Confidence Based Dual Reinforcement Q-Routing: An adaptive online network routing algorithm

Shailesh Kumar
Dept. of Elec. and Computer Engineering
The University of Texas at Austin
Austin, TX 78712 USA
stumarScs.utexas.edu

Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 USA
risto@cs.utexas.edu

Abstract

This paper describes and evaluates the Confidence-based Dual Reinforcement Q-Routing algorithm (CDRQ-Routing) for adaptive packet routing in communication networks. CDRQ-Routing is based on the Q-learning framework of Q-Routing. The main contribution of this work is the increased quantity and improved quality of exploration in CDRQ-Routing, which lead to faster adaptation and better routing policies learned as compared to Q-Routing, the state-of-the-art adaptive Bellman-Ford Routing, and the non-adaptive shortest path routing. Experiments over several network topologies have shown that at different loads, CDRQ-Routing learns superior policies significantly faster than Q-Routing. Moreover, CDRQ-Routing learns policies that sustain higher load levels than Q-Routing. Analysis shows that overhead due to exploration is insignificant as eqmpared to the improvements in CDRQ-Routing.

1 Introduction

In a communication network information is transferred from one node to another as data packets [Tanenbaum, 1989]. The process of sending a packet P(s, d) from its source node s to its destination node *d is* referred to as *packet routing* [Bellman, 1958]. Normally this packet takes multiple "hops" and on its way, spends some time waiting in the queues of intermediate nodes, while they are busy processing the packets that came earlier. Thus the delivery time of the packet, defined as the time it takes for the packet to reach its destination, depends mainly on the total time it has to spend in the queues of the intermediate nodes. Normally, there are multiple routes that a packet could take, which means that the choice of the route is crucial to the delivery time of the packet for any (s,d) pair. If there was a global observer with current information about the queues of all nodes in the network, it would be possible to make *optimal* routing decisions: always send the packet through the route that has the shortest delivery time at the moment. In the real world, such complete, global information is not available, and the performance of the global observer is an upper bound on actual performance. Instead, the task of making routing decisions has to be shared by all the nodes, each using only local information. Thus, a routing policy is a collection of local decisions at the individual nodes. When a node *x* receives a packet P(s, *d)* originating at node *s* and destined for node d, it has to choose one of its neighboring nodes *y* such that the packet reaches its destination as quickly as possible.

The simplest policy is the shortest-path algorithm, which always routes packets through the path with the minimum number of hops. This policy is not always good because some intermediate nodes, falling in a popular route, might have large queues. In such cases it would be better to send the packet through another route that may be longer in terms of hops but results in shorter delivery time. Hence as the traffic builds up at some popular routes, alternative routes must be chosen to keep the average packet delivery time low. This is the key motivation for adaptive packet routing strategies that learn alternate routes through exploration as the current routing policy begins to lead to degraded performance.

Learning effective routing policies is a challenging task for several reasons [Kumar, 1998]. *(i)* The goal is to optimize a global metric, the *average packet delivery time of all packets,* using local information, *(ii)* There is no ''training signal" available for directly evaluating a routing policy until the packets have reached their destination. *(iii)* When a packet reaches its destination, such a training signal could be generated, but to make it available to the nodes responsible for routing the packet, the training signal would have to travel to all these nodes, consuming a lot of network resources, *(iv)* Finally, it is not known which particular decision in the sequence of routing decisions deserves credit for the performance, and how much (the credit assignment problem). Thus, a way of efficiently exploring the network environment and continually updating the decision makers based on the local information is necessary in order to learn good routing policies.

Bellman-Ford Routing [Bellman, 1957] (BF) is by far the mast widely used distance vector adaptive routing algorithm. In BF, each node has two tables which con-

tain, for each possible destination, (1) a cost *(cost table)* or minimum delivery time for sending a packet to that destination and (2) the node's neighbor *{routing table)* to which the packet should be forwarded to reach the destination for the corresponding cost. Neighboring nodes exchange their cost tables frequently for adaptation. The drawback being an enormous overhead of exploration (exchange of routing information between nodes) and a slow rate of learning. Q-Routing [Boyan and Littman, 1994; Littman and Boyan, 1993] uses the Q-learning framework [Watkins and Dayan, 1989] for this task. Each node makes its routing decisions based on the local routing information, which is a table of Q-values that estimate the quality of the alternative routes. These values are updated each time the node sends a packet to one of its neighbors. This way, as the node routes packets, its Q-values gradually incorporate more global information. Such exploration has been shown capable erf adapting to load changes and to perform better than the non-adaptive shortest-path routing with high loads.

