New results on Q-routing protocol for wireless networks

Alexis Bitaillou, Benoît Parrein, Guillaume Andrieux

New results on Q-routing protocol for wireless networks*

Alexis Bitaillou[1], Benoît Parrein[1], and Guillaume Andrieux[2]

[1]University of Nantes, LS2N, Polytech Nantes, Nantes, France
[2]University of Nantes, IETR, IUT La Roche-sur-Yon, La Roche-sur-Yon, France

In the 90s, Q-routing assisted by reinforcement learning was introduced by Boyan and Littman with interesting results in terms of quality of service. Some recent works continue to promote the idea through improvement of the algorithm or specialized extensions. In this paper, we propose a simple modification to workaround the greedy behaviour of Q-routing by considering epoch notion. In comparison with the original Q-routing and the standard OLSRv2 under Qualnet simulator, we show that our extension provides an interesting improvement in terms of packet delivery ratio on the original irregular grid of Boyan and Littman with wireless links.
Keywords: Ad-hoc networks, Q-routing, Wireless networks, Qualnet simulator

1 Introduction

In the 90s, two new approaches appears to solve routing problem: *i)* bio-inspired algorithm and *ii)* reinforcement learning based algorithm. Q-routing [2] is one of the reinforcement learning based routing algorithm appeared. In their paper, Q-routing shown promising results. On their personal simulator, Q-routing offers a better average end-to-end delay than the Bellman-Ford protocol in high load condition. In fact, in congestion state, the Q-routing proposes alternative route based on the end-to-end delay while Bellman-Ford protocol is focused on the shortest path in terms of hops count. Those results have many potential applications especially for mesh and mobile ad-hoc networks (MANET). From their original work, many derived works has been proposed. A part of those are improvements of the algorithms such as AQFE [9]. Most of them are evaluated on home-made simulator. But, there are also specializations of Q-routing for specific applications such as mobility or cognitive radio.

*Supported by the COWIN project from the RFI Wise and Atlanstic 2020, Région Pays de la Loire

In this paper, we demonstrate how a short congestion can potentially degrade performance of Q-routing. So, we propose to integrate to Q-routing an epoch-inspired mechanism. Epoch mechanism is a method from machine learning to prevent some side effects of greedy behaviour such as local optimum problem. We evaluate our modified Q-routing with epochs on several scenarios on ad-hoc wireless networks. We compare it to the original Q-routing and nuOLSRv2, an implementation of OLSRv2 [3]. Our results show that our modification improved slightly the performance of Q-routing. It offers better performance than nuOLSRv2 in most our scenarios.

The organization of the paper is the following. In Section 2, we summarize some previous works about Q-routing. In Section 3, we detail the implementation of our distributed Q-routing protocol. Section 4 defines the experimental setup. Section 5 provides results in terms of QoS and a discussion as well. The last section concludes the work and draws some perspectives.

2 Related work

In this section, we see in more details Q-routing algorithms and other related works.

2.1 Q-routing

Watkins and Dayan [12] created a reinforcement learning algorithm called Q-learning in 1994. Two years later, Boyan and Littman [2] proposed to integrate Q-learning in routing algorithm. They named their algorithm Q-routing in reference to Q-Learning. In this algorithm, each node x looks for the lowest Q-value, defined using the Q function. The estimated delivery time from node x to node d by node y is noted: $Q_x(d, y)$. They define Q-value of function Q as:

$$\Delta Q_x(d, y) = \eta(q + s + t - Q_x(d, y)) \tag{1}$$

where η is the learning rate (usually 0.5 in [2]) q the unit of time spent in node x's queue, s the unit of time spent during the transmission between x and y and t as

$$t = \min_{z \in \text{neighbour of } y} Q_y(d, z). \tag{2}$$

In this case, the effective delivery time is the reward R and defined as: $R = q + s + t$. At the beginning, the Q-values are initialized with the value 0. Q-routing has a greedy strategy, so the first choice is very important. In order to make the first choice equitably, an exploration phase is needed to discover all the choices. During this phase, the Q-value is not updated.

Several networks topologies are tested in their work including an 6×6 irregular grid. The authors argue that only local information is used to proceed. The presented results of [2] concern only the 6×6 irregular grid. Q-routing is compared to Bellman-Ford's shortest path algorithm. In their works, Q-routing is not always able to find the shortest

path under low network load. Nevertheless, the latency is similar to the shortest path in low load condition. Q-routing clearly outperforms the shortest path in high load condition (even if the high load condition is not well-defined in [2]). However, when the traffic load decreases, Q-routing keeps the high load policy. The original approach is thus not adapted to dynamic changes.

