Q-routing: From the Algorithm to the Routing Protocol

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

[1] University of Nantes, LS2N, Polytech Nantes, Nantes, France
alexis.bitaillou@univ-nantes.fr
[2] University of Nantes, IETR, IUT La Roche-sur-Yon, La Roche-sur-Yon, France

Abstract. Routing is a complex task in computer network. This function is mainly devoted to the layer 3 in the Open Standard Interconnection (OSI) model. In the 90s, routing protocols assisted by reinforcement learning were created. To illustrate the performance, most of the literature use centralized algorithms and "home-made" simulators that make difficult *(i)* the transposition to real networks; *(ii)* the reproducibility. The goal of this work is to address those 2 points. In this paper, we propose a complete distributed protocol implementation. We deployed the routing algorithm proposed by Boyan and Littman in 1994 based on Q-learning on the network simulator Qualnet. Twenty-five years later, we conclude that a more realistic implementation in more realistic network environment does not give always better Quality of Service than the historical Bellman-Ford protocol. We provide all the materials to conduct reproducible research.

Keywords: Routing protocol · Q-learning · Quality of Service · Qualnet · Reproducible research

1 Introduction

Routing is a complex task in computer networks. A common solution uses the shortest path algorithm as the Bellman-Ford routing protocol [2]. But the shortest path is not necessarily the one that maximizes the Quality of Service (QoS), especially in wireless networks. In order to solve routing problem, two original approaches appeared in the 90s: *(i)* bio-inspired algorithm and *(ii)* Q-routing. In bio-inspired approaches, the idea is to model ant colonies as routing algorithm [11]. In 1992, a new reinforcement learning algorithm was created by Watkins and Dayan: Q-learning [12]. Two years later, Boyan and Littman [3] experience Q-Learning in routing algorithm named Q-routing. 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

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

Published by Springer Nature Switzerland AG 2020
S. Boumerdassi et al. (Eds.): MLN 2019, LNCS 12081, pp. 58–69, 2020.
https://doi.org/10.1007/978-3-030-45778-5_5

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). But the work of Boyan and Littman is not complete. First, they do not supervise other QoS metric like the Packet Delivery Ratio (PDR). Second, their implementation is not totally specified. Even if their algorithm is distributable, we don't know if their implementation is really distributed. Third, their simulator is "home-made" and the simulation parameter is not highly depicted that makes this work hard to reproduce.

In this paper, we propose to evaluate the performances of Q-routing in a more realistic environment provided by a reference discrete event simulator like Qualnet. All of our experiences are available in a public git repository that makes this research reproducible and upgradable[1]. Furthermore, our implementation is fully distributed that enables to consider deployments in MANET. In such realistic conditions, we highlight that our Q-routing implementation over Qualnet simulator experiences routing message flooding and routing loops that leads to high packet loss rate in the original grid used by Boyan and Littman. We propose some counter-measures at the end of the paper.

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

2 Related Works

In this section, we provide more details on Q-routing in the related works.

2.1 Q-routing

Watkins and Dayan [12] created Q-Learning, a reinforcement learning algorithm in 1994. Two years later, Boyan and Littman proposed to integrate Q-Learning in routing algorithm. They name 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:

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

where η is the step size α in Q-Learning (usually 0.5 in [3]) 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 $t = \min_{z \in neighbour\ of\ y} Q_y(d, z)$. In this case, the effective delivery time is the reward R and defined as: $R = q + s + t$.

[1] https://gitlab.univ-nantes.fr/ls2n-rio/qrouting-qualnet, it assume a valid Qualnet license.

The Q-value is initialized with 0. Q-routing is greedy. It always chooses the lowest Q-value. Several networks topologies are tested in the work of [3]: 7-hypercube, the 116-nodes LATA telephone network and a 6×6 irregular grid. The authors argue that only local information is used to proceed. The presented results of [3] concern only the 6×6 irregular grid. Q-routing is compared to Bellman-Ford shortest path algorithm. The average latency is higher in the exploration phase because the packets are randomly sent. Then, the latency is similar to the shortest path in low load condition. Q-routing is not always able to find the shortest path under low network load. But Q-routing clearly outperforms the shortest path in high load condition (even if the high load condition is not clearly defined). When the traffic load decreases, Q-routing keeps the high load policy. The original approach is thus not adapted to dynamic changes.