This paper presents a new adaptive routing algorithm called Confidence-based Dual Reinforcement Q-Routing (CDRQ-Routing; [Kumar, 1998]) that improves both the *quality* and *quantity* of exploration of Q-Routing. The *quality* of exploration is improved by associating confidence values (between 0-1) with each of the Q-values in the network. These values represent how reliably the corresponding Q values represent the state of the network. The amount of adaptation for a Q-value, in other words the learning rate, depends on the confidence values of the new and the old Q-values (whereas in Q-Routing a fixed learning rate is used for all updates). This component of CDRQ-Routing is called Confidence based-Q-Routing (CQ-Routing; [Kumar and Miikkulainen, 1998]). The *quality* of exploration is increased by including backward exploration (whereas in Q-Routing only forward exploration is used). As a result, with each packet hop, two Q-value updates take place, one due to forward exploration and the other due to backward exploration. This component of CDRQ-Routing is called the Dual Reinforcement Q-Routing (DRQ-Routing; [Kumar and Miikkulainen, 1997]). Essentially, DRQ-Routing combines Q-routing with *dual reinforcement learning,* which was first developed for a satellite communication problem, where the two ends of the communication system co-adapt using the reinforcement signal for the other end [Goetz *et al.,* 1996]. CDRQ-Routing balances the complementary and independent improvements due to both these components into one algorithm. The Q-Routing algorithm is described in section 2, followed by the CDRQ-Routing in section 3. The performance of the two algorithms are evaluated experimentally in section 4 and compared to the standard shortest-path algorithm. The amount of overhead generated by these algorithms is analyzed in section 5, and a number of directions for future research outlined.

2 Q-Routing

In Q-routing, the routing decision maker at each node *x* makes use of a table of values $\mathbf{Q_x(y,d)}$, where each value is an estimate, of how long it takes for a packet to be delivered to node d, if sent via neighbor y, excluding time spent in node *x's* queue. When the node has to make a routing decision it simply chooses the neighbor *y* for which $\mathbf{Q_x(y,d)}$ is minimum. Learning takes place by updating the Q values. On sending P(s,d) to y, *x* immediately gets back y's estimate for the time remaining in the trip, $\mathbf{Q_y(\hat{z},d) = \min_{z \in N(y)} Q_y(z,d)}$, where N(n) denotes the set of neighbors of node n. Node *x* can revise $Q_x(y,d)$ based on this feedback and the queue length q_y of y, using a learning rate n_f;

$$\Delta Q_x(y,d) = \eta_f(\overbrace{Q_y(\hat{z},d) + q_y}^{\text{new estimate}} - \overbrace{Q_x(y,d)}^{\text{old value}}), \quad (1)$$

In other words, the information about the remaining path is used to update the Q value of the sending node. Such exploration can be termed *forward exploration* (figure 1).

3 Confidence-based Dual Reinforcement Q-Routing

Both the quantity and quality of Q-Routing are improved in CDRQ-Routing using two components, CQ-Routing and the DRQ-Routing, respectively.

3.1 CQ-Routing

When a Q-value is not updated for a long time, its reliability in representing the true state of the network goes down. In Q-Routing there is no way of quantifying the reliability of a Q-value. Moreover, in equation (1) the learning rate is constant for all updates, although it should depend on how reliable the updated $(\mathbf{Q_x(y,d)})$ and estimated $(\mathbf{Q_y(\hat{z},d)})$ Q-values are. These issues are addressed in CQ-Routing. In CQ-Routing, the accuracy, or reliability, of each Q-value $\mathbf{Q_x(y,d)}$ is quantified by an associated *confidence value* (C-value), $\mathbf{C_x(y,d)} \in$ [0,1]. $\mathbf{C_x(y,d)}$ close to 1 indicates that Qa:(y,d) represents the network state accurately, while $\mathbf{C_x(y,d)}$ dose to 0 indicates that $\mathbf{Q_x(y,d)}$ is almost random. The base case C-values corresponding to the base case Q-value, $Q^*(y,y) = S$, are $C.(y,y) = 1$, $\forall y \in N(x)$ and $V^* \in V$, the set of all nodes in the network. The C-values corresponding to all the other Q-values, which are initialized randomly, are initially set to 0.

