Learning with Q-Routing in Networks

Srinivasa Pranav, Naman Priyadarshi, Siddarth Shanmuga Sundaram

Abstract

Traditional dynamic routing algorithms used in the network layer, link-state routing and distance vector routing, utilize Dijkstra or Bellman-Ford's algorithm respectively. These network routing algorithms are susceptible to overloading popular links present in multiple shortest path trees, making these links severe bottlenecks for network performance. As a result, Q-learning, a reinforcement learning algorithm, can be applied to alter a packet's route on-the-fly using estimates of the delay along each path. This is implemented in networks through acknowledgements which backpropagate minimum router-to-destination packet delays.

Overview of Q-Routing

$$Q'(d, n_1) = min\{Q(d, n_2)\} + \text{Transmission Time}(R \text{ to } n_1) + \text{Waiting Time}(R)$$

$$diff = Q'(d, n_1) - Q(d, n_1)$$

$$Q(d, n_1) \mathrel{+}= \alpha * diff$$

Q(d, n): current estimate for delay from router n to device d. New estimates are integrated into our old estimate for Q(d, n) through exponential averaging. When selecting a path to the destination device, the router observes the Q values of neighboring routers and selects the neighbor with minimum estimated delay.

These estimates are initialized to all be 0, but once the packet reaches the desired destination, we will have a more accurate estimate for the delay to that device, and this information will propagate backwards. As more packets are able to complete different paths in the initial exploration phase, we can use the improved estimates to find the minimum delay paths to between devices.

Simulation Structure

The simulation was written in Python by creating two classes, one for Devices and one for Routers. Routers maintained a queue with the current packets it needs to transmit, as well as its Q values which encode the current estimates for delay to all the devices. We would check at a given time step if there were any idle devices and with some probability p we would send a packet from these devices. Then, we would check for the shortest transmission and forward our simulation by that amount of time. When the desired amount of simulation time has passed, we collect data about the average delay per bit that packets experienced between two devices in the Dijkstra's routing approach vs the Q-Routing approach.

Network and Device Setup

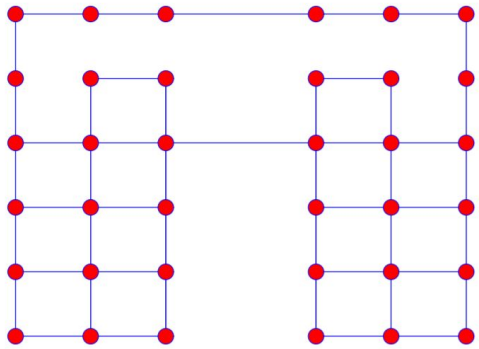


Figure 1: The irregular 6×6 grid topology

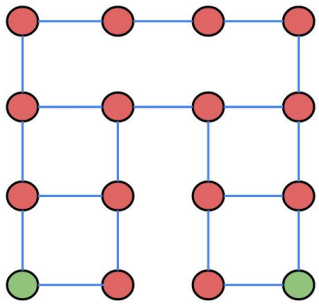
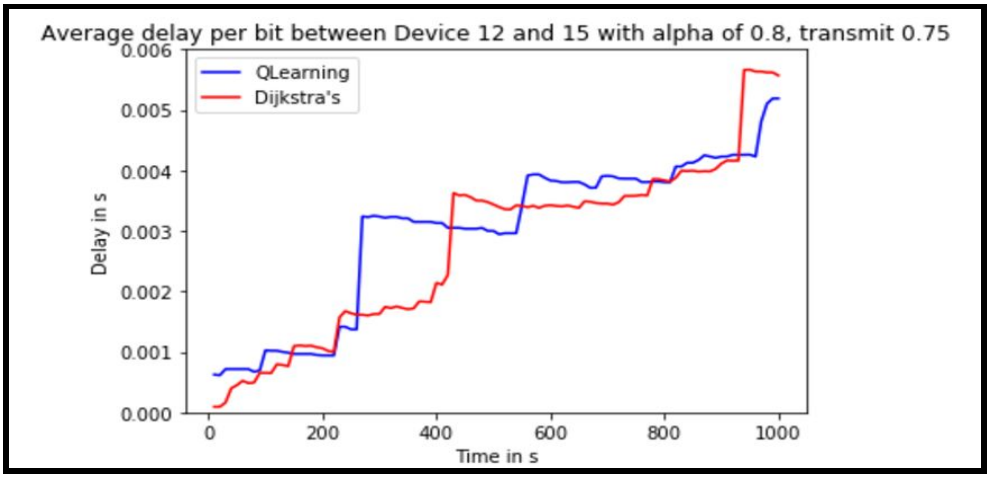