2.2 Predictive Q-routing

Choi and Yeung [4] proposed an improvement for Q-routing in 1996. Their algorithm is Predictive Q-routing (PQ-routing). It corrects the problem of sub-optimal policy after a high network load. Unlike Q-routing, PQ-routing is not memory-less. It keeps track of the best Q-value. Under low network load, PQ-routing uses the shortest path algorithm to get the optimal policy. Under high network load, PQ-routing uses the latency as main metric. Thanks to its memory, it can come back to the optimal policy when the network load decreases. PQ-routing is composed of 4 tables: Q (estimation), B (best Q-value), R (recovery), U (last update). Q is defined like in [3] (cf. Eq. 1). Each table can be finely tuned by parameters. These tables are updated at each packet reception.

PQ-routing is compared to Q-routing. Two network topologies are tested: a 15-nodes network and the 6×6 irregular grid from [3]. PQ-routing performs better than Q-routing independently of the network load. Under high network load, PQ-routing is quite similar to Q-routing. These results are also obtained on a "home-made" network simulator. The average delivery time is the only metric provided. PQ-routing has higher memory requirements due to additional tables and higher computational cost because the 4 tables need to be updated.

2.3 Dual Reinforcement Q-routing

In 1997, Kumar and Miikkulainen [8] proposed to add backward exploration to Q-routing. As there is forward and backward, they name their algorithm Dual Reinforcement Q-Routing (DRQ-Routing). The evaluated network is the 6×6 irregular grid from [3]. They use once again a "home-made" simulator. They define low network load as 0.5 to 1.5 packets per simulation step, medium as 1.75 to 2.25 and high as 2.5 or more. DRQ-Routing is compared to Q-routing and shortest path. The average delivery time is the unique metric of comparison. According to their results, DRQ-Routing outperforms Q-routing in low network load. It outperforms Q-routing and shortest path in medium and high network load. Moreover, DRQ-Routing learns twice faster than Q-routing. They

use unbounded FIFO queues. This means that a packet cannot be dropped by queue overflow. This simplifies the simulation but it cannot be applied in realistic network.

2.4 Other Related Works

There are several other related works about Q-routing. Many other extensions of Q-routing have been proposed: Gradient Ascent Q-routing [9], Enhanced Confidence-Based Q-routing [14], K-Shortest Paths Q-routing [7], Credence based Q-routing [6]. There are also extensions for wireless networks for Q-probabilistic routing [1] and for the Mobile Ad-hoc Networks (MANETs) [10]. Xia *et al.* [13] propose to use Q-routing in cognitive radio context. The average delivery time is the only metric used for most of those papers. Arroyo-Valles *et al.* do not use average delivery time. Instead, they prefer to use packet delivery ratio. Except [13] on OMNET++, those related works use their own simulator. Unfortunately [13] do not give any details about their implementation.

3 A Distributed Q-routing Implementation

In this section, we describe our implementation of Q-routing and the complete experimentation set-up. Our experimental plan concerns two topologies: one simple with two main paths and the 6×6 grid of [3].

3.1 Implementation

We have implemented Q-routing based on the Bellman-Ford implementation of Qualnet. The routing table has been replaced by the function Q (see Eq. 1). We reuse some parameters from Bellman-Ford implementation such as 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). Our protocol is totally distributed. As Bellman-Ford protocol, nodes have access to local information only. Every 10 s, nodes broadcast their routing to their 1-hop neighbourhood. During a periodic updates, all routes are broadcast. Additionally, there are aperiodic updates. Aperiodic updates help to broadcast more quickly new route for example. There are also triggered to broadcast new latency value. During aperiodic updates, only the new or modified routes are sent.