2.2 Q^2-routing

The original Q-routing considers only the latency. Q-routing will select a low latency route even if the path loss more packets. Recently, Hendriks *et al.* [7] proposed an extension of Q-routing considering also the packet delivery ratio and the jitter. Their algorithm is called Q^2-routing. They adapted the Q function to include these QoS metrics:

$$Q_x(d, y) = (C_d \times C_j \times C_l)((1 - \alpha)Q_x(d, y) + \alpha r) \tag{3}$$

where C are coefficient depending on the traffic QoS requirement, α is the learning rate and r is $q + s + t$ in (1).

In their paper, they evaluated Q^2-routing on a topology composed of 3 paths on ns-3. It compared to an implementation of the original Q-routing and AODV [5]. Packets loss and delay appeared during the simulation on different paths in order to test Q^2-routing features. According to their results, Q^2-routing outperforms AODV and Q-routing in most of the test cases in terms of PDR, average delivery time and jitter. However, their scenario is designed to advantage Q^2-routing as the simulation event can only detect by Q^2-routing and some of them by Q-routing.

2.3 AQ-routing

Q-routing is a greedy algorithm. The mobility can easily degrade the performances. Serhani *et al.* [11] proposed an extension for Q-routing in order to improve performances in mobility scenario. They named their extension Adaptive Q-routing (AQ-routing). AQ-routing takes several concepts from OLSR [4] such as HELLO packets but also ETX metric [6]. Unlike the original Q-routing, AQ-routing doesn't use latency as routing metric. It uses a metric based on link stability:

$$Q_{metric_{ij}} = \alpha_{ij} \cdot \varphi(MF_j) + (1 - \alpha_{ij}) \cdot \lambda ETX_{ij} \tag{4}$$

where MF is the Mobility Factor, α the learning rate, $\varphi(MF_j)$ is defined as: $\varphi(MF_j) = \frac{a}{1-e^{\frac{-MF_j}{b}}}$. In their paper, they compared AQ-routing to OLSR (standard and with ETX metric version) on ns-3. On static test case, AQ-routing offers the best PDR but the worst average delivery time. On mobility test case, AQ-routing provides a stable average delivery time and the best PDR. Start to 4 m/s, the average delivery time is better with AQ-routing than with OLSR ETX. To obtain this performance, Serhani *et al.* have increased the complexity of Q-routing especially the computation of the reward.

2.4 Other extensions and derived works

There are many other extensions and derived works of Q-routing. For example, Kavalerov *et al.* [8] have improved Q-routing "Full Echo" with Adaptive Q-routing Full Echo (AQFE) and Adaptive Q-routing Random Echo (AQRE). AQFE improved the stability Q-routing Full Echo by adding a second dynamic learning rate. AQFE outperforms Q-routing on the original test cases of Boyan and Littman [2]. After the learning phase, AQFE can become unstable under some conditions. In order to reduce this instability, AQRE doesn't send update to all neighbours but to a set randomly chosen. Finally, they proposed Adaptive Q-routing with Random Echo and Route Memory (AQRERM) [9], an improved version of AQRE. However, AQFE and its derived have only been tested on home-made simulator. So, other quality of service metrics such as PDR are not evaluated.

Besides improvements, specialized extensions have been made. For example, Paul *et al.* [10] created an extension of Q-routing for cognitive radio. Zhang and Ye [13] made a Q-routing optimized for optical networks-on-chips.

3 Q-routing implementation details

In this section, we describe our implementation of Q-routing fully distributed and deployable on wired and wireless networks.

3.1 Implementation overview

We do not implement Q-routing from scratch, but our implementation is based on the Bellman-Ford basic implementation of Qualnet simulator. This basic implementation is bare-bones, there is no auxiliary function as we can have in OLSR [4] for example. We redefine the maximum route length (16 hops), the timeout delay (120 s), the maximum number of routes per packet (32 routes per packets), and the periodic update delay (10 s). Nodes have access to local information only. Additionally, we add the parameter η from Eq. (1) and the exploration phase duration. The routing table has been replaced by the function Q inspired of Eq. (1). The two next subsections describe how we totally distributed our implementation of the Q-routing protocol.

3.2 Latency measurement and header format

Q-routing aims to minimize the average delivery time. The Q function uses the duration of the transmission and the duration in-queue. To measure these times, we extend the header of the routing packet. The header of the routing packet contains a timestamp. Thanks to this information, the receiver can estimate the delay. The delay is sent back during the next update. This method has a little network overhead but needs two assumptions to work correctly. The latency is computed by the difference between the timestamp in the header and the moment when the packet is received according the local clock. So, the clock of the nodes needs to be synchronized to compute this difference.

As the reward uses the latency in micro-second, the synchronization need optimally to be of the order of the micro-second. In fact, the synchronization can be less precise, but the difference between the clocks has to smaller than the lowest latency. This is the first assumption. This mechanism can be replaced by using the "echo" function of ICMP.