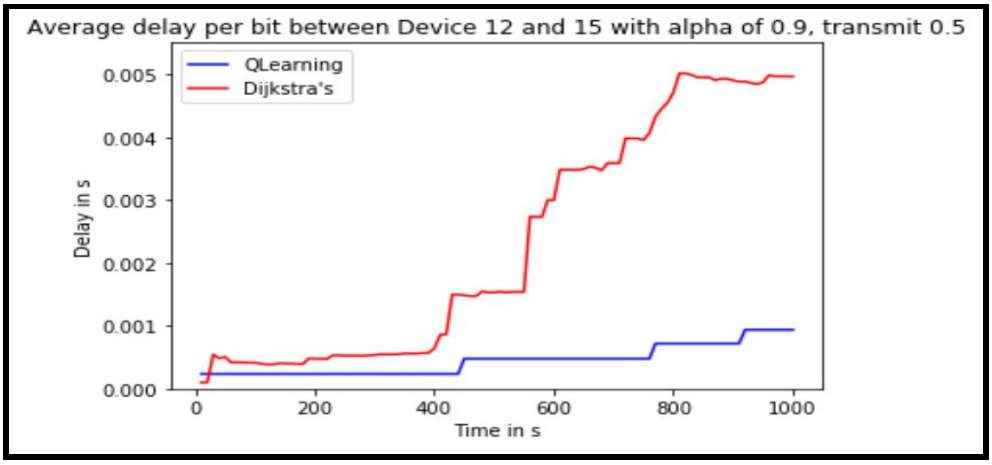
Figure 2: Our irregular 4x4 grid topology

As we did not have the computational resources to test the Q-routing algorithm on the common 6x6 topology (Figure 1) used in peer-reviewed papers, we improvised by designing a similar but smaller 4x4 topology (Figure 2). To collect and compare data, we focused on packets between the two furthest green routers shown in Figure 2.

Results and Findings

We varied the learning rate, α, and the transmission probability, p, to find when Q-routing outperformed Dijkstra's algorithm. Plotting our results, we found two scenarios that resulted in Q-routing being more efficient than regular dynamic routing: packet routing with a high learning rate and low transmission probability as well as routing with a low learning rate and high probability of transmission. With a high α and low p, Q-values change in larger step sizes but infrequently, effectively replacing and updating the previous value with the new Q-value. In the case of low α and high p, Q-values change more frequently but in smaller step sizes, resulting in an averaging of new and old Q-values.





Conclusion

Q-routing offers a solution to improve routing performance in congested networks, as opposed to simply using the shortest path algorithms. By varying the learning rates and packet transmission probabilities, we found that Q-routing was optimal in the following two cases: keeping a high learning rate and low transmission probability, or a low learning rate with high transmission probability. It should be noted that Q-routing outperforms the shortest paths algorithm only in congested networks; if the network has little congestion, it makes more sense to simply use a shortest paths algorithm like Dijkstra's. Finally, although this routing scheme with learning improved performance in congested networks, it is currently infeasible to implement in large scale networks like telecommunications and the Internet due to its excessive space requirements.

Alternatives

There are a handful of alternatives and helpful additions to Q-routing. Predictive Q-routing (PQ-routing), is a memory based algorithm that stores the best Q-values learned so far and reuses them by predicting network traffic trends. Though PQ-routing begins to outperform regular

dynamic routing and Q-routing faster (due to its predictive component), it uses more memory for routers' neighbors, which may be a significant constraint as the network size increases.

Approximate Q-learning is an alternative that reduces total network memory use from the worst case N^3. By using generalized Q-values for groups of destinations (US East Coast, for example) it reduces the size of Q-value heaps stored at each router and also improves runtime.

Dual Reinforcement Learning is an alternative that speeds up learning. In the original scheme, Q values are updated only by observing new estimates as packets are forwarded. But in Dual Reinforcement Learning, the Q value for the source device is passed along with the packet when moving the next router. Thus, the routers can learn values from both directions at the same time. Decreasing this learning time makes Q-routing more adaptive to network changes.
Lastly, in some applications, maximizing reliability of communication can be far more valuable than minimizing delay. In such cases, routers can maximize on Q-values modified to estimate path reliability based on proportion of dropped packets.

Code, Data, and Slides

Final code and IPython Notebook used for data analysis and graphing can be found on Github.
Presentation can be found on Google Drive.

References

Boyan, Justin A., and Michael L. Littman. "Packet routing in dynamically changing networks: A reinforcement learning approach." *Advances in neural information processing systems* (1994): 671-671.

Littman, Michael, and Justin Boyan. "A distributed reinforcement learning scheme for network routing." *Proceedings of the international workshop on applications of neural networks to telecommunications*. Psychology Press, 1993.

Choi, Samuel PM, and Dit-Yan Yeung. "Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control." *Advances in Neural Information Processing Systems* (1996): 945-951.