The header of the routing packet contains a timestamp. Thanks to this information, the receiver can know the delay thanks to this timestamp. This method limits the network overhead but nodes have to use only one queue. The clock of the nodes needs to be synchronized. The level of synchronization determines the accuracy. The function Q is updated when the routes are updated, at least every 10 s. We define a route as a destination, a subnet mask, a next hop, a distance, a latency, two timestamps (local and from the original node), the incoming and outgoing interfaces. The first timestamp is defined when a node gets the latency

from its 1-hop neighbour. This timestamp is not modified when the information is broadcast. The second timestamp is local and is used for checking the time out. This timestamp is updated when the node receives an update for this route. Q-routing has an exploration phase during 2 s. During the exploration phase, the Q-values are not updated.

3.2 Simulation Parameters

We use Qualnet 8.2 (from Scalable Networks Technology) as network simulator. The networks are composed of symmetric 10 Mb/s wired links. In order to prevent side effect, we used an abstract link layer. All links propagation delays are set to 1 ms that defines the latency of one hop. Unlike Kumar [8], each node has a finite FIFO queue of 150k packets. With Qualnet (and other discrete event simulators), the seed of the pseudo-random generator has a high impact on the results. For the same seed, the number of trials doesn't have a significant importance. Both foreground traffic and background traffic are constant bit rate (CBR) traffic flow. All CBR messages are 512 bytes long. CBR messages are sent in UDP packets. The CBR receiver drops disordered messages. We compare Q-routing to Bellman-Ford protocol because it uses the shortest path. The Table 1 sums up the simulation parameters.

Table 1. Simulation parameters

Element	Parameter	Value
Network	Link	Symmetric 10 Mb/s wired link
	Link propagation	1 ms
	Link layer	Abstract MAC
Node	Number of queue	1 FIFO queue
	Queue size	150k packets
CBR	Message size	512 bytes

A Toy Example. Before evaluating Q-routing on the topology of [3], we test it first on a simple topology as depicted on Fig. 1. Our test CBR is between node 1 and node 4 which are the source and the destination respectively. In this simple network, a large background traffic appears on the shortest path between node 2 and node 3 as shown Fig. 2. The goal is simply to verify that our Q-routing implementation prefers the longer path (through node 5) as soon as congestion occurs. The CBR source starts sending at 1 s and stop at the end of the simulation. The interval between two messages is 1 s. Background traffic appears at 15 min. The simulation time is only 60 min for this toy example. We test 10 different and arbitrary seeds.

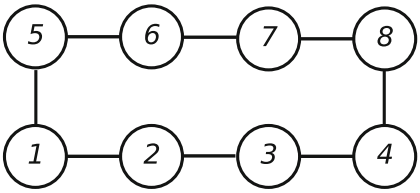

Fig. 1. Simple topology to test our implementation. Numbers correspond to the node ID.

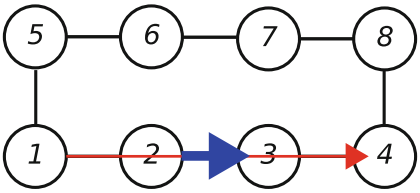

Fig. 2. CBR traffic flow location on the simple topology at 15 min. Numbers correspond to the node ID. (Color figure online)

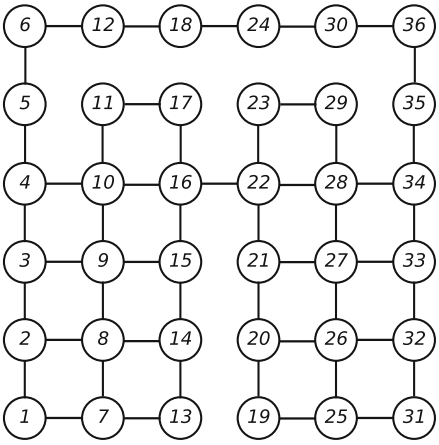

Fig. 3. The 6×6 irregular grid used by Boyan and Littman [3]

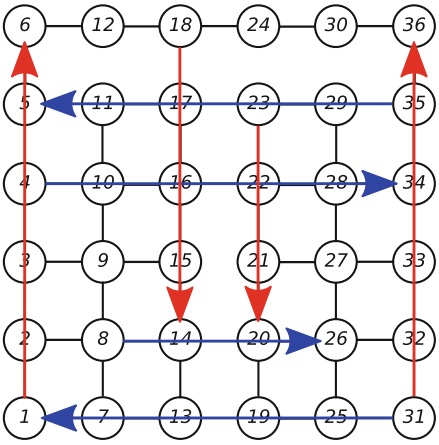

Fig. 4. CBR location on the 6×6 irregular grid (Color figure online)