The second assumption concerns the number of queues. Nodes need to have only one queue. If a node has more than one queue, the measure of duration in-queue will depend on the number of queues, the quantity and the priority of packets in the queues and finally the scheduler. So, in order to not depend on these parameters, the measure of the duration in-queue is more accurate when nodes have only one queue.

3.3 Route update mechanism

As nodes have only access to local information only, our protocol needs a mechanism to update their routing table. We propose to reuse the routing management to propagate routes. There are two types of update: periodic and triggered. During a periodic update, all nodes broadcast all their routes to their 1-hop neighbourhood. Periodic updates occur every 10 s. As broadcast a new route will be too slow with periodic updates, there is triggered and asynchronous update. To broadcast a new with periodic updates only, it needs in the worst case 10 s per hop. For example, on a topology in line of 6 nodes, the new route will broadcast from an end to the other in 50 s in the worst case. Triggered update happens when a new route is available, or a route has been modified. A triggered update is not sent if a periodic update will be sent in less than 150 ms.

We define a route as triplet value: destination, mask and next hop. We complete the structure by adding a timestamp, the value of the t from (1) and the current latency. The timestamp comes from the last timestamp of the routing packet of the destination. A new route is accepted if the distance is less than 16 hops. As we explained, the routing packets are timestamped. The timestamp is also integrated to data structure of the original route and acts as sequence number. A route update is always accepted if the timestamp of the update is newer than the current timestamp. If the timestamp of the update is equal to the current timestamp, the update is only accepted if it minimizes the Q-value of Eq. (1).

3.4 Epoch mechanism

In order to limit the greedy behaviour of Q-routing, we propose to add epoch-inspired mechanism. Epoch mechanism is a concept from machine learning in order to reset reward periodically and workaround the problem of local optimum. We define arbitrary the duration of the epoch to 300 s. At the end of the epoch, Q-routing creates a new empty Q-table. Q-routing starts an exploration phase, in which the new Q-table is filled. During this phase, the current Q-table works normally. At the end of the learning phase, the new Q-table replaces the current Q-table. This mechanism helps also to purge stale routes.

4 Experimental set-up

In this section, we describe the complete experimentation set-up and the results of our simulation. Our experimental plan concerns two wireless topologies: one simple with two main paths and the adaptation of 6×6 grid of Boyan and Littman [2]. The Table 3 sums up the modified or specific parameters. We use the default value for the other parameters. We benchmark three routing protocols: our implementation of Q-routing, our Q-routing with epoch mechanism and nuOLSRv2 (OLSRv2 [3] Niigata University implementation). OLSRv2 is the successor of OLSR, it is standardized routing protocol specialized in mobile ad-hoc networks (MANETs). We use Scalable Qualnet 8.2 as network simulator. 30 seeds are used for each combination of parameters.

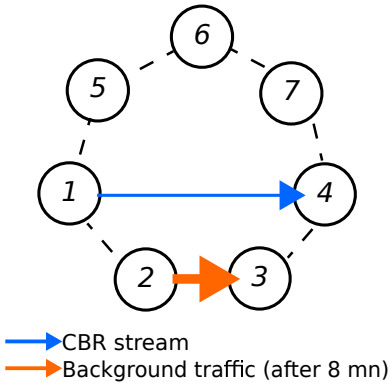

Figure 1: Our wireless toy example. Numbers correspond to the node ID.

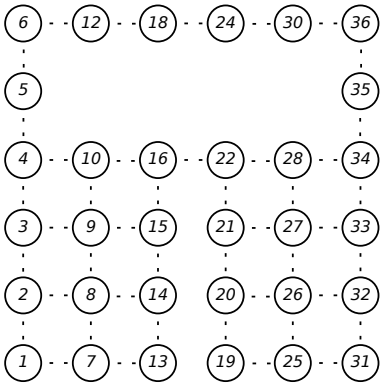

Figure 2: Adapted wireless irregular grid. Numbers correspond to the node ID.

4.1 Q-routing and its greedy behaviour, a toy example

Before evaluating Q-routing and Q-routing epoch on a complex topology, we evaluate them on a simple test case as depicted on Figure 1. Our test CBR is between node 1 and node 4 which are the source and the destination respectively. In this simple network, large background traffic appears on the shortest path between node 2 and node 3. In order to have two distinct paths, we move path away each other. The CBR source starts sending at 60 s and stop 60 s before the end of the simulation. The interval between two messages is 10 ms. To be sure that the routes are stables when background traffic appears, it starts after 8 minutes of simulation.