In CQ-Routing, the learning rate depends on the C-values of the Q-values that are part of the update. More specifically, when node *x* sends a packet *P(s, d)* to its neighbor y, it gets back not only the best estimate of node *y* for the remaining part of P(s,d)'s journey, namely $\mathbf{Q_y(\hat{z},d)}$, but also the confidence value associated with this Q value, namely, $\mathbf{C_y(\hat{z},d)}$. Now when node *x* updates its $Q_s(y,d)$ value, it first computes the learning rate $\eta_f = \eta(\mathbf{C_{old}, C_{est}})$ which depends on both $\mathbf{C_x(y,d)}$

$(=C_{old})$ and $C_y(\hat{z}, d)$ $(=C_{est})$. The *learning rate function*, $\eta(C_{old}, C_{est})$ is chosen such that it is high if either (or both) C_{old} is low or C_{new} is high:

$$\eta(C_{old}, C_{est}) = \max(C_{est}, 1 - C_{old}) \in [0,1]. \quad (2)$$

The C-values are themselves updated such that (1) If a Q-value is not updated in the last time step, then its C-value decays with a decay constant $\lambda \in (0,1)$ and (2) if a Q-value is updated in the last time step, then its C-value is updated based on C_{old} and C_{est}:

$$C_{upd} = \begin{cases} \lambda C_{old} & \textit{(Q-value not updated)}, \\ C_{old} + \eta(C_{est} - C_{old}) & \textit{(Q-value updated)} \end{cases} \quad (3)$$

3.2 DRQ-Routing

Exploration of Q-values is done locally between neighboring nodes during a packet hop to avoid excessive exploration overhead. In Q-Routing, only one Q-value $(Q_x(y,d))$ is updated when a packet *P(s,d)* hops from *x* to *y* (forward exploration), however, one more Q-value $(Q_y(x,s))$ can also be updated in the same hop. This idea of using information about the traversed path for exploration in the reverse direction is called *backward exploration* [Kumar, 1998] (figure 1) and is derived from Dual Reinforcement Learning, which was first developed for adaptive signal predistorters in satellite communications [Goetz *et al*., 1996]. DRQ-Routing incorporates backward exploration into the Q-Routing algorithm. When node *x* sends a packet *P(\$, d)* to its neighbor *y,* the packet can take along Q-value information of node x, which can be used by *y* for updating its own estimate pertaining to *x.* Later when node *y* has to make a decision, it has the updated Q-value for *x.* The only exploration overhead is a slight increase in the size of the packets. More specifically, *P(s,* d), currently at node x, carries to node *y* the estimated time it takes for a packet destined for node *s* from node x_f that is $Q_x(\hat{z}, s) = \min_{x \in N(x)} Q_x(z, s)$. With this information, node *y* can update its estimate $Q_y(x, s)$, of sending a packet to node *s* via its neighbor x, using the queue length q_x of node *x* and a learning rate n_b:

$$\Delta Q_y(x,s) = \eta_b(\overbrace{Q_x(\hat{z},s) + q_x}^{\textit{new estimate}} - \overbrace{Q_y(x,s)}^{\textit{old value}}) \quad (4)$$

In other words, the information about the path the packet has traversed so far is used to update the Q value of the receiving node. This way the packet is used to carry routing information from one node to the next as it moves from source to destination. In DRQ-Routing, both forward exploration and backward exploration are used to update two Q-values in each hop. Figure 1 illustrates these two updates as the packet P(s, *d)* hops from node *x* to its neighbor *y.*

3.3 CDRQ-Routing

CDRQ-Routing combines both the CQ-Routing and DRQ-Routing components. At each hop of packet

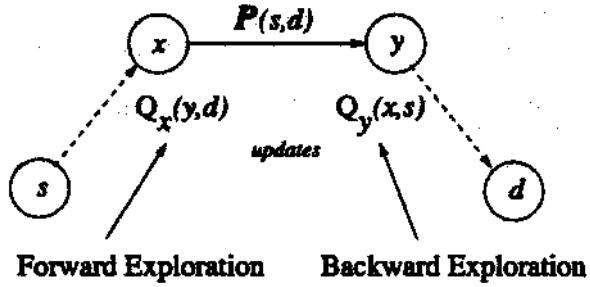

Figure 1: Forward and Backward Exploration.

