Q^2-Routing : A Qos-aware Q-Routing algorithm for Wireless Ad Hoc Networks

Thomas Hendriks
University of Antwerp
thomas.hendriks@student.uantwerpen.be

Miguel Camelo
IDLab - imec - University of Antwerp
miguel.camelo@uantwerpen.be

Steven Latré
IDLab - imec - University of Antwerp
steven.latre@uantwerpen.be

***Abstract*—In the last decade, several routing algorithms have been proposed in ad hoc wireless networks. However, most of them require either a high bandwidth, to maintain a full routing table, or suffer a high delay with packet flooding over the network, when the routes are discovered on-demand. As a solution, hybrid approaches, i.e. algorithms that combine on-demand route discovery with proactive updates of the available routes, have shown a good trade-off between low communication overhead and quality of the found routes. One of the approaches used in hybrid algorithms is Multi-Agent Reinforcement Learning (MARL), where the routing problem is addressed as a complex distributed control and learning problem. However, state-of-the-art MARL routing algorithms suffer from some limitations such as either lack of exploration or exploration at the cost of a high communication overhead, slow convergence under network dynamics, or no support for Quality of Service (QoS). In order to overcome such limitations, in this paper, we propose the Q^2-Routing algorithm, which merges existing techniques in wireless routing and enhances them by using techniques from the MARL domain. Simulation results showed that the proposed algorithm is able to outperform well-known ad-hoc routing algorithms in dynamic environments under QoS constraints.**

***Index Terms*—Wireless Routing, Wireless Networks, Ad-hoc Networks, Machine Learning, Reinforcement Learning, Multi-Agent Reinforcement Learning, Q-Learning.**

I. Introduction

A wireless network is a network consisting of several nodes that have wireless transmitting and receiving capabilities. In general, there are two modes that wireless networks can be deployed in: infrastructure and ad hoc. The first relies on a device managing the network and the second relies on self-managing nodes. Ad hoc networks have dynamic, sometimes rapidly-changing, random, multihop topologies which are likely composed of relatively bandwidth-constrained wireless links [1]. Typical applications of ad-hoc networks are military environments, emergency and rescue operations, personal area networking, environmental sensing, etc. [2], [3].

The Internet Engineering Task Force (IETF) suggests a set of desirable properties for routing in Mobile Ad Hoc Networks (MANETs) [4]. These include but are not limited to distributed operation, loop-free routing, on-demand route discovery combined with proactively looking for optimal links, and scalability. Additionally, the routing algorithms should have low computational cost, a small memory footprint and small bandwidth requirements [2], [5].

Traditional routing algorithms for wired and managed wireless networks, e.g. link state- and distance vector-based algorithms, have either a communication overhead that is too high to be used in an ad hoc environment or a slow convergence under rapidly changing conditions of the network topologies [6]. In other words, these routing protocols fail in wireless ad hoc networks because they lack the ability to cope with dynamic and resource-constrained networks.

Since traditional routing algorithms are not suitable for these networks, several algorithms have been proposed for wireless ad-hoc networking. These algorithms generally fall into three categories: proactive, reactive and hybrid [2]. Proactive algorithms create routes before packets are sent avoiding additional work when a route is needed. Maintaining the routing table requires a lot of bandwidth and most of the stored information is never used. In reactive algorithms, the routes are created when they are needed, and therefore only required routes are maintained. As a result, a delay is added before the first packet can be sent and the route discovery phase involves packets being flooded over the network.

In order to overcome some of these limitations, several hybrid algorithms have been proposed [1]. These algorithms try to find a good trade-off between both pro- and re-active algorithms. One of the approaches used in hybrid algorithms is Distributed Artificial Intelligence (AI), where the routing problem is addressed as a complex distributed control and information representation problem. AI routing algorithms can be based on either Machine Learning (ML) [7], [8], or Evolutionary Algorithms (EA) [9], [10]. In this paper, we will focus on hybrid approaches based on ML, and specifically on Multi-Agent Reinforcement Learning (MARL) [11].

MARL routing algorithms learn routes by trial and error, with no a priori knowledge about the topology, and under the topology changes. The convergence and adaptability of these algorithms under network dynamics depend on finding a good balance between exploration, sending packets via unknown routes aiming to discover new routes, and exploitation, routing using the best-known route. Although RL-based routing algorithms enhance the capabilities of traditional ad hoc routing algorithms, they still suffer from some limitations such as either lack of exploration or exploration at the cost of a high communication overhead, slow convergence under network dynamics and no support for Quality of Service (QoS).

Due to these limitations on most of the RL-based ad-hoc routing algorithms, in this paper, we propose the Q^2-Routing algorithm, a novel routing algorithm which merges existing techniques in wireless routing and enhances them by using techniques from the ML domain. The main contribution of

978-1-5386-6876-4/18/$31.00 ©2018 IEEE

this paper is a novel ad hoc routing algorithm based on Multi-Agent Reinforcement Learning (MARL) that is hybrid, i.e. on-demand route discovery combined with proactively updates of the available routes, fully distributed, loop-free, with both low computational cost and reduced communication overhead. The remainder of this paper is structured as follows: Section II will address several related works and identifies their problems. Section III discusses the proposed algorithm in detail and Section IV contains results obtained via simulations. Finally Section V contains the conclusion and discusses possible future extensions to this work.