4.1.1 Scenario 1: one second congestion

For the first test, the background traffic appears at 8 minutes and stopped just a second after. The background traffic throughput is 5120 kb/s. The objective is to demonstrate the disadvantage of the greedy behaviour of Q-routing. In order to observe the clear

change of the average hop count, the simulations run over 30 minutes. We benchmark Q-routing and our Q-routing epoch.

4.1.2 Scenario 2: alternative path

The goal is simply to verify that our Q-routing implementation prefers the longer path (through node 5) as soon as congestion occurs. The background traffic start at 8 minutes and stop one minutes before the end of the simulation. The simulation time is 15 minutes because it is enough for this simple test case. We benchmark Q-routing, our Q-routing epoch and nuOLSRv2.

4.2 Adapted wireless irregular grid

In [1], we used the irregular 6×6 grid in [2]. But, when we replaced basically the wired link by wireless link, the irregular grid becomes a regular grid. We changed the location of some nodes. The topology is composed of two grids 4 nodes by 3 linked by two paths. Four nodes have been removed compared to the wired grid to keep the irregularity property. The Figure 2 illustrates this topology in a logical and compact form. For example, the node 9 can only communicate with nodes 3, 10 and 8. We benchmark Q-routing, our Q-routing epoch and nuOLSRv2.

4.2.1 Scenario 3: located congestion

In this case, there are 4 CBR streams on the adapted irregular wireless grid. The CBR streams start one minute after the beginning and stop one minute before the end of the simulation. All the CBR streams have the same throughput. The location of the CBR streams is detailed in Table 1. This test case shares the same idea of the tests on the toy example. The goal is to verify that Q-routing can balance CBR streams between two paths. The simulation time is 15 minutes

Source (Node ID)	Destination (Node ID)
35	5
4	34
9	27
26	8

Table 1: CBR streams location on the wireless grid (scenario 3)

4.2.2 Scenario 4: diffused traffic

In this case, there are 15 CBR streams on the adapted irregular wireless grid. The location of those CBR streams has been defined randomly. Their starting time and their stop time have been defined randomly but the CBR streams must start after one minute. All CBR streams have the same throughput. The settings of the CBR streams

are defined once and don't change between the simulations. The location of the CBR streams is detailed in Table 2. The goal is to evaluate Q-routing and Q-routing epoch on the adapted wireless grid with non-constant traffic. The simulation time is 30 minutes

Source (Node ID)	Destination (Node ID)	Start (s)	End (s)
19	12	62	721
28	9	175	537
31	35	217	1262
28	22	371	665
2	5	463	1088
14	15	632	951
25	6	832	1714
7	15	1006	1653
26	8	1168	1212
30	9	1241	1713
3	32	1303	1569
21	4	1592	1683
6	32	1613	1715
21	14	1661	1755
20	7	1705	1762

Table 2: CBR streams location on the wireless grid (scenario 4)

Feature	Parameter	Value
Network	Link	Wireless IEEE 802.11a 9 Mb/s link
		IEEE 802.11e link layer
	Topologies	Ring and irregular grid
Node	Number of queues	1 FIFO queue
	Mobility	No
CBR	Message size	512 bytes
	Start	1 min (scenario 2 and 3)
	End	1 min before the end (case 2 and 3)
Simulation	Seed	30 different seeds
	Duration	15 min (scenario 2 and 3)
		30 min (scenario 1 and 4)
Q-routing	Exploration	15 s (scenario 1 and 2)
		45 s (scenario 3 and 4)
	Epoch	300 s
	η	0.9

Table 3: Simulation parameters

5 Results

In this section, we present the results of the experimentation. We focus on three metrics: the average end-to-end delay (or average delivery time), the packet delivery rate (PDR) and the jitter. All those metrics are measured at the application layer (layer 7). Disordered messages are dropped by the receiver. Only the messages received and accepted contribute to the average delivery time and the jitter. The throughput of the CBR streams is expressed at the application layer.

5.1 Toy example

5.1.1 Scenario 1: one second congestion

For this first test case, we benchmark Q-routing and our modified Q-routing with epoch. We focus on the average hop count for the CBR stream. The Figure 3 shows the box plot of the average hop count for Q-routing over the time. Start to the congestion, the average delivery time increases. Q-routing uses the longer alternative path and stills use it after the congestion up to the end of the simulation. According to the Figure 4, Q-routing epoch uses also the longer alternative path, but unlike Q-routing, it finally returns on the shortest path. This little test case shows the advantage of the epoch-inspired mechanism. It makes Q-routing less sensitive to very short congestion. The reactivity of Q-routing could be increased by decreasing the epoch duration.

