Comparison of the Q-Routing and Shortest Path Routing Algorithms.

F. Tekiner, Z. Ghassemlooy and T.R. Srikanth
School of Engineering & Technology, Northumbria University, Newcastle upon Tyne,UK.
Email: ftekiner@ieee.org

Abstract- In this paper, we compare the self-adaptive Q-Routing and dual reinforcement Q-Routing algorithms with the conventional shortest path routing algorithm. The Q-Routing algorithm embeds a learning policy at every node to adapt itself to the changing network conditions, which leads to a synchronised routing information, in order to achieve a shortest delivery time. Unlike Q-Routing, the shortest path routing algorithm routes the packets based on the link with the least delay irrespective of the traffic pattern on the link. Here simulations are carried out under problematic conditions by taking into account the node and link failures. Results show that the adaptive (learning) approach performed better than the traditional non-adaptive approach under problematic conditions.

Keywords: Routing algorithm, Q-Routing, dual reinforcement Q-Routing, shortest path routing.

I. INTRODUCTION

In today's fast growing Internet, traffic conditions changes and failures occur at some parts of the network from time-to-time, in unpredictable manner. Therefore, there is a need for an algorithm to manage traffic flows and deliver packets from sources to the destination in a reasonable time.

The process of transmitting packets from its source node *s* to its destination node *d* in a network is called *routing*. Usually packets can make many hops at the intermediate nodes on its way to its destination. At each node a packet is received, stored and then routed to the next hop until it reaches its destination.

Routing algorithms are classified according to their adaptability to the changing traffic patterns in the network as dynamic (adaptive) and static (non-adaptive) algorithms [1]. In case of non-adaptive algorithms, the routing decision is based on local information; hence, routing problem is distributed over the network. On the other hand in non-adaptive algorithms there is a need for a global knowledge which makes it hrder to adapt to the changes in the network.

Shortest path routing algorithm [2] is one such widely deployed non-adaptive routing algorithm that routes a packet based on Dijkastra's well-known shortest path algorithm. In this approach, packets arriving at a node take the same shortest path to a particular destination on the network. The main drawback of this approach is that it does not take into consideration the changes in the network's traffic dynamically. For example, queuing results in congestion in the intermediate nodes that results in packets being delayed before reaching the destination. In such cases, it would be advantageous to forward packets to alternative routes that may not be the shortest path, but can eventually result in shorter delivery time.

The routing algorithms that are in use lacks intelligence, they need human assistance and interpretation in order to adapt themselves to the failures and changes. However, introducing intelligence not necessarily improves the performance of the algorithm but it improves the adaptability to the changes in the network.

In the recent years, agent based systems and reinforcement learning have been widely applied to routing problems. This is due to the fact that these methods do not need any supervision and they are distributed in nature. Swarm intelligence particularly ant based systems; Q-learning methods and hybrid agent based distance vector routing algorithms have shown promising and encouraging results.

Here, the focus is on the Q-learning, which is a form of reinforcement learning. Q-Routing [3] is an adaptive routing algorithm that routes packets based on the learnt routing information from its neighbours. Initially this algorithm builds a routing table based on the delivery times[1] (Q values) of the packets to every node in the network. These delivery times are updated every time a node forwards a packet for a particular destination, changes depending on the traffic at a given time to the destination. Based on this information, a node can choose an alternative route when the queues are congested in the intermediate nodes, thus resulting in faster delivery, unlike shortest path algorithm. In this type of learning pattern, routing information is synchronised on all nodes in the network. Bandwidth utilisation of this algorithm

[1]The delivery time for a packet is the time difference of the time the packet is introduced in the network and the time it reached its destination.

is very low, as it needs a creation and transfer of a learning packet for every data packet forwarded.

Dual reinforcement Q-Routing (DRQR) [4] is a modified version of the Q-Routing algorithm, where learning occurs in both ways. Since, the learning process occurs in both ways the learning performance of the Q-Routing algorithm doubles. However, it adds more overheads to the network.

This paper investigates the performance of all three algorithms by simulating them on 14 nodes NFSNET. Their throughput and packet delay analysis will be discussed in section 5.