II. Related Works

Two of the most well-known routing protocols in ad hoc networks are the Destination-Sequenced Distance Vector (DSDV) [6] and the Ad hoc On-Demand Distance Vector (AODV) [12] protocols. DSDV is a proactive algorithm that maintains updated routing tables at every node by flooding either small incremental updates or, when big changes occur, the entire routing table. DSDV suffers from route fluctuation caused by the way nodes handle received advertisements and does not use unidirectional links, which can lead to segmentation in the network [13]. AODV is a unicast reactive routing protocol that stores routing information in routing tables only for routes created on demand on each individual node. It is fault tolerant and some non-standard versions of it support QoS [14]. The main flaw of AODV is the large delays resulting from route re-discovery as a result of topology changes [12].

Given that AODV and DSDV are not well suited for highly dynamic ad hoc networks, other algorithms and protocols have been proposed. An example of such a routing protocol is the Better Approach To Mobile Adhoc Networking (BATMAN) algorithm [15]. BATMAN is a proactive routing solution that uses tables to contain routing information, but unlike other proactive approaches, the routing information only pertains to local neighbors and not to the network as a whole. Another approach for routing in highly mobile ad hoc networks is the Distance Routing Effect Algorithm for Mobility (DREAM) algorithm [16]. DREAM does not rely on flooding as the BATMAN algorithm does. Instead, DREAM tracks location information for the entire network proactively and uses it to predict where nodes will be and route packets to other nodes near to those predicted locations.

In the domain of MARL, the first RL based routing algorithm proposed in the literature was Q-Routing [8], which uses Q-Learning [17] as the learning algorithm. Q-values are stored in a table with an entry for every destination that is currently being used paired with every reachable neighbor. Based on a reward function, the RL algorithms try to find routes with low delay. Two versions of the algorithm were proposed in [8]. The first version does not include exploration and only the seemingly optimal path for a given situation is found. In the second version, an effort was made to include exploration by allowing nodes to exchange Q-values in an idealized way. Based on experiments with this "full-echo" modification, it was concluded that exploration does not improve the algorithm's performance.

An extension to [8], called Predictive Q-Routing, (P-QRouting) was proposed in [18]. The authors address the missing exploration issue of the original Q-Routing by using a fraction of traffic to do exploration. This exploration is done based on past experiences recorded on the path being explored. Nodes store the best Q-values for each path as well as the most recent values and an estimated recovery rate. These values are used to direct exploration traffic based on the expected state of the path, as determined by its recovery rate. Similar to Q-Routing, P-QRouting uses a routing policy where actions are chosen based on Q-values but enhanced by information from the recovery rates. Results showed that P-QRouting performs better than Q-Routing and does find optimal routes.

Another routing protocol for wireless ad-hoc routing based on Q-Learning is the Reinforcement Learning Based Geographic Routing Protocol (RLGR) [19]. RLGR is designed to be used in sensor networks, where nodes have access to their own location information and can exchange this information with neighbors. In RLGR, the battery lifetime of the network nodes is the most important metric to be optimized. In RLGR, negative rewards are sent back if the node has no energy or no route to the sink exists. Likewise, positive rewards are sent back if the node is itself the sink. Otherwise, a metric calculated based on the remaining delay and distance to the sink is sent back. RLGR uses an ϵ greedy approach to incorporate exploration and prevents loops using custom packet headers.

Q-Probabilistic Routing (Q-PR) [20] takes routing decisions based on both a probabilistic Bayesian statistics model and reinforcement learning. Nodes send packets to sets of nodes and only one node in the set is expected to forward the packet. To ensure no packet duplication, the candidate sets must be made up of neighboring nodes. Nodes that overhear a packet being forwarded by another node will discard their own copy. If a node must forward a packet, it makes a candidate set and decides, using a Bayesian decision model, to either discard or transmit the packet. Q-values are proportional to the expected number of retransmissions and are supplemented with a second metric based on a neighbor's past retransmissions.

Finally Cognitive Heterogeneous Routing (CHR) [21] and Self-aware Q-Routing (Q-CPN) [22] algorithms have been proposed. CHR is designed for networks where nodes have access to both WiFi and LTE interfaces. CHR uses Q-Learning to determine which interface to use for a packet transmission, where Q-values are based on network statistics like load, successful packet transmission and transmission rate. Q-CPN is designed for Cognitive Packet Networks (CPN) and supports QoS for traffic but it does not use reinforcement learning to accomplish this. Instead, CPN capabilities are used to ensure QoS. Q-CPN adds non-data traffic for exploration and sends custom ACK packets as replies to learning traffic to update Q-values.

In general, we can notice that while traditional ad hoc routing algorithms are limited for supporting all the desirable properties of routing in ad hoc networks, most of the hybrid routing algorithms based on machine learning do not include exploration and as such cannot guarantee to find the optimal

route. A related problem to this is that algorithms that do include exploration, like RLGR, do so using data traffic. As data traffic is routed via unknown paths with sub-optimal metrics, there is a high probability of packet loss and need for retransmission. Exchange of learning information in current algorithms based on reinforcement learning also often happens for every packet or using broadcasts, which is inefficient. Finally, few algorithms are available that consider more than one metric for QoS, and they usually do not include it in the learning process itself.