The Irregular Grid. We evaluate then our implementation on the original 6×6 irregular grid from [3]. The Fig. 3 illustrates this particular grid and the Fig. 4 shows the location of the CBR couples. The location of the CBR flow is arbitrary because Boyan and Littman [3] don't give so many details in their works. However, we experience that the location of the couples has a great impact on the results. As in [3], the CBR traffic flow will be alternatively "horizontal" and "vertical" every 30 min. The "vertical" CBR traffic flow (in red) will be active then it will be the "horizontal" CBR traffic flow (in blue). All CBR sources have the same throughput for a given simulation as depicted on the 2^{nd} column of the Table 3. We used 36 different and arbitrary seeds in order to validate our simulation. The simulation time is 180 min.

4 Results and Discussion

In this section, we present the results of the experimentation. We focus on two metrics: the average end-to-end delay (or average delivery time) and the packet delivery rate (PDR). Both are measured at the application layer (layer 7).

Table 2. Comparison between Q-routing (Q-r) and Bellman-Ford protocol (B-F) with background traffic on the toy example, 10 seeds.

Protocol	Throughput of bg traffic	Avg. EtE delay (ms)		PDR (%)	
		Average	SD	Average	SD
B-F	8.2 Mb/s	4.36	<0.01	100	0
B-F	10.2 Mb/s	82.9	<0.01	66.6	0.07
B-F	11.7 Mb/s	67.7	0.01	50.3	0.08
B-F	13.7 Mb/s	67.9	0.11	50.1	0.09
Q-r	8.2 Mb/s	4.37	<0.01	100	0
Q-r	10.2 Mb/s	6.57	0.02	100	0.04
Q-r	11.7 Mb/s	6.57	0.02	100	0.04
Q-r	13.7 Mb/s	6.58	0.05	100	0.07

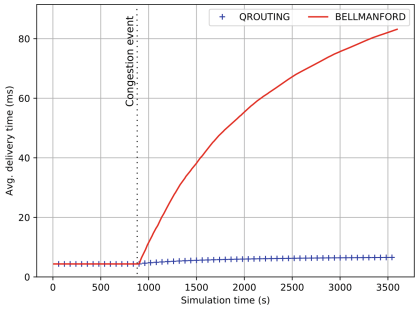

Fig. 5. Average End-to-End delay over the simulation, measured by the CBR between node 1 and node 4.

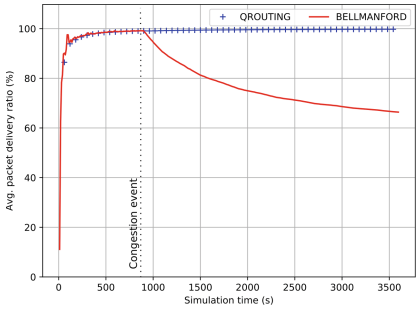

Fig. 6. PDR over the simulation, measured by the CBR between node 1 and node 4.

Results on the Toy Example. The Table 2 sums up the results with different throughput of background traffic for the simple grid of Fig. 1. On average we experience low delay and the highest PDR for Q-routing for the considered traffic pattern. The singularity at 10.2 Mb/s for BF protocol means that we are in a congested status with relative good PDR but a degraded end-to-end delay.

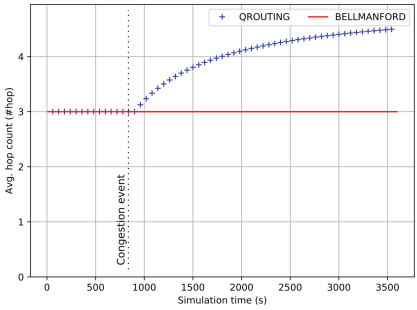

Fig. 7. Average hop count over the simulation, measured by the CBR between node 1 and node 4.