II. SHORTEST PATH ROUTING ALGORITHM

Shortest path algorithm also known as the Dijkstra's algorithm, is one of the most widely deployed routing algorithms in today's internet works, OSPF [5]. Shortest path routing (SPR) algorithm computes the routes based on the cost metric, which is expressed in different metrics by the network administrators as a function of delay, data rate, financial cost etc.

Every node in the network computes its routes to every other node in the network. The shortest path algorithm can be best explained by considering the network as a directed graph, where, every node is treated as an edge in the digraph and every link as the vertex connecting any two edges [6]. The cost of these vertices should always be non-negative.

Let V be a set of all the edges of the digraph and $c[s,w]$ be the array of the cost metrics of the vertices and $D[w]$ be the array of the least costs (of the shortest path) to any node to be computed. Initially every edge (node), before computing its routes, sets the cost of the shortest path to any other edge as infinity *i.e.* $D[w] = \infty$. If there are n edges in the digraph, every edge s starts computing the costs $D[w]$ of the shortest paths to other nodes by choosing its neighbour $w \in (V\text{-}s)$ with minimum $c[s,w]$ value. Then, it computes the routes for every edge $v \in (V\text{-}s\text{-}w)$, which is a neighbour of w, such that $D[v] = \min(D[v], D[w]+c[w,v])$, where $c[w,v]$ are the costs of the vertices connected to w. This process is continuous until all the edges of the digraph are visited. When the algorithm is stopped all the nodes in the network should have discovered the shortest path to the every other node in the network [1].

III. Q-ROUTING ALGORITHM

Q-Learning is a model free algorithm that learns from the delayed reinforcements and it is one of the easiest approaches to implement in reinforcement learning, thus one of the most popular [7]. Q-learning is applied to the routing problem in Q-Routing algorithm, where routing table in the distance vector algorithm [7] is replaced by the table of estimations (Q values) based on the link delay. In the Q-Routing same routing policies are used as in the distance vector routing algorithm. Q-function is used to update routing table entries in Q-Routing [3]. However, it has been suggested that neural networks can be used for approximating the Q-function by incorporating diverse parameters of the network such as time delays, queue lengths and etc [3].

In Q-Routing, when x sends a packet to the node d via its neighbour y it receives y's estimated remaining trip times to the destination d. Then it selects the neighbour with the smallest time (equation 1). Q-Routing uses forward exploration to update the Q values in the routing table that represents the delivery time estimation to the given destination (equation 2).

$$Q_y(z',d) = \min_{z \varepsilon N(y)} Q_y(z,d) \tag{1}$$

$$\Delta Q_x(y,d) = \eta_f (Q_y(z',d) + q_y - Q_x(y,d)) \tag{2}$$

$\Delta Q_x(y,d)$ is the new estimation value for node x to destination d via the neighbour node y (fig. 1). This new estimation is calculated by subtracting old estimation value $Q_x(y,d)$ from the sum of the estimation time for packet travelling from node y to destination d via neighbour z ($Q_y(\acute{z},d)$) and current queue delay for the packet in node x (q_x). η_f[2] is the learning rate parameter defined by the programmer. In fig. 1, x updates its $Q_x(y,d)$ value pertaining to the remaining path of packet p via node y.

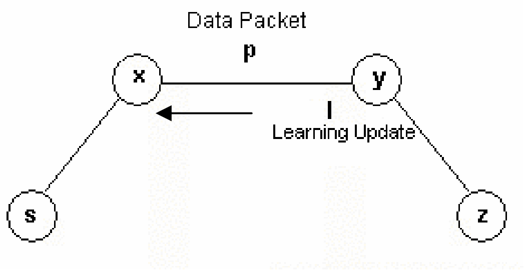

Figure 1: Forward exploration

[2] η_f is set to 0.7 in original Q-routing algorithm.

IV. DUAL REINFORCEMENT Q-ROUTING ALGORITHM

Backward exploration together with the forward exploration is applied in DRQR algorithm in order to improve the learning rate of the Q-Routing algorithm. In DRQR, exact delay values learnt from the backward learning have also been used in the routing tables in addition to the estimation values learnt from forward learning in Q-Routing. This has doubled the learning information available in the algorithm thus improved the learning rate of the algorithm.