$P(s,d)$ from node x to y, relevant Q-values, $Q_x(y,d)$ and $Q_y(x,s)$, are updated according to equations (1) and (4), respectively. The corresponding C-values $C_x(y,d)$ and $C_y(x,s)$ are updated using (3). The learning rates for these updates, η_f and η_b, are computed using (2). In both (2) and (3), C_{old} and C_{est} are chosen appropriately.

4 Experiments

The experiments described in this paper are based on a simulated communication network. Packets destined for random nodes are introduced into this network at random nodes. The number of packets introduced per unit simulation time step (pkt/sim-time) is called the *network load.* Multiple packets at a node are stored in its *unbounded* FIFO queue. In one time step, each node removes the packet in front of its queue, examines the destination of this packet and uses its routing decision maker to send the packet to one of its neighboring nodes. The *delivery time* of a packet is defined as the time between its introduction at the source node and its removal at the destination node. Delivery time is measured in terms of simulation time steps. Average packet delivery time, computed at regular intervals (50 time steps in the current experiments), is the average of all the packets arriving at their destinations during the last interval. This measure is used to monitor the network performance *while* learning is taking place. Average packet delivery time *after* learning has settled measures the quality of the final routing policy.

The performance of CDRQ-Routing was tested against Q-Routing, Bellman-Ford routing and non-adaptive shortest-path routing, on a number of network topologies including 128 node 7D-hypercube, 116-node LATA telephone network, and an irregular 6 x 6 grid (due to [Boyan and Littman, 1994; Littman and Boyan, 1993] and shown in figure 2). The results were similar in all cases; the discussion below focuses on the last one since it best illustrates adaptation. In this network there are two ways of routing packets between the left cluster (nodes 1 through 18) and the right cluster (nodes 25 through 36): the route including nodes 12 and 25 *{R\)* and the route including nodes 18 and 19 *(Rz).*

The shortest-path routing algorithm, which chooses the route with minimum hops, routes most of the traffic between the left cluster and the right cluster via route

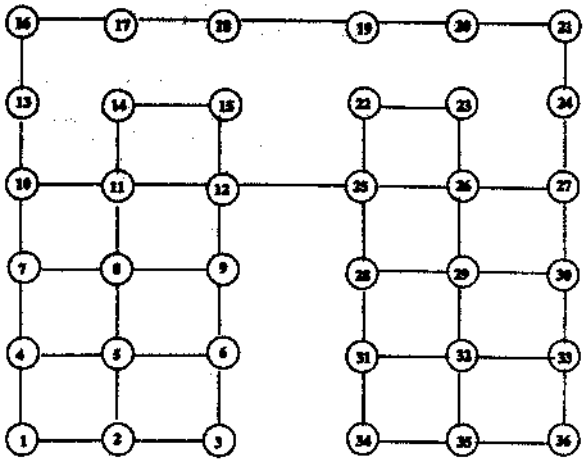

Figure 2: The 6 x 6 Irregular Grid.