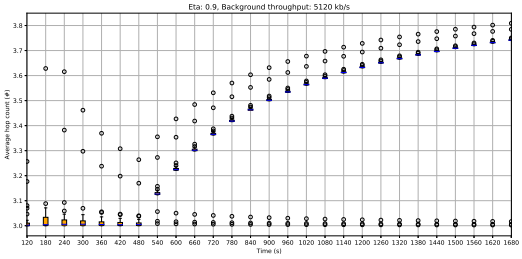

Figure 3: Average hop count for Q-routing (scenario 1).

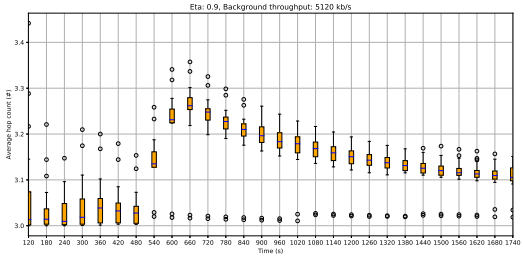

Figure 4: Average hop count for our Q-routing with epochs (scenario 1).

5.1.2 Scenario 2: alternative path

On this second test, we benchmark Q-routing and our modified Q-routing epoch and nuOLSRv2. We focus on the packet delivery ratio (PDR), the average delivery time and the jitter. The background traffic doesn't contribute to the average delivery time, the PDR and jitter. The Figure 5 shows the packet delivery ratio in function of the background traffic for Q-routing, Q-routing epoch and nuOLSRv2. According to our results, Q-routing and nuOLSRv2 have the same performance up to 3.5 Mb/s. Q-routing with "epochs" is less stable, the standard deviation is higher. nuOLSRv2 and

Q-routing with "epoch" have a singularity between 4.5 Mb/s and 4.8 Mb/s. From 4 Mb/s to 4.55 Mb/s, the packet delivery ratio falls suddenly to 70 % and increased up to 72 % between 4.55 Mb/s and 4.8 Mb/s. From 4.2 Mb/s, nuOLSRv2 drops a large part of the packet. The packet delivery ratio falls under 70 %. With Q-routing, the average PDR is higher (over 70 %), but the standard deviation is quite high. Q-routing epoch offers the better average packet delivery ratio than nuOLSRv2 from 4.8 Mb/s. The two versions of Q-routing outperform nuOLSRv2 in PDR only under high load condition (above 4.55 Mb/s). The Figure 6 shows the average delivery time in function of the background traffic for Q-routing, Q-routing epoch and nuOLSRv2. According to our results, the three protocols offer a comparable average delivery time up to 3.4 Mb/s. With nuOLSRv2, the average delivery time increases from 3.4 Mb/s up to peak at 260 ms at 3.56 Mb/s. It decreases after 3.56 Mb/s but the number of packets contributing to the metric decreases also. There is a singularity for nuOLSRv2 and Q-routing with "epochs" around 3.8 Mb/s. The singularity between 4 Mb/s and 4.8 Mb/s present in PDR is also present in average delivery time. With Q-routing, the average delivery time increased from 3.56 Mb/s to 3.9 Mb/s. It peaks at 60 ms on average. Q-routing offers the best average delivery time except on singular points. The Figure 7 shows the average jitter in function of the background traffic for Q-routing, Q-routing epoch and nuOLSRv2. The three protocols offer a comparable average jitter up to 3.4 Mb/s. The Q-routing with "epochs" has a better jitter than the original.

Our Q-routing epoch doesn't improve performances in terms packet delivery ratio and in average delivery time. It improves slightly the jitter on a range of background traffic throughput compared to the original. Q-routing delivers the best performance in terms of PDR and average delivery time under high load condition (above 4.5 Mb/s). The high standard deviation can be explained by the instability of the measured latency in wireless communication. This leads Q-routing making some wrong routing choice. This scenario by design puts nuOLSRv2 in difficulty as the best solution is to use the alternative path for the CBR stream.

5.2 Adapted wireless irregular grid

We evaluate Q-routing, Q-routing epoch and nuOLSRv2 on the wireless grid on the two tests cases. The average delivery time, the packet delivery ratio and the jitter are the average of all the CBR streams.

5.2.1 Scenario 3: located congestion

The four CBR streams contribute to the average delivery time, the packet delivery ratio and the jitter. The Figure 8 shows the packet delivery ratio following the throughput per CBR stream for Q-routing, Q-routing epoch and nuOLSRv2. Q-routing outperforms nuOLSRv2. The PDR with Q-routing is up to 44 % (at 410 kb/s per CBR) better than with nuOLSRv2. There is no significant difference between Q-routing and Q-routing epoch. The Figure 9 shows the average delivery time in function of the throughput per CBR stream for Q-routing, Q-routing epoch and nuOLSRv2. Q-routing provides low

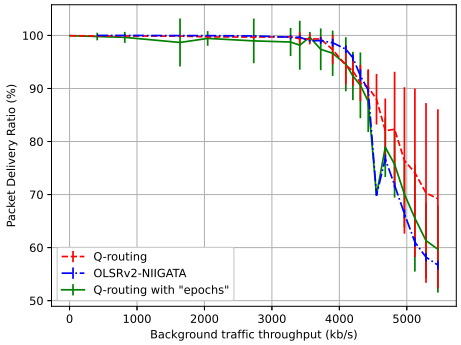

Figure 5: Packet delivery ratio on the toy example (scenario 2).