III. Enhanced Q-Routing with QoS support

The proposed work in this section is threefold. First, Reinforcement Learning (RL) and its application for routing are introduced. Second, several steps are proposed in order to overcome the limitations related to exploration and learning efficiency in Q-Learning based algorithms found in the literature. These steps avoid relying on route rediscovery phases to repair broken links, as seen in AODV and other reactive algorithms. After this, we describe our strategies to include support for the traffic being routed with Quality of Service (QoS). As a result, a novel routing algorithm called Q^2-Routing, based on reinforcement learning, is proposed.

A. Introduction to Reinforcement Learning

Reinforcement Learning (RL) is a machine learning technique in which a software agent only has limited knowledge about the environment, leading to a high degree of uncertainty concerning how the environment will react to the performed actions. Interactions with the environment are the only way for the agent to learn. In each state in the environment, the agent perceives a numerical reward, providing feedback to the agents actions. The agents goal is to learn which action to take in a given state, called the learning policy of the environment, in order to maximize the cumulative numerical reward. A commonly used RL algorithm is Q-Learning [17]. Using Q-Learning, knowledge regarding both reward prospects and environmental state transitions are obtained through interaction with the environment.

In Q-Learning, a Q-function is used to measure the quality of a state-action combination, based on the perceived rewards. Equation 1 shows how the Q-values are updated when action a is taken in state s, yielding a reward r and the environment change to state s'. In this equation, (s, a) is the state-action pair and $\alpha \in [0, 1]$ and $\gamma \in [0, 1]$ are the learning rate and the discount factor respectively. The learning rate α determines to what extent the newly acquired information overrides the old information, while the discount factor γ is a measure of the importance of future rewards.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \times max_a Q(s',a)) \quad (1)$$

In distributed systems, e.g. ad hoc wireless networks, the RL framework is enhanced to support the interaction among multi-agents. This framework is known as Multi-Agent Reinforcement Learning (MARL). The main challenge in MARL is designing strategies for sharing and merging knowledge among multiple agents with the aim of learning an optimal policy. In this paper, and following the traditional RL framework formulation, the distributed routing problem defines the environment as the network, both agents and states as the network nodes, actions as neighbor nodes and the reward value as a metric of some quality of the network performance. This is similar to the formulation introduced in [8].

B. A novel Q-Routing algorithm for ad hoc networks

Q^2-Routing is a hybrid routing algorithm in which nodes make routing decisions by choosing the neighbor associated with the optimal Q-value for a given destination as the next hop, similar to the approach used in Q-Routing [8]. However, Q^2-Routing adds an efficient exploration strategy together with a modified reward function to support QoS. The high-level pseudo-code of Q^2-Routing is shown in Algorithm 1.

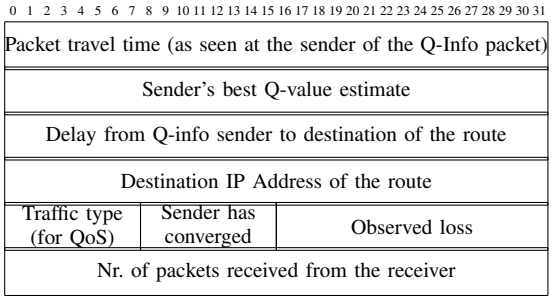
Algorithm 1 Q^2-Routing

```
   Phase 1: Bootstrapping
1: while No information about destination is known do
2:     Search for destination using broadcasting
3: Set Q-values to reflect initial path
   Phase 2: Learning
4: while Q-values at source have not converged do
5:     Source sends exploration traffic to improve Q-values
6: Decrease learning traffic data rate
   Phase 3: Data Routing
7: while Data remains to be sent do
8:     Source sends data traffic exploiting learned info
9:     Source sends learning traffic used to explore
10:    if large variations in optimal Q-value at source then
11:        source increases learning traffic data rate
12:    else
13:        source decreases learning traffic data rate
```

*1) **Bootstrapping**:* Q^2-Routing algorithm is a multi-phase hybrid algorithm that combines reactive and proactive elements for routing. When it is deployed in a network where no routing is needed, the algorithm is in the idle state. When a route becomes needed, the algorithm bootstraps itself using a route discovery phase similar to the one used in AODV. Packets are broadcast by the source and forwarded through the network until the destination replies. This bootstrapping phase sets up Q-values reflecting the initial path found so that Q-values can converge more quickly.

*2) **Learning**:* The proposed algorithm supplements data traffic with a secondary stream of traffic that is considered learning or exploration traffic. Learning information is only exchanged as responses to these learning traffic packets instead of for every packet sent as in Q-Routing. This increases the efficiency of the learning procedure and allows focusing on interesting links only. The packets used to exchange learning information are called Q-Info packets. Adding a separate data stream allows learning to be focused on routes that need exploring while limiting the amount of exploration being done on routes that are already sufficiently explored. Known information is regularly checked for correctness. The Q-Info packet

structure used in our implementation is shown below and the use of its fields are explained in the following paragraphs.



