Charles Bennett
EE550

Distributed Backpressure Routing with Q-Learning

Abstract— There has been a lot of research into the use of the Backpressure algorithm for ad-hoc networks [1, 2, 3]. The algorithm is lithe and thus works well in ad-hoc networks, especially because of its ability to make optimal decisions without knowledge of a path. However, in the case of distributed Ad-Hoc networks, link states between nodes are unknown. This report proposes incorporating Q-Learning into the Backpressure algorithm to solve this problem.

Background

Several papers have discussed effective schemes for a distributed version of the Backpressure algorithm for ad-hoc networks [1, 2]. In such networks, nodes may be distrusted across a network with no notion of relative proximity. [1] and [2] propose distributed versions of the Backpressure algorithms which work well in given situatoins. However, neither incorporate notions of medium access. Like [2], the following algorithm uses Q-Learning to introduce learning through feedback.

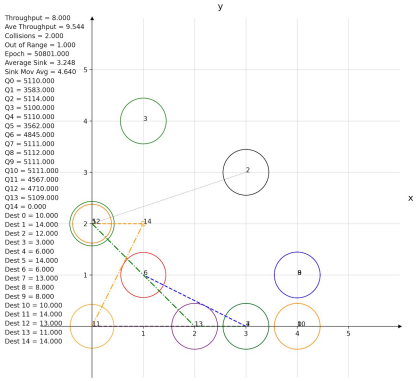
Scenario

The goal of this report is to introduce an algorithm designed for decentralized routing in ad-hoc mobile networks. Unlike other papers on the subject, this algorithm is based on a medium access model with unidirectional signaling.

For this analysis we have focused on the case of a 5 X 5 grid with 15 nodes each trying to communicate with a single source. Nodes, have limited range (Euclidian distance of 3), and thus must interact with each other to reach the destination. Nodes receive packets to send according to a Poisson random variable with parameter lambda. Nodes send to each other with unidirectional signaling which terminates at the sink. There is no orthogonal signaling or signal power sensing in this experiment. Nodes are presented only with a list of nodes and their queue sizes, they also know which node is the sink. At each iteration, they choose to send their packets to at most one destination. If the communication is successful, their queue is decremented in value.

Nodes in this scenario are completely blind. They initially have no notion of distance or relative positioning; thus Q-Learning is employed to learn about link states over time. Additionally, nodes only learn about link states through binary feedback. Nodes only know if a transmission is successful or not successful. They do not know, however, if a transmission is unsuccessful because it was out of range or because a collision occurred.

Collisions occur when the path between two sets of communicating nodes overlaps en-route. For example, if a node at position (0, 1) wanted to send to a node at position (1, 0) this would cause a collision if a pair of nodes at (0, 0) and (1, 1) wanted to communicate at the same time. To indicate a collision, the simulation draws dot-dashed lines (-.) between pairs of nodes whose communication has cause a collision. Furthermore, nodes who send to another node which is out of range will have unsuccessful communication. This is indicated by a dotted line (…). While nodes learn to share the medium, and about relative positions, it is common for collisions and out of range transmissions to occur. It is important to note that in this experiment, transmissions which are out of range of their destination may still cause collisions along their entire path. This makes out of range transmissions particularly damaging for throughput.



To further understand this scheme and the simulations used, let us look at the following generated simulation. Here the sink node is labeled with the number 14 (located at 2, 1) and all other nodes try to send to it. Each node receives packets to send following a Poisson arrival process with parameter lambda. Nodes can only send packets one at a time.

The sizes of each circle as well as the value on the LHS of the figure denote the queue sizes of each node. A larger circle thus indicates a larger queue. You can see that, for example, node 14 is a small circle because it is the sink and thus has no queue.

Thin dotted lines denote transmissions which are unsuccessful since they are out of range (out of range transmissions may still cause collisions). A dot-dashed line represents a transmission which has resulted in a collision. For example, 12 (who is in the same location as 5) has tried to send to 13 at the same time 11 has tried to send to 14 causing a collision.

Charles Bennett
EE550

ALGORITHM

A. Overview

The proposed algorithm uses a combination of Q-Learning and Backpressure to find optimal scheduling and path finding for the situation described above. This approach is completely different from [2] which As opposed to [2], this algorithm uses states which are time and space. Furthermore, Q-learning is used to estimate the link state over time. So this algorithm can be seen as exactly the same as regular backpressure discussed in [3] with the added feature that throughput values (mu) are estimated using Q-Learning.

B. Q-Learning Agents and States

In this algorithm, each link is its own Q-learning agent, like the scheme proposed in [2]. A large difference between [2] and the proposed algorithm is the state space. Whereas the state space in [2] is location based (links have Q values based on the Q values of other accessible links) the proposed state space is based on solely on the past actions and feedback on a given node. Thus, while [2] requires knowledge of all Q values in the network, the given algorithm only needs feedback on the results of its actions.

The state space for each Q-learning agent is encoded by the action of the last transmission and the result of that action. It is obvious that this state space is sufficient to describe any optimal solution to the given problem. In fact, if N is the number of nodes, then N-1 states at each node is sufficient to generate an optimal policy.

This was chosen to be the state space after trail and error, but it intuitively offers several advantages. First, encoding using the result of the last transmission allows optimal strategies which includecollisions.