In DRQR algorithm, when *x* sends a packet to node y to get its estimated remaining trip times, y also gets *x*'s estimated trip times for its link with *s*.

$$Q_x(z',s) = \min_{z \varepsilon N(y)} Q_y(z,s) \tag{3}$$

$$\Delta Q_y(y,s) = \eta_b(Q_x(z',s) + q_y - Q_y(x,s)) \tag{4}$$

In fig. 2, packet at node *x* arriving from source node s is sent to node y, also carries the estimated time that it takes from node *x* to *s*, $Q_x(\acute{z},s)$ (equation 3). With this information node *y* updates its own estimate $Q_y(x,s)$ for the entry node *x* associated with the destination *s* (equation 4). Therefore, in DRQR both backward and forward exploration can be used to update the Q entries. However, this adds an overhead to the packet and to the algorithm.

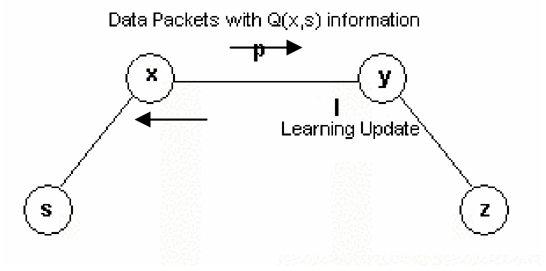

Figure 2: Forward and backward exploration.

In fig. 2, during the forward exploration, the node sending the data packet (node x) updates its $Q_x(y,d)$ value pertaining to the remaining path of packet *p* via node *y.* However, during the backward exploration, the receiving node *y* updates its $Q_y(x,s)$ value pertaining to the traversed path of packet *p* via node *x.* Main drawback of the both Q-Routing and DRQR approaches is that, although they are capable of detecting the link and node failures, they are incapable of restoring them.

V. IMPLEMENTATION AND EXPERIMENTS

SPR, Q-Routing and DRQR algorithms were implemented in the following environment:

- All of the algorithms are implemented in the C language in a parallel environment by using PVM under Linux OS.
- Assigning every node to a different process both on the same and distributed machines simulates parallel behaviour.
- Random uniform traffic distribution is used with varying packet creation rates.
- NSFNET (fig. 3) is used as the network topology with each link having equal cost.
- For each simulation, packet generation is stopped after creating 5000 packets per node and simulation is stopped after all packets are arrived to their destinations or detected and deleted from the network.
- Every simulation is run 3 times and the average of the results is used for accuracy.
- A random link fails every 5 seconds for a period of 4 seconds and traffic is directed to other neighbours.
- It is assumed that there is no packet loss.
- Packet payload is fixed (512 Kbytes).
- In OSPF, all the nodes are assumed to be in the same area (area 0) or no hierarchal routing is considered.
- The HELLO packets are exchanged between the nodes every 5 seconds to know whether its neighbouring nodes are active.
- The dead interval taken as 4 times the HELLO interval, which is the time duration the node decides if its neighbouring node is dead if doesn't get a HELLO packet from it [5].

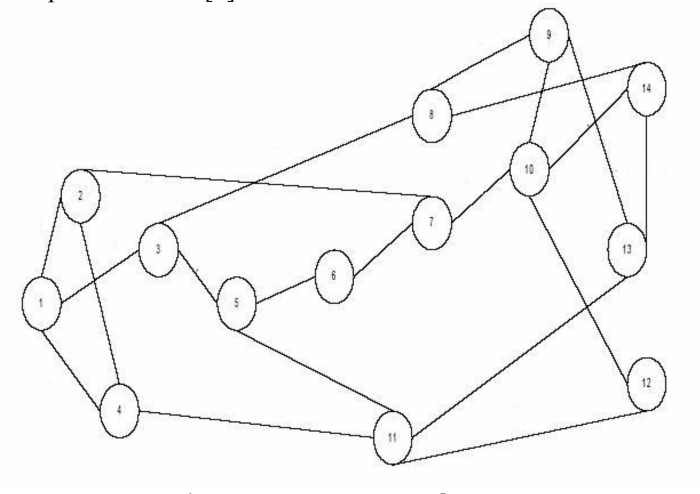

Figure 3: NFSNET topology.