In the learning phase, only learning traffic is sent. This traffic is used to set the Q-values to realistic values and stop the learning phase when the Q-values on the path from source to the destination have converged to within a threshold of θ. The convergence is measured as a percentage difference $PD_X(Y,D)$ using equation 2, for a node X receiving a Q-value update from a node Y for a destination D:

$$PD_X(Y,D) = \begin{cases} 1 & \frac{|Q_X(Y,D) - Q_Y(Z,D)|}{Q_Y(Z,D)} \leq \theta \\ 0 & Otherwise \end{cases} \quad (2)$$

where Z is the best neighbor seen from node Y. Q-values in a given node can only converge if the node itself has a converged path to the destination. Note that the destination always advertises its route to itself as converged. This ensures that the learned Q-values are realistic for the entire path and helps to increase the algorithm convergence speed.

During the learning phase, routing decisions for learning traffic at all nodes are optimal with probability $1-\epsilon$ and lead to unknown paths with probability ϵ. Finding a good trade-off between using learning traffic for exploration and exploitation is key for fast convergence with low communication overhead. For example, $\epsilon < 0.5$ means Q^2-Routing exploits the known routes more, ensuring faster convergence to a stable route. It also limits the learning traffic overhead, but (possibly) is not optimal. Algorithm 2 shows the high-level pseudo-code for the learning phase in the network nodes.

Algorithm 2 Learning phase packet forwarding

```
1: while PD_X(Y,D) ≠ 1 do
2:     Generate random value rnd ∈ [0,1]
3:     if rnd < ε then
4:         Send learning packet to random next hop
5:     else
6:         Send learning packet to perceived optimal next hop
```

Note that in Q-Routing, Q-values are exchanged among nodes for every sent packet allowing nodes to quickly resolve routing loops and find alternative paths to be used instead. As Q^2-Routing only uses learning packets to update Q-values and these learning packets can be routed out of loops randomly, routing loops are resolved slowly. As such, Q^2-Routing uses a different approach. Nodes in Q^2-Routing resolve routing loops by trying to route packets seen more than δ times optimally to the source and treating it as learning traffic. In this way, Q-Info packets are exchanged in the loop and the Q-values causing the routing loop are updated quickly.

*3) **Data Routing**:* After the learning phase stops, the proactive data routing phase starts. When this phase begins, the rate of sending learning traffic is sharply reduced and the transmission of data traffic commences. However, learning traffic data rate can be increased or decreased at the source based on observed variations in Q-values in order to avoid putting a strain on the network.

During the data routing phase, the probabilities used for routing learning traffic are inverted for the source and ϵ is replaced by the parameter ρ in intermediate nodes, where $\rho \geq \epsilon$. This means learning traffic is sent via the optimal route with probability ϵ and via unknown paths with probability $1-\epsilon$ at the source, while in the intermediate nodes learning traffic is routed optimally with probability ρ. In this way, Q^2-Routing is able to explore alternative routes while still able to discover changes in the optimal route during the data routing phase.

The reason for using different probabilities in the intermediate nodes during the data routing phase is that these nodes must always have a reliable Q-value to provide to the source. To allow this, information about the optimal path must be kept up to date, i.e. exploitation is more important. For the source, exploring new routes is more important in this phase so the probability of using the best route for learning can be set lower.

The learning traffic rate is lowered when few changes in Q-values are observed and increased if large values are observed. A large change in a Q-value is determined by the percentage difference between the old Q-value and the new Q-value using equation 2. If $PD_X(Y,D) \neq 1$ and $LM_X(Y,D) \neq 1$, when $LM_X(Y,D)$ is described in equation 3 similar to $PD_X(Y,D)$, then the learning traffic data rate is increased by 50%. Adding the threshold τ was needed because it was observed that, for low values of θ, the learning traffic data rate would quickly go out of control and congest the network. Similarly, if a Q-value converges, the learning rate may be lowered by 50% every 15 seconds at the source node to ensure it is not unnecessarily sending learning traffic into the network.

$$LM_X(Y,D) = \begin{cases} 1 & \frac{|Q_X(Y,D) - Q_Y(Z,D)|}{Q_Y(Z,D)} \leq \tau \\ 0 & Otherwise \end{cases} \quad (3)$$

Learning traffic sent by the source node in the data routing phase uses the path that seems to be most likely to provide interesting information. This can be done via the optimal known route or via a route in need of exploration. The optimal path must be explored to ensure the information known by the source is still accurate. Routes in need of exploration are explored to discover alternative routes. If the source already has recent knowledge about all paths available to it, learning packets that are not sent to update the optimal path are discarded to limit the communication overhead.

C. Supporting QoS inside the learning algorithm

In order to add support for QoS in Q^2-Routing, multiple Q-values are associated with every neighbor-destination pair. One Q-value is used per defined traffic class, and this can vary based on the implementation. In this proposal, we support 3 QoS classes, each one represented by a unique Q-value. These classes are based on the QoS classification proposed in [23]. These multiple Q-values are updated simultaneously by every received Q-Info packet, which is extended in this algorithm to include information relevant to the QoS constraints.