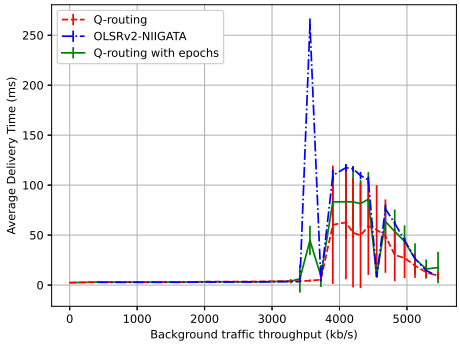

Figure 6: Average delivery time on the toy example (scenario 2).

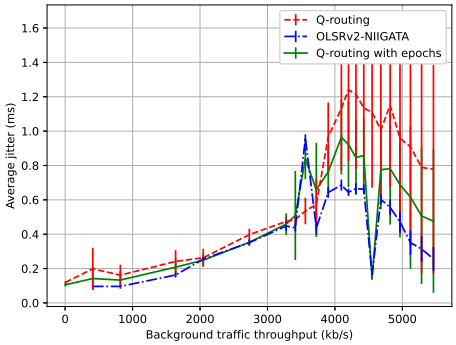

Figure 7: Average jitter on the toy example (scenario 2).

latency under low load condition (up to 820 kb/s). The latency increased up to overtake the latency with nuOLSRv2. There is no significant difference between Q-routing and Q-routing epoch. The Figure 10 shows the jitter following the throughput per CBR stream for Q-routing, Q-routing epoch and nuOLSRv2. nuOLSRv2 has the worst average jitter. Q-routing outperforms nuOLSRv2. There is no significant difference between Q-routing and Q-routing epoch.

The two versions of Q-routing outperforms under low load condition because Q-routing can more easily balance the CBR streams. The CBR streams (4, 34) and (35, 5) can use the ”upper” path even if the path is longer. nuOLSRv2 uses the upper path only for the CBR stream (5, 35) in the best case. As the link between the nodes 16 and 22 is saturated, nuOLSRv2 loses packets, but also increases the average delivery time. Interestingly, when the throughput per CBR increased, Q-routing loses its advantage in terms of average delivery time face to nuOLSRv2.

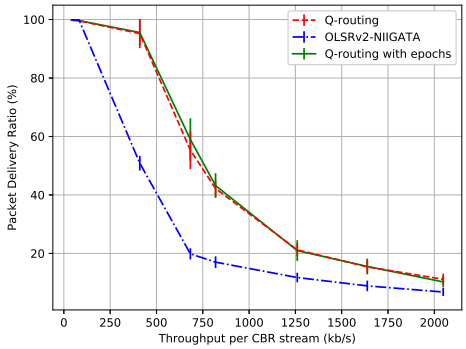

Figure 8: Packet delivery ratio on the scenario 3

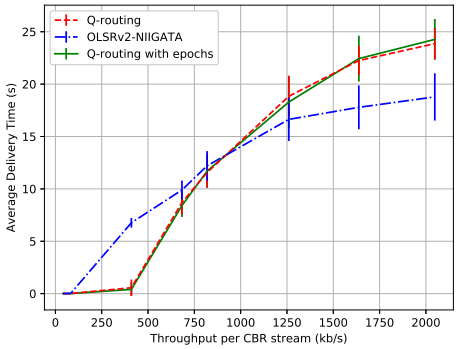

Figure 9: Average delivery time on the scenario 3

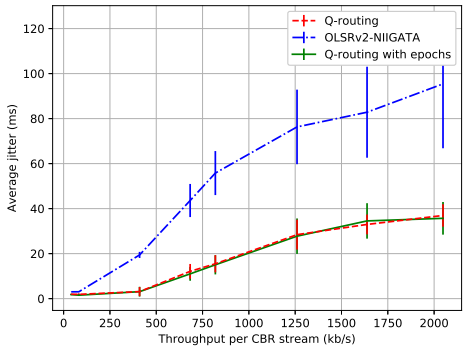

Figure 10: Average jitter on the scenario 3.

