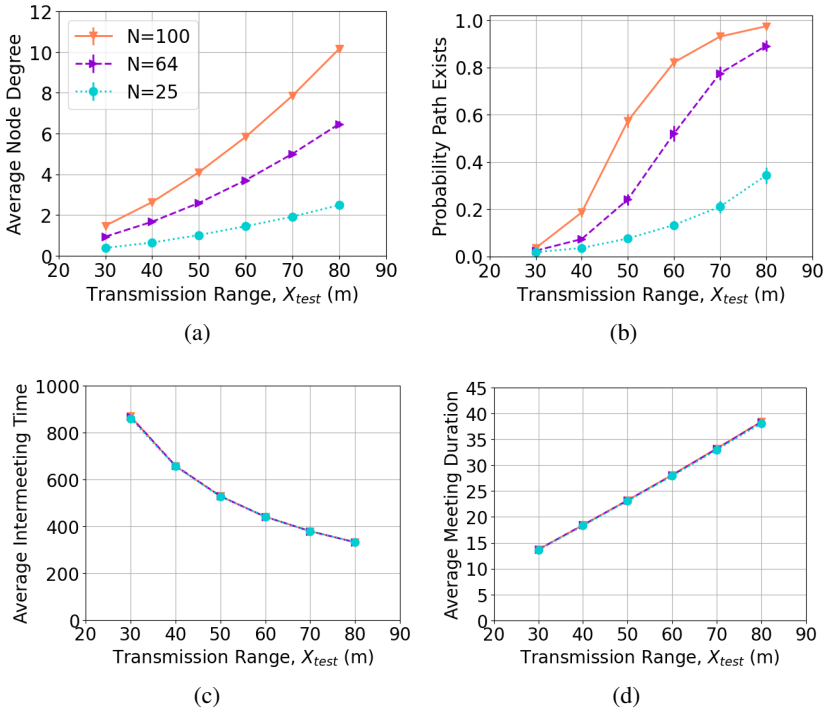


Figure 5: Quantifying the connectivity of the network scenarios considered in our simulation experiments.

Fig. 5 quantifies the connectivity of the network scenarios we consider in our simulations. In Figs. 5(a) and (b) we show that the different numbers of devices and transmission ranges we consider in our simulation experiments encompass the continuum from disconnected to well-connected networks. In Figs. 5(c) and (d), we show the inter-meeting time $T_{v,d}$ and meeting duration $M_{v,d}$ of §6: these metrics are computed online between each possible pair of devices, and so are independent of the number of devices. For additional details about our simulations and extensive simulation results, please see [36].

B Details of Utility and Seek-and-Focus Strategies

Utility-based forwarding [56] maintains a timer at each device v for each device d in the network, denoted as $\tau_v(d)$, which is the time elapsed since device v last met device d, as illustrated in Fig 6. We implemented the timer transitivity as defined in [56]: when two devices, u and v encounter each

Table 3: Parameters settings.

Symbol	*Meaning*	*Utility*	*Seek-and-Focus*
U_{th}	Utility threshold	10	100
U_f	Focus threshold	-	20
$prob$	Random forwarding probability	-	0.5
$time\_until\_decoupling$	Time before sending a packet back to a device	-	10
T_{focus}	Max duration to stay in focus phase	-	10
T_{seek}	Max duration to stay in re-seek phase	-	50

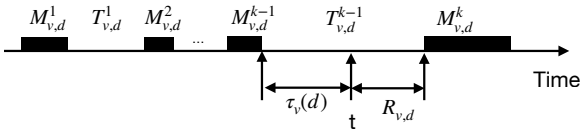


Figure 6: Contacts between devices v and d, where $\tau_v(d)$ is the raw timer (without transitivity) referred to in the utility-based and seek-and-focus forwarding strategies [56] for delay tolerant networks.

other, if $\tau_u(d) < \tau_v(d) - t(d_{u,v})$, where $t(d_{u,v})$ is the expected time for a device to move a distance of $d_{u,v}$, then $\tau_v(d)$ is set to $\tau_v(d) = \tau_u(d) + t(d_{u,v})$. As for many mobility models, a smaller timer value on average implies a smaller distance to the device, the timer evaluates the "utility" of a device in delivering a packet to another device. A device v chooses the next hop for a packet as follows: v first determines which neighboring device, u, has the smallest timer to the packet's destination, d. If $\tau_v(d) > \tau_u(d) + U_{th}$, i.e., the timer of v to destination d is larger than the timer of u to d by more than the utility threshold, U_{th}, the packet is forwarded to device u; otherwise, the packet is not forwarded. We optimize U_{th} for $N_{train} = 25$ and $X_{train} = 50$m, which are the same settings on which the DRL agent used in testing was trained; the parameter value we use is shown in Table 3.

Seek-and-focus forwarding [56] combines the utility-based strategy (*focus phase*) and random forwarding (*seek phase*). If the smallest timer (among all neighboring devices) to the destination is larger than the focus threshold U_f, the packet is in seek phase, forwarded to random neighbor with probability $prob$. Otherwise, the packet is in the focus phase, and the carrier of the packet performs utility-based forwarding with utility threshold U_{th}. In addition to U_f, $prob$, and U_{th}, Seek-and-focus has three more parameters: the *time_until_decoupling* which controls the amount of time a device is not allowed to forward a packet back to a device it received the packet from, T_{focus} which controls the maximum duration to stay in focus phase before going to re-seek phase (random forwarding of the packet to get out of a local minimum), and T_{seek} which controls the maximum duration to stay in re-seek phase until going to seek phase (random forwarding of the packet until reaching a device with timer smaller than U_f). We optimize these parameters for $N_{train} = 25$ and $X_{train} = 50$m, which are the same settings on which the DRL agent used in testing was trained; the parameter values we use are shown in Table. 3.