The update function for the Q-values differs based on the traffic class it is associated with. This is done to ensure QoS constraints for the traffic class can be incorporated in the update function. If metrics are found to be in violation of QoS constraints, the Q-values are punished. The magnitude of this punishment is determined based on which constraint is being violated and by how much. The new update function is shown in equation (4). This is the normal Q-Learning update function proposed in [17], extended to punish the Q-value if constraints are not met.

$$\begin{aligned} Q_X(Y,D) =& (C_d \times C_j \times C_l) \times \\ & ((1-\alpha) \times Q_X(Y,D) + \alpha \times (r)) \end{aligned} \tag{4}$$

$$\text{with } r = \text{packet travel time + packet queue time + next hop's Q-value estimate}$$

The computations for the coefficient values are shown in equations (5), (6) and (7). These equations show how the type of QoS constraint being violated determines the size of the punishment for the Q-value.

$$C_d = \begin{cases} 1.0 & \text{if the delay metric was acceptable} \\ 1 + \dfrac{\text{observed delay value}}{\text{maximum allowable delay}} & \text{otherwise} \end{cases} \tag{5}$$

$$C_j = \begin{cases} 1.0 & \text{if the jitter metric was acceptable} \\ 2 + \dfrac{\text{observed jitter value}}{\text{maximum allowable jitter}} & \text{otherwise} \end{cases} \tag{6}$$

$$C_l = \begin{cases} 1.0 & \text{if packet loss (PL) is acceptable} \\ 4.0 & \text{if PL } < \text{ 2 times the allowed amount} \\ 8.0 & \text{if PL } < \text{ 10 times the allowed amount} \\ 12.5 & \text{otherwise} \end{cases} \tag{7}$$

To accommodate the classification proposed in [23], the values that had to be added to the Q-Info packet were path delay and packet loss. These values are calculated at every node individually and sent to their neighbors, allowing them to calculate their own values. This ensures that if a single node early on a path is causing a large amount of packet loss, the rest of the nodes on that path are not punished. However, paths unable to guarantee QoS are punished and will not be used.

In addition to supporting QoS, a final novel feature, called differential routing, was proposed. The differential routing feature allows packets to be routed differently based on their traffic class. This is a way of ensuring that important traffic is not hindered by other less important traffic streams. In our implementation, packets are routed via the best available path while leaving one route per traffic type that is more important than the traffic type being routed.

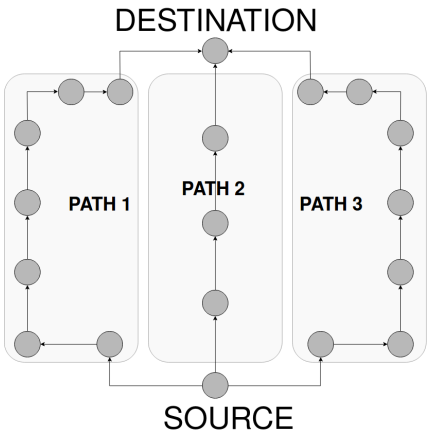


Fig. 1. Structure and location of the nodes in the simulated network.

IV. Results

In order to evaluate and compare the performance of the proposed algorithms, we implemented Q^2-Routing in ns-3 [24]. Experiments were done on a wireless ad hoc network topology of 19 nodes. The physical deployment of the nodes in the network creates three main paths between a predefined source and destination. A graphic representation of this network is shown in Figure 1. From the three paths, two paths denoted as path 1 and path 3 go via the edges of the network towards the destination. These paths are 8 hops long. The third path, denoted as path 2, is 4 hops long and is in the middle of the network. The experiments were performed by varying the levels of packet loss, delay, and jitter on the three paths available at the source. In this way, we were able to verify Q^2-Routing's ability to adapt to changes under network dynamics.

As a benchmark algorithm, we selected the AODV algorithm since this protocol is widely used in real deployments and its routing overhead is low. In addition, a non-QoS-aware version of Q^2-Routing, which we called Exploration Q-Routing (EQ-Routing), was used as a baseline RL algorithm to compare due to its enhanced capabilities in comparison to similar RL-based algorithms. For example, setting $\epsilon = 0$ allows it to perform as an improved version of Q-Routing [8]. For all the experiments, we used a discount factor γ equal to 1 and a learning rate α of 0.5. The other parameters were set to $\epsilon = 0.4$, $\rho = 0.5$, $\theta = 0.09$, $\tau = 0.1$, and $\delta = 1$. These values were found to guarantee a good trade-off between fast convergence and low routing overhead for our simulations. Values for all the parameters were determined experimentally.

A. QoS support and performance: Packet loss and delay

The first experiment sets an initial path and subsequently varies the packet loss and delay on the available paths. Table I shows the changes in the network conditions. Table II shows how the different algorithms react to the changes in network conditions. Figure 2 shows the behavior of the three algorithms as the packet delay in function of simulated time.

TABLE I
EVENTS ADDED TO THE SIMULATION IN THE PACKET LOSS EXPERIMENT

Time(s)	Path	Type	Value
0	3	delay	19 ms
0	2	delay	1 ms
0	1	delay	31 ms
1000	2	loss	66 %
2000	2	delay	60 ms
3000	3	loss	27 %
4500	2	delay	3 ms
4500	2	loss	0 %

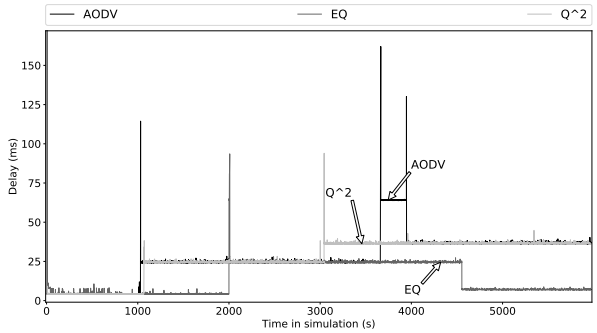


Fig. 2. Packet delay of traffic class B routed by the three algorithms at the destination. Packets are affected by changing packet loss and delay.