5.2.2 Scenario 4: diffused traffic

The 15 CBR streams contribute to the average delivery time, the packet delivery ratio and the jitter. The Figure 11 shows the packet delivery ratio following the throughput per CBR stream for Q-routing, Q-routing epoch and nuOLSRv2. The three routing protocol have the same shape. The packet delivery ratio decreases when the throughput per CBR increases. Q-routing epoch offers the best packet delivery. However, the difference with Q-routing and nuOLSRv2 is limited. There is 8 % between Q-routing epoch and nuOLSRv2 and 3 % between the two Q-routing in the best case (at 820 kb/s). The Figure 12 shows the average delivery time in function of the throughput per CBR stream for Q-routing, Q-routing epoch and nuOLSRv2. As the packet delivery ratio, the curves of the average delivery time have the same shape. On average, nuOLSRv2 is the slowest of the three protocols. Q-routing is up to 1 s faster. Q-routing epoch improve slightly this metric. The Figure 13 shows the jitter following the throughput

per CBR stream for Q-routing, Q-routing epoch and nuOLSRv2. nuOLSRv2 has the worst average jitter. Q-routing epoch offers the best average except between 1.6 Mb/s and 2.7 Mb/s where there is an instability.

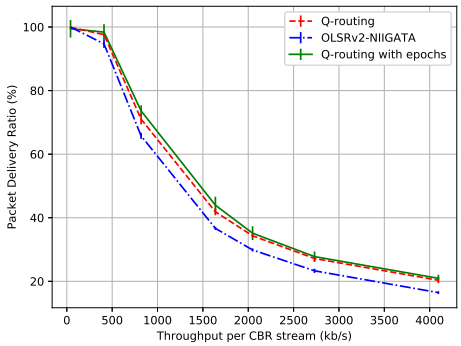

Figure 11: Packet delivery ratio on the scenario 4.

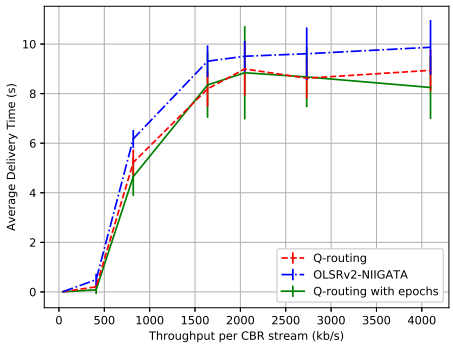

Figure 12: Average delivery time on the scenario 4.

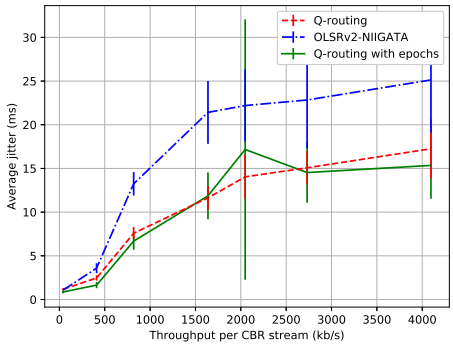

Figure 13: Average jitter on the scenario 4.

This scenario is the less static in terms of traffic so the performance of Q-routing was quite unexpected due to its greedy behaviour. Q-routing epoch provides a slight improvement but less than expected. The difference of performance between Q-routing and nuOLSRv2 can be explained by the unneeded verbosity of nuOLSRv2. In fact, nuOLSRv2 broadcasts more packets than Q-routing. Those packets are useful in mobility scenarios but this one.

5.3 Discussion

The tests over a simple wireless topology were very encouraging. Q-routing epoch gives the results expected. It returns on the shortest path when the congestion is finished.

It also chooses the alternative path to bypass the congestion. Q-routing outperforms nuOLSRv2 and Q-routing in PDR and in average delivery time up to 4.8 Mb/s. For the jitter, the improvement is not so obvious. On the wireless grid, the difference between Q-routing and Q-routing epoch is very slight, even on the scenario designed to favour Q-routing epoch (scenario 4).

Q-routing epoch has a higher computational cost and memory requirement than Q-routing. During the exploration phase, it has to maintain two Q-tables. The size of the Q-table is proportional to the number of 1-hop neighbour and the number of destinations. However, Q-routing epoch can remove staled routes from the Q-table at the end of the exploration phase. The size of the Q-table has an impact on update. On the wireless grid, each node broadcast up to 4 packets per update, so the routing overhead is high. This routing overhead can be reduced but at cost of a computation overhead. Those observations are agreed to [7] on difficulties for Q-routing to scale to large networks.

Another point concerns nuOLSRv2. The different scenarios are not designed to advantage nuOLSRv2. They don't include mobility. Their size in number of nodes is quite limited. For example, the wireless grid is composed of only 32 nodes. nuOLSRv2 implements advanced draft of OLSRv2. But, nuOLSRv2 is not totally compliant with the rfc7181. However, we think reasonably that the performance of nuOLSRv2 reflects the performance of a rfc7181-compliant implementation.