The performance parameters for the simulations are:

i. **Average packet delay:** is the average delay a packet experiences while being routed from source to destination.
ii. **Average throughput per packet:** is the average number of packets being forwarded by a node for the duration of the simulation.

During the simulations, FIFO Queues is used in the input and output buffers. A random link failure is introduced artificially in every 5 seconds in order to test the behaviour of the algorithm in problematic conditions. On detection of a failure SPR recomputed all its routes, whereas Q-Routing and DRQR chose an alternative route apart from the failed one. The failed link is restored after 4 seconds, while the shortest path algorithm recomputed its routes. However, Q-Routing and DRQR fail to restore the original learning policy for that link and recomputed it once the link is recovered.

VI. RESULTS AND DISCUSSIONS

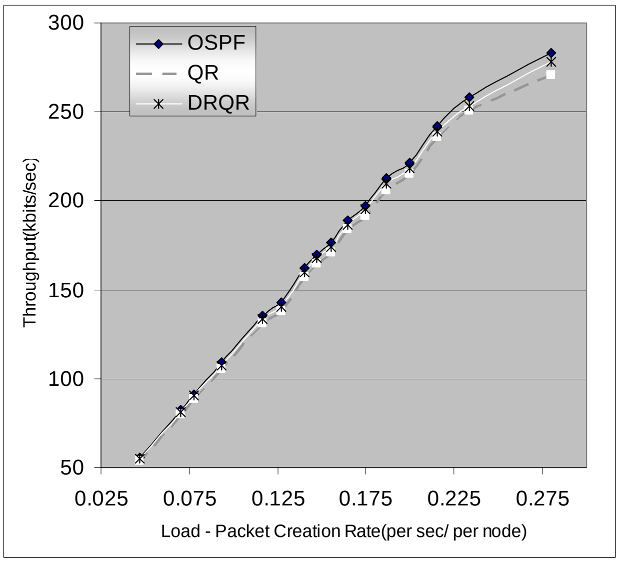

Figure4: Average packet delay versus system load (packet creation rate).

From fig. 4, it can be seen that as the system load increases average packet delay increases for all algorithms. Therefore, for Q-Routing algorithm, the delivery times are lower compared to the SPR and DRQR algorithms as it was expected. For example, when packet creation rate is around 0.3s Q-Routing algorithm delivers packets ~0.2s faster than others. This is because, at low loads the SPR algorithm ignores the bottlenecks and the failures in the network and floods the packets to the same shortest paths. However, Q-Routing and DRQR algorithms learn routing policy faster and routes packets over alternative routes to avoid congestion. However this deviation in the delivery time value remains low.

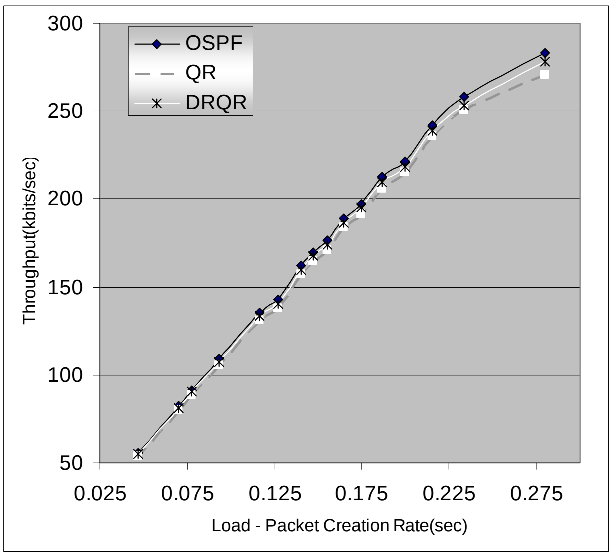

Figure5: Throughput versus system load (packet creation rate).

For the DRQR algorithm, the delivery time at high loads was reasonably high compared to Q-Routing, which can be accounted for the packet overhead, which makes the packet to carry the learning update along with the data. However, fig. 5 shows that in SPR throughput is higher compared to other two, as utilisation of the network is lower on Q-Routing and DRQR because of the learning messages.