The results shown in Table II and Figure 2 reflect that Q^2-Routing switches from the fastest path 2 to the slower path 3 because of the loss starting at time $t = 1000$. AODV does the same because of hello broadcasts being dropped. This immediately shows the only situation that we found where AODV is able to outperform the non-QoS aware version of our algorithm, as it is oblivious to packet loss. Changes in the delay at $t = 2000$ make EQ-Routing change to path 3.

After introducing some packet loss at time $t = 3000$, Q^2-Routing is shown to react and switch away from path 3 to the slower path 1. AODV eventually does the same but not without switching to path 2 for some time first, which is still lossy. EQ-Routing does not react. Near the end of the simulation at time $t = 4500$ we remove delay and packet loss from path 2, making it the new optimal route. Q^2-Routing does not rediscover this, now optimal, path. It was unable to do this because the loss metrics on path 2 had not recovered and were still showing a large amount of loss even though there was none. As it is oblivious to packet loss, EQ-Routing was able to find this new optimal route on path 2.

The ratio of successful packets sent over time reflects these observations. Included in Figure 3 we can see that Q^2-Routing loses some successful packets around $t = 1000$ to packet loss but recovers quickly and does not lose successful packets due to the second packet loss around $t = 3000$. EQ-Routing on the other hand suffers greatly, AODV is only affected by the second occurrence of packet loss though this result is due to randomly dropping hello broadcasts.

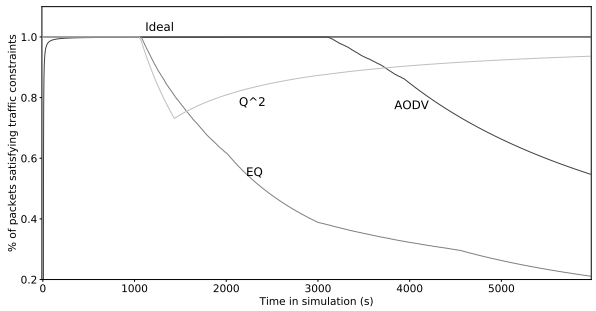


Fig. 3. Packet success ratio over time for packets of traffic class B routed by the three algorithms at the destination. Packets are affected by changing packet loss and delay.

TABLE II
TABLE SHOWING WHICH PATH IS TAKEN AT WHICH TIME FOR WHICH ALGORITHM FOR THE PACKET LOSS EXPERIMENT

Time(s)	AODV	EQ	Q^2
0	2	2	2
1000	3	2	3
2000	3	3	3
3000	2*	3	1
4500	1	2	1

B. QoS support and performance: Jitter

The second experiment shows the impact of jitter on the different algorithms. After some time in the simulation 60 ms of jitter is introduced on the optimal path 2. This jitter remains for 1000 seconds and is then removed. Table III shows the changes in the network conditions and Table IV shows how the different algorithms react to the changes in network conditions.

Figure 4 shows the resulting delay. Q^2-Routing chooses a slower but jitter-free route while AODV and EQ-Routing do not react. Q^2-Routing also quickly recognizes the disappearance of jitter and switches to the new optimal link. In Figure 5 a similar behavior is observed. When the jitter starts at time $t = 1000$ AODV, EQ-Routing and Q^2-Routing experience a decrease in successful packet delivery ratio. Q^2-Routing is the only algorithm that avoids the jittery path. This can be seen in the graph of Q^2-Routing in Figure 5 going towards the optimal successful packet delivery ratio before the jitter disappears at time $t = 2000$. For AODV and for EQ-Routing, the ratio keeps dropping until the jitter disappears.

TABLE III
EVENTS ADDED TO THE SIMULATION IN THE JITTER EXPERIMENT

Time(s)	Path	Type	Value
0	3	delay	70 ms
0	2	delay	24 ms
0	1	delay	58 ms
1000	2	jitter	60 ms
2000	2	jitter	0 ms

TABLE IV
PATH SELECTED OVER THE TIME FOR EACH ALGORITHM DURING THE FIRST EXPERIMENT WITH JITTER

Time(s)	AODV	EQ	Q^2
0	2	2	2
1000	2	2	3
1300	2	2	1
2000	2	2	2

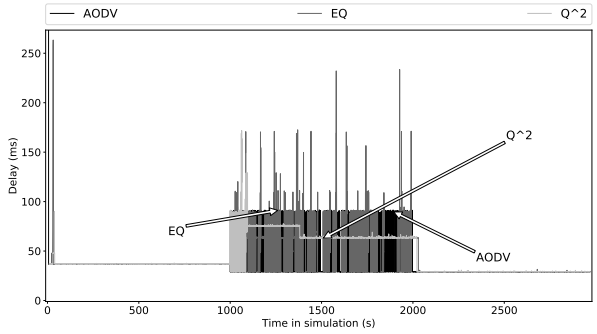


Fig. 4. Packet delay of traffic class A routed by the three algorithms at the destination. Packets are affected by changing jitter and delay in the network.