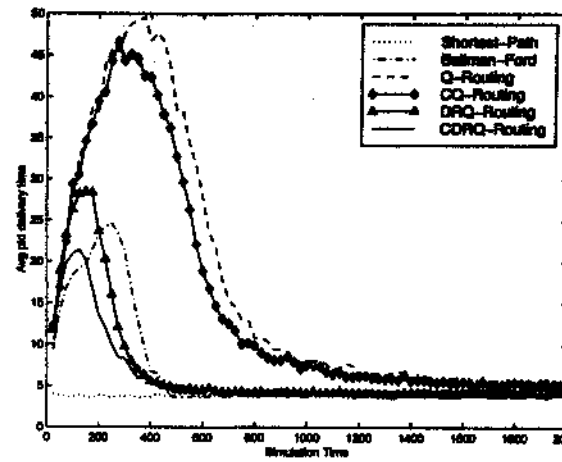

Figure 3: Learning at low load

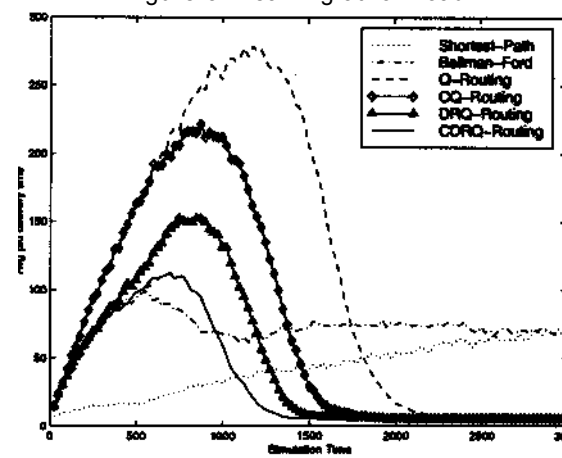

Figure 4: Learning at medium load

R_1 For low loads (e.g. 1.25 pkt/sim-time), this routing policy works fine and throughout the simulation, the average packet delivery time is low (figure 3). Bellman-Ford shows a small learning period after which it also converges to close-to-optimal routing policy by around 400 sim-time. At medium loads (such as 2.25 pkt/sim-time), the shortest-path strategy breaks down as nodes 12 and 25 become flooded with packets. The average delivery time increases linearly with simulation time (figure 4). Bellman-Ford learns a stable but inferior routing policy at medium load. At high loads (such as 3.0 pkt/sim-time) the flooding of node 12 and 25 takes place at an even faster rate. Bellman-Ford breaks down at high loads and shows similar trends as shown by shortest path at low loads. Thus Bellman-Ford is good only at low loads but its performance degrades as load increases.

The main result, however, is the comparison between Q-Routing and CDRQ-Routing. To demonstrate the independent and complimentary contributions of the components of CDRQ-Routing, the performance of CQ-Routing and DRQ-Routing are also shown. The Q-tables in Q-Routing and CDRQ-Routing at all nodes were initialized to low random values. In Q-Routing and DRQ-Routing, the learning rate for forward exploration, n_f, was set at 0.85, while that of backward exploration in DRQ-Routing, n_b was set at 0.95. In CQ-Routing and CDRQ-routing, the learning rates are computed using the confidence values of equation (2). The decay constants A in CQ-Routing and CDRQ-Routing were set at 0.95 and 0.90 respectively. These learning rates and decay constants were found experimentally to give the best performance.

The results shown in figures 2,3 and 4 for low, medium and high loads, are averages of 50 test runs, each time with different random start. Statistical significance was computed using standard t-test [Press *et al.*, 1995] at 99% confidence. During the first few hundred time steps the average packet delivery times are small because the packets destined for distant nodes have not yet reached their destinations, and statistics are available only for packets destined for nearby nodes (with small delivery times). As distant packets start arriving, the average packet delivery time increases, while learning is still in progress. Eventually the learning converges, and each of the curves settles down indicating a stable routing policy.

At low loads, CDRQ-Routing learns an effective routing policy almost three times faster than Q-Routing (figure 3). CQ-Routing is only slightly better than Q-Routing while DRQ-Routing is significantly better. This is because at low loads the number of packets in the network and consequently the number of Q-value updates is low. Hence increasing the amount of exploration *per packet hop* has a significant effect on the speed of learning. At medium loads, CDRQ-Routing learns a more effective policy nearly twice as fast as Q-Routing (figure 4). CQ-Routing and DRQ-Routing, are both significantly better than Q-Routing throughout the learning phase (200 through 2000 time steps). CDRQ-Routing is better than CQ-Routing and DRQ-Routing, which shows that both components of CDRQ-Routing contribute indepen* dently and in complementary ways to CDRQ-Routing. At high load levels, CDRQ-Routing converges to a rout-