Table 3. Comparison between Q-routing and Bellman-Ford on the irregular grid, 36 seeds.

Protocol	Throughput	Avg. EtE delay (ms)			PDR (%)			Avg. drop of IP packets (number of packets)		
		Average	Median	SD	Average	Median	SD	No route to host	Expired TTL	Queue overflow
B-F	4.1 kb/s	9.27	9.27	<0.01	100	100	<0.01	2	0	0
B-F	41 kb/s	9.27	9.27	<0.01	100	100	<0.01	3	0	0
B-F	410 kb/s	9.27	9.27	<0.01	100	100	<0.01	18	0	0
B-F	4.1 Mb/s	9.27	9.27	<0.01	100	100	<0.01	164	0	0
B-F	8.2 Mb/s	51.0	51.4	7.70	88.0	88.5	0.74	49.0×10^3	0	10.4×10^6
B-F	10.2 Mb/s	242	242	12.9	75.3	75.4	1.17	295×10^3	14	26.8×10^6
B-F	13.7 Mb/s	254	255	14.1	56.5	56.5	0.96	398×10^3	3	62.8×10^6
Q-r	4.1 kb/s	9.33	9.33	<0.01	99.9	99.9	<0.01	14	32	0
Q-r	41 kb/s	9.33	9.33	<0.01	99.9	99.9	<0.01	102	318	0
Q-r	410 kb/s	9.79	9.70	0.38	99.9	99.9	<0.01	980	2776	0
Q-r	4.1 Mb/s	11.8	11.7	0.84	99.7	99.8	0.20	10×10^3	10×10^3	140×10^3
Q-r	8.2 Mb/s	129	135	91.6	82.7	79.7	9.74	21×10^3	54×10^3	17×10^6
Q-r	10.2 Mb/s	680	674	32.6	65.1	66.7	8.12	115×10^3	258×10^3	45×10^6
Q-r	13.7 Mb/s	673	675	42.4	47.1	48.2	5.73	141×10^3	280×10^3	85×10^6

This singularity disappears for Q-routing. Figures 5, 6 and 7 give a temporal representation of the simulation. The background traffic is 10.2 Mb/s. The dotted line represents the appearing of the congestion. With Q-routing, the average delivery time stays low compared to Bellman-Ford protocol (Fig. 5) when the congestion occurs at 15 min (900 s). Moreover, Q-routing drops only few packets as depicted on Fig. 6. Finally, the average hop count (Fig. 7) shows that Q-routing bypass the congested path through a longer way. There is no more than 1 packet lost. The throughput is 1 packet per second. The results do not show clearly the convergence time. If we consider that dropped messages are due to congestion, we estimate it no higher than 2 s. Q-routing performs pretty well on our simple test. But Q-routing uses a greedy strategy. Even if the congestion disappears, the packets will continue to pass by the longest route.

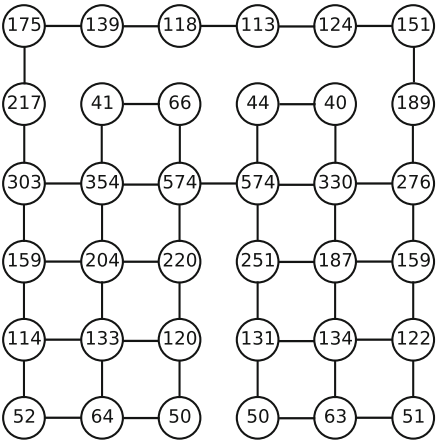

Fig. 8. Policy summary: Bellman-Ford under 4.1 Mb/s (medium load).

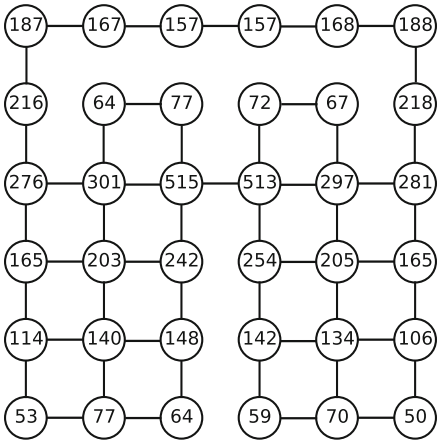

Fig. 9. Policy summary: Q-routing under 4.1 Mb/s (medium load).