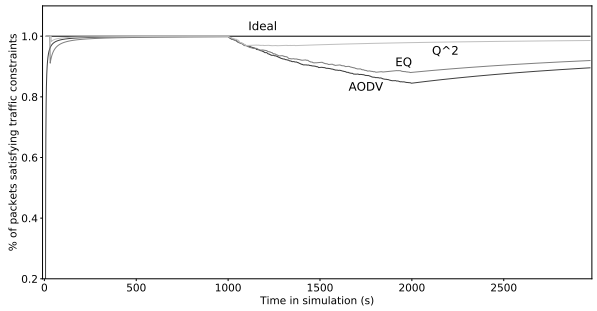


Fig. 5. Packet success ratio over time for packets of traffic class A routed by the three algorithms at the destination. Packets are affected by changing jitter and delay in the network.

C. Differential routing

The final experiment displays the proposed differential routing feature for Q^2-Routing. We use three traffic classes, with class A being most important, B the second most important and C the least important as defined in [23]. In the experiment the three paths were initially set with different delay values as shown in table V. We introduced packet loss on the second fastest path at time $t = 150$ and then switched the delays so that the lossy route becomes the lowest delay route at time $t = 1000$ to ensure there was enough time for the packet loss to be noticed.

Table VI shows how the paths used for the traffic classes change as the network conditions change. The observed delays in traffic packets belonging to the three classes are shown in Figure 6. Notice how the traffic of class A is routed via the lowest delay path, class B traffic is routed via the second fastest path, and class C traffic is routed via the slowest path. After the changes in delay, the results show that class A traffic chooses the second fastest path to avoid the loss on that path while traffic class B uses the lossy, fast route. The best effort traffic remains on the slowest route.

TABLE V
EVENTS ADDED TO THE SIMULATION IN THE DIFFERENTIAL ROUTING EXPERIMENT

Time(s)	Path	Type	Value
0	3	delay	12 ms
0	2	delay	180 ms
0	1	delay	70 ms
150	1	loss	0.5 %
1000	3	delay	190 ms
1000	2	delay	45 ms
1000	1	delay	12 ms

TABLE VI
PATH SELECTION OVER THE TIME FOR EACH TRAFFIC CLASS FOR THE DIFFERENTIAL ROUTING EXPERIMENT

Time(s)	Class A	Class B	Class C
0	3	1	2
1000	2	1	3

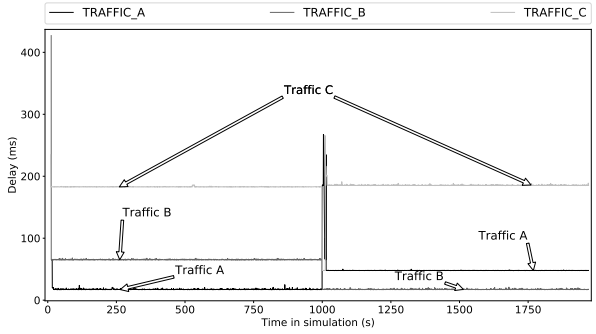


Fig. 6. Packet delay belonging to three different traffic classes as they are routed through the network by Q^2-Routing under delay and packet loss changes.

D. Communication Overhead

We define the communication overhead of a routing protocol r_1 with respect to a routing protocol r_2 as the ratio $\frac{cp_{r1}}{cp_{r2}}$, where cp_{r1} and cp_{r2} are the routing overhead, i.e. total number of control (non-data) packets sent by the routing protocol to create and maintain the routing table, of r_1 and r_2 respectively. Note that a communication overhead lower than 1 means better performance for r_1 compared to r_2. The communication overhead of EQ-Routing and Q^2-routing with respect to AODV for the experiment with jitter and the experiment with packet loss are shown in Table VII. Q^2-Routing has a lower communication overhead in both experiments, while EQ-Routing has a higher communication overhead in the packet loss experiment setup. This result is expected since EQ-Routing was not able to leave the lossy route. EQ-Routing received varying Q-value updates due to

the packet loss and increased its learning traffic because of that.

TABLE VII
COMMUNICATION OVERHEAD OF EQ-ROUTING AND Q^2-ROUTING WITH RESPECT TO AODV.

	Packet loss experiment	Jitter experiment
EQ-Routing	2.27	0.91
Q^2-Routing	0.96	0.94

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a novel Reinforcement Learning (RL)-based routing algorithm for ad hoc wireless networks. The proposed algorithm, called Q^2-Routing, improves upon existing RL algorithms and extends them by adding support for QoS. The algorithm was shown to perform well in a number of simulations with different network and traffic conditions. Simulation results showed that Q^2-Routing outperforms AODV in most cases and it is able to find a stable route that meets the QoS requirements while being able to adapt to changes in the network conditions.

By adding exploratory traffic, which changes dynamically according to the network state, the results showed that the routing overhead is similar to the low overhead of AODV. Additionally, the proposed algorithm avoids negatively impacting user data traffic by using separate traffic for learning. We were able to develop a method of learning that is more efficient than methods proposed in previous works. Finally, Q^2-Routing allows differential routing of packets belonging to different traffic classes. The proposed method ensures packets of less important traffic classes are not routed via routes that are needed for important traffic.