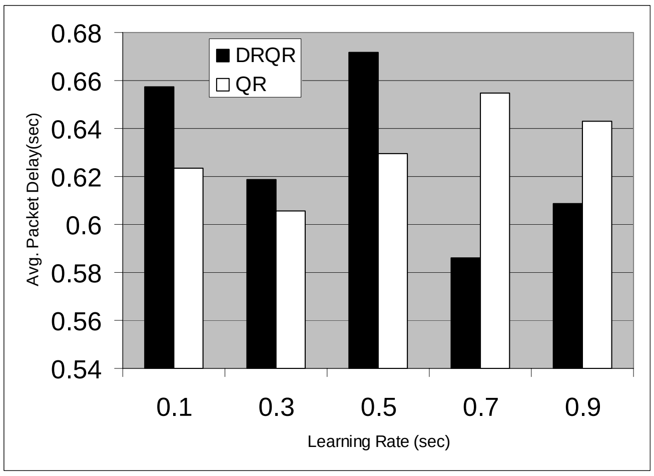

Figure6: Average packet delay vs. learning rate.

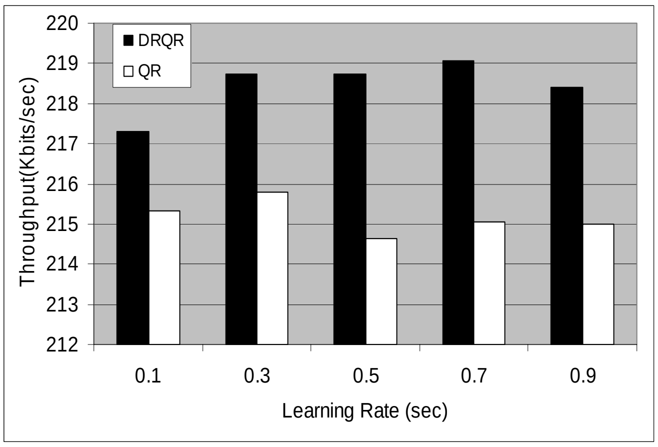

Figure7: Throughput vs. learning rate

Learning rates of Q-routing and DRQR for particular packet creation duration (0.175) has been varied and shown in fig. 6 and fig. 7. It is shown that the average delivery time of packets for Q-Routing remains low at 0.7 (fig. 6), while the throughput remains high at 0.7 for DRQR (fig. 6). However, these figures are only valid for our test environment and effect of learning rate needs to be investigated in the future for different systems for a more general figure rather than using constant values based on experience.

VII. CONCLUSION

Q-Routing does not always guarantees finding the shortest path. Moreover they can only find a single path, but cannot explore multiple paths. In the Q-Routing, routing is proportional to the data traffic. Therefore network emerges lower in low data traffic, hence routing algorithm works slower. This is not a desired property as a routing algorithm should perform the same under all traffic conditions. At the same time the shortest path algorithms also ignores the bottleneck as the network traffic increases. However, the need for a reasonable and efficient algorithm, which is consistent and adaptive, is as strong as ever.

REFERENCES

[1] Tanenbaum, A., “Computer Networks”, Prentice Hall, Fourth edition, Chp. 5., 2003.

[2] Dijkstra, E.W., “A Note on Two Problems in Connexion with Graphs”, Numerische Amthematik 1: 267-271, 1959.

[3] Boyan, J. A., Littman, M.L., “Packet Routing in Dynamically Changing network: A Reinforcement Learning Approach”, Advances in Neural Information Processing Systems 6. MIT Press, Cambridge, MA, 1994.

[4] Kumar, S., Miikkulainen, R., "Dual Reinforcement Q-Routing: An Online Adaptive Routing Algorithm", Artificial Neural Networks in Engineering, 1997.

[5] John T.Moy, "OSPF Version 2", IETF 2328.6.Bellman, R. E, "On a Routing Problem", Quarterly of Applied Mathematics 16 (1978), 315-333, 1998.

[6] D. P. Bertsekas and R. Gallager,Data networks, Prentice Hall, ISBN: 0132016745, Chp. 5., 1991

[7] R. S. Sutton and A. G. Barto, “Reinforcement learning: an introduction”, MIT Press, Chp.1-3, 1994.