Results on the Irregular Grid. Boyan and Littman [3] expressed their results in the unit of their "home-made" simulator. The time is in simulator time. The network load and its unit are not defined. We express the duration in seconds and the network load in bytes per second. Thus, we cannot compare directly their results to ours. The Table 3 sums up the results for the irregular grid. We experience many packet drops. We add to this table the origin of those drops: (1) route starvation i.e. the "no route to host" message; (2) abnormal route length (monitored by the Time-To-Leave parameter); (3) queue overflow. For Q-routing, the number of packets dropped due to expired TTL is relatively high even at low throughput. Routing loops can only explain the number of packets dropped due to expired TTL. Performance under low network load is similar to Bellman-Ford protocol. The behaviour of the Q-routing protocol changes between 410 kb/s and 4.1 Mb/s values. The average delivery time increases and the PDR decreases up to reach 47.1% at 13.7 Mb/s. The number of dropped packets by queue overflow reaches 140×10^3 packets at 4.1 Mb/s. At 13.7 Mb/s, the number of dropping by queue overflow is very high, around 85×10^6 dropped packets for 144×10^6 messages sent. From a topological point of view, the Figs. 8 and 9 resume the routing policies by showing the average number of routes passing by each node under medium load at the end of the simulation. For Bellman-Ford protocol, the distribution of routes is very similar to [3]. However, the Fig. 9 gives a different result compared to [3]. With Q-routing, nodes 16 and 22 are less solicitate but the number of routes is not so balanced as in [3].

We extract the number of routing packets sent and received. With the Bellman-Ford protocol, there are not so much aperiodic updates because the minimal distance to a destination is constant in our scenario. For example, under low network load, aperiodic updates represent 7 packets per hour and per node. The situation is different with Q-routing. The average number of packets sent

for periodic updates is around 15×10^3 packets over the simulation. The average number of packets sent for aperiodic updates is around 661×10^3 packets. Aperiodic updates represent about 90% of routing packets. At 4.1 kb/s, on the irregular grid, there are about 190 routing packets per seconds for 4 data packets per seconds.

4.1 Discussion

Although the test over a simple topology is very encouraging, our test on the irregular grid doesn't give a similar result to [3]. As the authors of this work, we expected Q-routing to outperform Bellman-Ford protocol under high network load condition. The implementation of the Q-routing in a real packet simulator in a distributed manner is the main reason from our point. Traffic that needs to pass through those links will be penalized. The quantity of packets dropped by queue overflow is really important. This is not considered in [3]. We made some additional tests with a larger queue in order to give more chance to have a higher latency but the results have not changed significantly.

Packets dropped by expired TTL are the main cause of dropping under low network load. An additional mechanism is needed in order to prevent routing loops. For example, source routing or tracking packet's ID can be used periodically to check the route. Another possibility is to change the reward in the Q function. Indeed, the distance (in hop count) could be considered in order to help the choice of the best route. The reward can also take account of the number of dropped packets.

Moreover, Q-routing has a higher memory requirement than Bellman-Ford protocol. Indeed, Q-routing needs memory to store all destinations by all the next hops whereas Bellman-Ford protocol stores also all destinations only once. Furthermore, computational costs are higher with Q-routing due to data structure and the quantity of data. For example, on the grid, a node with 4 neighbours like node 16 has to store 144 routes. When it broadcasts all routes, it needs 5 packets. With the Bellman-Ford protocol, nodes have 36 routes. They need to send just 2 packets to broadcast all routes. The network overhead is also higher with Q-routing. The value of the latency changes a lot so the number of aperiodic updates can be very high. In order to limit the number of aperiodic updates, several solutions are possible. For example, the aperiodic update can be schedule only if the difference between the old and the new value is greater than a threshold. Moreover, we can introduce partial flooding mechanism as it is proposed in MANET protocol as OLSR [5]. In this protocol, Multi-Point Relays reduce drastically routing message flooding.