As future work, there are several research directions to extend this work. A first direction is to reduce the memory requirements of the learning algorithms as the size of the Q-tables is proportional to the number of neighbors, destinations and the number of traffic classes. A second direction is to increase the speed and efficiency of the learning by piggybacking learning values on traffic packets and by setting learning rates dynamically based on an observed reliability of Q-values. A third direction is to consider node mobility. Node mobility may invalidate Q-values at the moving node as well as at the nodes remaining stationary. Although nodes in the old location do notice the node has moved and will consider it unavailable, the moving node will be sending Q-Info packets containing the learned Q-values in the old location which will cause unpredictable routing behavior.

ACKNOWLEDGMENTS

Part of this research has been funded through the SMILE-IT project, which is funded by IWT (SBO).

REFERENCES

[1] C. E. P. (Ed.), *Ad hoc networking*. Addison-Wesley Longman, 2000.
[2] M. A. Alsheikh, S. Lin *et al.*, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 1996–2018, Fourthquarter 2014.
[3] D. G. Reina, S. L. Toral *et al.*, "The role of ad hoc networks in the internet of things: A case scenario for smart environments," in *Internet of Things and Inter-Cooperative Computational Technologies for Collective Intelligence*, 2013, pp. 89–113.
[4] S. Corson and J. Macker, "Rfc 2501: Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations," IETF, Tech. Rep., January 1999. [Online]. Available: https://tools.ietf.org/html/rfc2501.html
[5] S. Sahhaf, W. Tavernier *et al.*, "Routing at large scale: Advances and challenges for complex networks," *IEEE Network*, vol. 31, no. 4, pp. 108–118, July 2017.
[6] G. He, "Destination-sequenced distance vector (dsdv) protocol," Helsinki University of Technology, Finland, Helsinki, Finland, Tech. Rep., 06 2002.
[7] J. Wieselthier, C. Barnhart, and A. Ephremides, "A neural network approach to routing without interference in multihop radio networks," *IEEE Trans. on Communications*, vol. 42, no. 1, pp. 166–177, 1994.
[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 6 (NIPS)*, J. Cowan, D. and Tesauro, G. and Alspector, J., Ed., 1994, vol. 6, pp. 671–678.
[9] C. Lozano-Garzon, M. Camelo *et al.*, "A multi-objective routing algorithm for wireless mesh network in a smart cities environment," *Journal of Networks*, vol. 10, no. 01, pp. 60–69, 2015.
[10] G. Di Caro, F. Ducatelle, and L. M. Gambardella, "Anthocnet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks," *Transactions on Emerging Telecommunications Technologies*, vol. 16, no. 5, pp. 443–455, 2005.
[11] B. D. S. Lucian Buşoniu, Robert Babuška, "Multi-agent reinforcement learning: An overview, innovations in multi-agent systems and applications," *Innovations in Multi-Agent Systems and Application 1*, vol. 1, pp. 183–221, 2010.
[12] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561," July 2003.
[13] C. E. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," in *Proceedings of the conference on Communications architectures, protocols and applications*, September 1994, pp. 234–244.
[14] C. E. Perkins and E. M. Belding-Royer, "Quality of service for ad hoc on-demand distance vector routing," Working Draft, IETF Secretariat, Internet-Draft, November 2001. [Online]. Available: https://tools.ietf.org/html/draft-perkins-manet-aodvqos-00
[15] A. Neumann, C. Aichele *et al.*, "Better approach to mobile ad-hoc networking (b.a.t.m.a.n.)," IETF Secretariat, Internet-Draft, April 2008. [Online]. Available: https://tools.ietf.org/html/draft-wunderlich-openmesh-manet-routing-00
[16] S. Basagni, I. Chlamtac *et al.*, "A distance routing effect algorithm for mobility (dream)," in *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, ser. MobiCom '98. New York, NY, USA: ACM, 1998, pp. 76–84.
[17] Watkins, C. J. C. H. and Dayan, P., "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992698
[18] S. P. M. C. and D.-Y. Y., "Predictive q-routing: A memory-based reinforcement learning approach to adaptive traffic control," in *In Advances in Neural Information Processing Systems 8 (NIPS8*. MIT Press, 1996, pp. 945–951.
[19] S. Dong, P. Agrawal, and K. Sivalingam, "Reinforcement learning based geographic routing protocol for uwb wireless sensor network," in *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*. IEEE, 2007, pp. 652–656.
[20] R. Arroyo-Valles, R. Alaiz-Rodriguez *et al.*, "Q-probabilistic routing in wireless sensor networks," in *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*. IEEE, 2007, pp. 1–6.
[21] A. Al-Saadi, R. Setchi *et al.*, "Routing protocol for heterogeneous wireless mesh networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9773–9786, 2016.
[22] E. Gelenbe, Z. Xu, and E. Seref, "Cognitive packet networks," in *Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on*. IEEE, 1999, pp. 47–54.
[23] "ITU-T Recommendation G.1050: Network model for evaluating multimedia transmission performance over Internet Protocol," July 2016.
[24] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*, 2010, pp. 15–34.