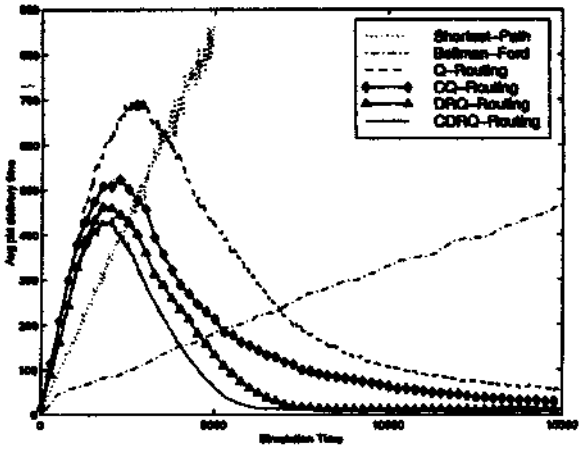

Figure 5: Learning at high load

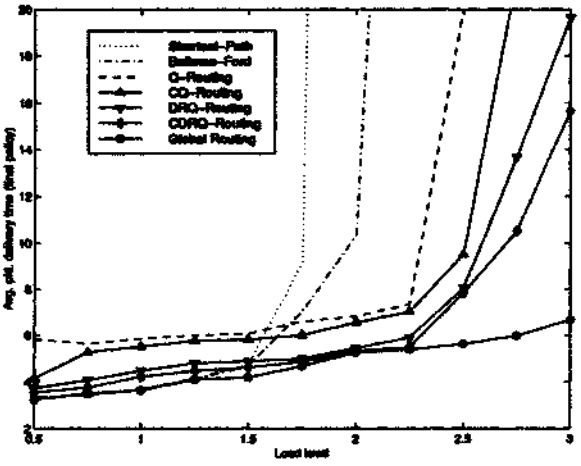

Figure 6: Average packet delivery times at different network loads.

ing policy which is more than twice as good, in terms of average packet delivery time, as the policy to which Q-Routing converges (figure 5). Again, CDRQ-Routing learns faster than both its components.

The results are summarized in figure 6, which shows the average packet delivery times at different load levels after the learning has converged. This measures the quality of the final policy in terms of how much load it can sustain. The plots are averages over 10 simulations. The performance of the "global" routing policy with complete information about all nodes (section 1) is also shown as a benchmark. Shortest path routing breaks down as load increases beyond 1.5 pkts/sim-time due to excessive traffic buildup along R_1. Bellman-Ford can sustain a little higher load levels but breaks down at around 2 pkt/shn-time. CDRQ-Routing learns significantly better policies than Q-Routing at all loads. Moreover, the performance of CDRQ-Routing is close to the global routing until 2.25pkts/sim-time and it can sustain load levels up to 3.0 pkt/sim-time while Q-Routing breaks down after 2.25 pkt/sim-time.

5 Discussion and Future Work

Exploration makes it possible for a routing algorithm to adapt. In this paper, Q-Routing was compared with CDRQ-Routing, which uses both forward and backward exploration, together with confidence values for exploration. CDRQ-Routing has more exploration per packet hop and the quality of exploration is also higher. As a result, CDRQ-Routing learns better routing policies than Q-Routing significantly faster. Moreover, compared with Bellman-Ford routing, CDRQ-Routing is much superior in terms of speed of convergence, quality of the routing policy learned, load level sustenance and amount of exploration overhead, making them more practical for use in communication networks.

Exploration adds overhead into the routing algorithm. It is important to analyze the tradeoff between the improvements and the overhead incurred. In Bellman-Ford, exploration overhead comprises of frequent exchanges of cost tables (which are as large as the number of nodes in the network) between every pair of neighboring nodes. This leads to prohibitive overhead and is a serious drawback for Bellman-Ford. The Q-Routing and CDRQ-Routing, on the other hand do not suffer from this drawback. In forward exploration, when a node *y* receives a packet from node x, it sends back an estimate and a confidence value to node x. The estimate does not enter node x's queue, but instead node *x* waits for the estimate and processes it before the next packet in its queue. The transmission of this estimate over the link takes $(\delta e/p)$ time, where p is the size of the data packet (that takes *S* units of transmission time) and e is the size of the estimate packet containing a Q-value and a C-value. The percentage overhead due to forward exploration is e/p (since the transmission time is proportional to packet size). In backward exploration, an additional Q value and its C-value are appended to the packet, increasing the packet size to (p+e). The transmission time of the larger packet is $(p+e)\delta/p$. Again the percentage overhead due to backward exploration is only e/p. The total overhead due to both forward and backward exploration is therefore, 2e/p. Since estimate packet contains only a Q-value and a C-value, this overhead is less than 2%, while the adaptability they establish improves the performance of the routing algorithm multi-fold.