5 Conclusion

In this paper, we presented a distributed implementation of the Q-routing algorithm. We experienced it on the professional packet driven simulator Qualnet. Q-routing works well on a simple topology composed of 8 nodes. However, the

Quality of Service parameter as the end-to-end delay and the packet delivery ratio are degraded as soon as Q-routing protocol is deployed on the irregular grid proposed in [3]. High network overhead and routing loops are the main explanation in real networks conditions. They also explain routing starvation. We provide all the materials to conduct reproducible research. Auxiliary functions to prevent flooding, the integration to existing MANET protocols (as OLSR) and the extension of Q function are the perspective of this work.

References

1. Arroyo-Valles, R., Alaiz-Rodriguez, R., Guerrero-Curieses, A., Cid-Sueiro, J.: Q-probabilistic routing in wireless sensor networks. In: Sensor Networks and Information 2007 3rd International Conference on Intelligent Sensors, pp. 1–6, December 2007. https://doi.org/10.1109/ISSNIP.2007.4496810
2. Bellman, R.: On a routing problem. Q. Appl. Math. **16**(1), 87–90 (1958). https://doi.org/10.1090/qam/102435
3. 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)
4. Choi, S.P., Yeung, D.Y.: Predictive Q-routing: a memory-based reinforcement learning approach to adaptive traffic control. In: Advances in Neural Information Processing Systems, pp. 945–951 (1996)
5. Clausen, T.H., Jacquet, P.: Optimized Link State Routing Protocol (OLSR). RFC 3626, October 2003. https://doi.org/10.17487/RFC3626. https://rfc-editor.org/rfc/rfc3626.txt
6. Gupta, N., et al.: Improved route selection approaches using Q-learning framework for 2D NoCs. In: Proceedings of the 3rd International Workshop on Many-core Embedded Systems, MES 2015, Portland, OR, USA, pp. 33–40. ACM, New York (2015). https://doi.org/10.1145/2768177.2768180
7. Hoceini, S., Mellouk, A., Amirat, Y.: K-shortest paths Q-routing: a new QoS routing algorithm in telecommunication networks. In: Lorenz, P., Dini, P. (eds.) ICN 2005. LNCS, vol. 3421, pp. 164–172. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31957-3_21
8. Kumar, S.: Dual reinforcement Q-routing: an on-line adaptive routing algorithm. In: 1997 Proceedings of the Artificial Neural Networks in Engineering Conference (1997)
9. Peshkin, L., Savova, V.: Reinforcement learning for adaptive routing. In: Proceedings of the 2002 International Joint Conference on Neural Networks, IJCNN 2002 (Cat. No. 02CH37290), vol. 2, pp. 1825–1830, May 2002. https://doi.org/10.1109/IJCNN.2002.1007796
10. Santhi, G., Nachiappan, A., Ibrahime, M.Z., Raghunadhane, R., Favas, M.K.: Q-learning based adaptive QoS routing protocol for MANETs. In: 2011 International Conference on Recent Trends in Information Technology (ICRTIT), pp. 1233–1238, June 2011. https://doi.org/10.1109/ICRTIT.2011.5972411
11. Schooenderwoerd, R., Holland, O., Bruten, J., Rosenkrantz, L.: Ants for load balancing in telecommunication networks. Technical report 96-35, HP Labs, Bristol (1996)
12. Watkins, C.J.C.H., Dayan, P.: Q-learning. Mach. Learn. **8**(3), 279–292 (1992). https://doi.org/10.1007/BF00992698

13. Xia, B., Wahab, M.H., Yang, Y., Fan, Z., Sooriyabandara, M.: Reinforcement learning based spectrum-aware routing in multi-hop cognitive radio networks. In: 2009 4th International Conference on Cognitive Radio Oriented Wireless Networks and Communications, pp. 1–5, June 2009. https://doi.org/10.1109/CROWNCOM.2009.5189189
14. Yap, S.T., Othman, M.: An adaptive routing algorithm: enhanced confidence-based Q routing algorithm in network traffic. Malays. J. Comput. Sci. **17**(2), 21–29 (2004)