6 Conclusion

In this paper, we present an evaluation of our modification of Q-routing on the wireless standard IEEE 802.11. We experienced it on the professional packet driven simulator Qualnet on several scenarios. Q-routing epoch reacts as expected on the first scenario. The second scenario shows that our modification doesn't improve the performance of Q-routing. However, Q-routing epoch give a slight improvement on the wireless irregular grid compared to the original Q-routing. The two versions of Q-routing outperforms in PDR nuOLSRv2 in all our tests. Except the scenario 2 where the results are similar, they also clearly outperforms in average jitter nuOLSRv2. Nevertheless, the results in average delivery time depend on the scenario. We show that Q-routing doesn't need to be modified to give good results on our wireless grid even with diffused and changing traffic. However, our implementation of Q-routing doesn't have the auxiliary function of OLSRv2 and can't scale like it. So, our results can hardly generalize on specific scenarios or bigger topologies.

References

[1] Bitaillou, A., Parrein, B., Andrieux, G.: Q-routing: From the Algorithm to the Routing Protocol. In: Boumerdassi, S., Renault, r., Mühlethaler, P. (eds.) Machine Learning for Networking. pp. 58–69. Lecture Notes in Computer Science, Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-45778-5_5

[2] Boyan, J.A., Littman, M.L.: Packet routing in dynamically changing networks: A reinforcement learning approach. In: Advances in neural information processing systems. pp. 671–678 (1994)

[3] Clausen, T.H., Dearlove, C., Jacquet, P., Herberg, U.: The Optimized Link State Routing Protocol Version 2. RFC 7181 (Apr 2014). https://doi.org/10.17487/RFC7181, https://rfc-editor.org/rfc/rfc7181.txt

[4] Clausen, T.H., Jacquet, P.: Optimized Link State Routing Protocol (OLSR). RFC 3626 (Oct 2003). https://doi.org/10.17487/RFC3626, https://rfc-editor.org/rfc/rfc3626.txt

[5] Das, S.R., Perkins, C.E., Belding-Royer, E.M.: Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Jul 2003). https://doi.org/10.17487/RFC3561, https://rfc-editor.org/rfc/rfc3561.txt

[6] De Couto, D.S.J., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multi-hop wireless routing. In: Proceedings of the 9th annual international conference on Mobile computing and networking. pp. 134–146. MobiCom '03, Association for Computing Machinery, San Diego, CA, USA (Sep 2003). https://doi.org/10.1145/938985.939000, https://doi.org/10.1145/938985.939000

[7] Hendriks, T., Camelo, M., Latré, S.: Q2-Routing : A Qos-aware Q-Routing algorithm for Wireless Ad Hoc Networks. In: 2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). pp. 108–115 (Oct 2018). https://doi.org/10.1109/WiMOB.2018.8589161, iSSN: 2160-4886

[8] Kavalerov, M., Likhacheva, Y., Shilova, Y.: A reinforcement learning approach to network routing based on adaptive learning rates and route memory. In: SoutheastCon 2017. pp. 1–6 (Mar 2017). https://doi.org/10.1109/SECON.2017.7925316

[9] Kavalerov, M., Shilova, Y., Likhacheva, Y.: Adaptive Q-Routing with random echo and route memory. In: 2017 20th Conference of Open Innovations Association (FRUCT). pp. 138–145 (Apr 2017). https://doi.org/10.23919/FRUCT.2017.8071304, iSSN: 2305-7254

[10] Paul, A., Banerjee, A., Maity, S.P.: Residual Energy Maximization in Cognitive Radio Networks With Q-Routing. IEEE Systems Journal pp. 1–10 (2019). https://doi.org/10.1109/JSYST.2019.2926120

[11] Serhani, A., Naja, N., Jamali, A.: AQ-Routing: mobility-, stability-aware adaptive routing protocol for data routing in MANET–IoT systems. Cluster Computing **23**(1), 13–27 (Mar 2020). https://doi.org/10.1007/s10586-019-02937-x, https://doi.org/10.1007/s10586-019-02937-x

[12] Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning **8**(3), 279–292 (May 1992). https://doi.org/10.1007/BF00992698

[13] Zhang, W., Ye, Y.: An Approximate Thermal-Aware Q-Routing for Optical NoCs. In: 2019 IEEE/ACM Workshop on Photonics-Optics Technology Oriented Networking, Information and Computing Systems (PHOTONICS). pp. 22–27 (Nov 2019). https://doi.org/10.1109/PHOTONICS49561.2019.00009