Unbounded FIFO queues were used in the current simulations for simplicity. In the real world, the queue buffers of the network routers are finite, leading to possible congestion in heavily loaded parts of the networks. Extension of the CDRQ-Routing to address the problem of finite buffer networks is an important future direction. This extension would make CDRQ-Routing a more realistic routing strategy that does not only route optimally, but can also sustain higher loads in finite buffer networks to avoid congestion. Also the processing speeds and link delays *8* are assumed in this paper. While this assumption is realistic for a small scale LAN, it is not valid for a heterogeneous communication network like the Internet. Ability of CDRQ-Routing to learn effective routing

policies on such networks is another direction of future

6 Conclusion

In this paper a new adaptive network routing algorithm, CDRQ-Routing, was presented. It combines Q-Routing and Dual Reinforcement learning to get increased explorative capabilities, and introduces a confidence measure to quantify the reliability of Q-values. CDRQ-Routing is superior to Q-Routing and Bellman-Ford Routing both in terms of quality and quantity of exploration as shown by simulation results. CDRQ-Routing learns a better routing policy more than twice a fast as Q-Routing at various load levels. At high loads, the CDRQ-Routing policy performs more than twice as well as Q-Routing policy in terms of average packet delivery time, while Bellman-Ford routing breaks down at high loads. Moreover, CDRQ-Routing can sustain higher load levels than shortest-path routing, Bellman-Ford Routing and Q-Routing. The additional overhead of adding backward exploration and C-values is less than 0.2%, which makes CDRQ-Routing an efficient and practically viable adaptive network routing algorithm specially as compared to the Bellman-Ford Routing.

References

[Bellman, 1957] R. £. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[Bellman, 1958] R. E. Bellman. On a routing problem. *Quarterly of Applied Mathematics,* 16:87-90, 1958.

[Boyan and Littman, 1994] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6.* MIT Press, Cambridge, MA, 1994.

[Goetz *et al,* 1996] Patrick Goetz, Shailesh Kumar, and Risto Miikkulainen. On-line adaptation of a signal predistorter through dual reinforcement learning. In. *Machine Learning: Proceedings of the 13th Annual Conference (Ban, Italy),* 1996.

[Kumar and Miikkulainen, 1997] Shailesh Kumar and Risto Miikkulainen. Dual reinforcement Q-routing: An on-line adaptive routing algorithm. In *Proceedings of the Artificial Neural Networks in Engineering Conference,* 1997.

[Kumar and Miikkulainen, 1998] Shailesh Kumar and Risto Miikkulainen. Confidence-based Q-routing: An on-line adaptive network routing algorithm. In *Proceedings of the Artificial Neural Networks in Engineering Conference,* 1998.

[Kumar, 1998] Shailesh Kumar. Confidence based dual reinforcement q-routing: An on-line adaptive network routing algorithm. Master's thesis, Department of Computer Sciences, The University of Texas at Austin, Austin, TX-78712, USA, 1998. Tech. Report AI98-267.

[Littman and Boyan, 1993] M. Littman and J. A. Boyan. A distributed reinforcement learning scheme for network routing. In *Proceedings of the First Internation Workshop on Applications of Neural Networks to Telecommunications,* pages 45-51, Hillside, New Jersy, 1993. Erlbaum.

[Press *et al.,* 1995] W. H. Press, S. A. Teukolsky, W. T. Vellering, and B. P. Flannery. *Numerical Recipies in C.* Cambridge University Press, Cambridge, UK, 1995.

[Tanenbaum, 1989] A. Tanenbaum. *Computer Networks.* Prentice Hall, second edition, 1989.

[Watkins and Dayan, 1989] C. J. C. H. Watldns and P. Dayan. Q-learning. *Machine Learning,* 8:279-292, 1989.