As mentioned above, each link acts as an independent Q learning agent. Nodes are not originally aware of their relative proximity to other nodes and thus each node has N-1 Q-learning agents if N is the total number of nodes. Initial values are set to 1, and

C. Definition

$$\arg_J \max \left(\mu_{Q-Learn}\left(i,j,t,state(i,j,t)\right)\cdot\left(Q_i(t) - Q_j(t)\right)\right)$$

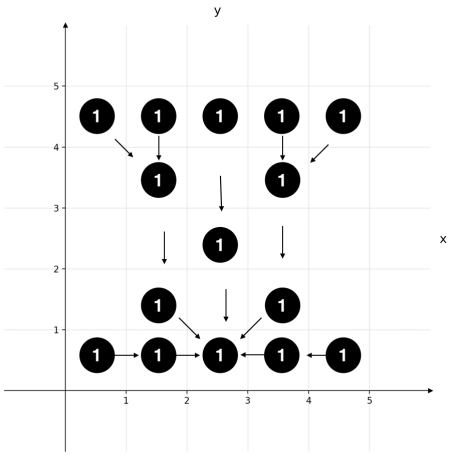
Nodes make decisions based on the above equation. A state, as mentioned before, each edge leaving a node (an edge being the potential of any node i sending to node j) maintains the same state and time t. This state is determined based on where the node last send data at time t-1 and whether that transmission was successful. Based on this state space the variable above (mu q-learn) is computed using traditional q-learning. It received a reward (and future discounted rewards) if an action at a given state is successful and a penalty otherwise. Thus, with the addition of this q-learning variable taking the place of the link state, this is just regular backpressure routing.

A. Analysis of Hourglass Shape

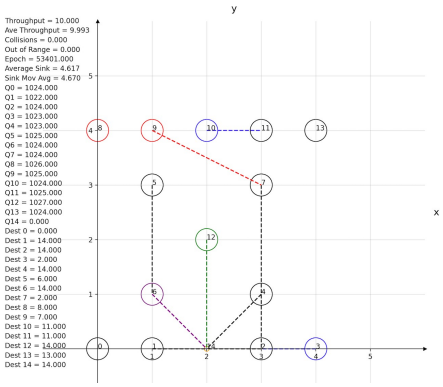
For this portion of the analysis, let us consider a special
case of this set up. Take a 5x5 grid with 15 nodes making up an hour glass shape. That it, 5 nodes lining the top and bottom of the grid, with 5 nodes making an X in its center. The sink is said to be at the center bottom of the shape. We say that nodes may send no more than 3 units of Euclidian distance to require the cooperation of nodes. Furthermore, nodes can only send one packet at a time. We also assume Poisson arrivals at each node with a parameter lambda.

Let us first calculate what the maximum lambda is that could be supported by this scheme. Looking at the below figure, we can simulate what it is like to have 1 packet at each node. We see that the bottleneck generated by the top ten nodes, which must reach the source through the three points at [3, 1], [3, 3] and [2, 2]. It is then clear that this graph can clear at most 3 packets from the top ten nodes, so the arrivals at the top ten nodes must be on average less than 10/3. However, it can also be shown that it takes 4 steps to clear this data. Thus, if the algorithm can handle an arrival rate of 10/4 then it is finding an optimal solution for routing/scheduling.

Charles Bennett
EE550

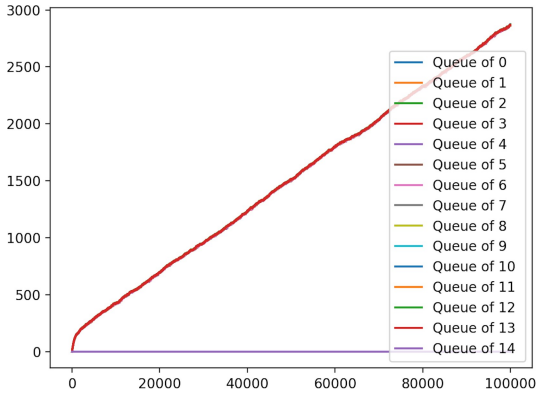


We simulate how to algorithm does in this situation to see if it can stabilize queues and find optimal routing/scheduling. An accompanying animation shows this routing happening in real time with figures like the following…



Where the size of each circle represents the queue size of every node.

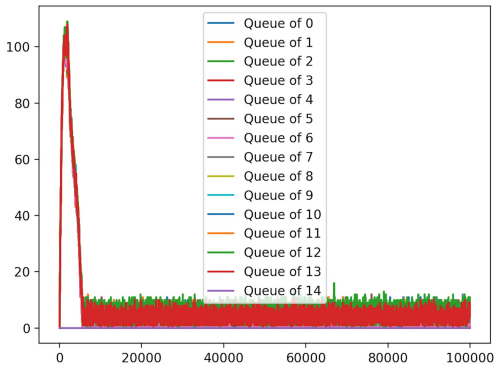
Running this simulation over 100,000 epochs lead to the following results. With an arrival rate of lamda = 1/3, we see that queue sizes increase linearly with the number of epochs.



Plotting the Queue values, we see an almost constant rate of increase. This is not mean rate stable. And as expected queue sizes would go to infinity over time.

Charles Bennett
EE550

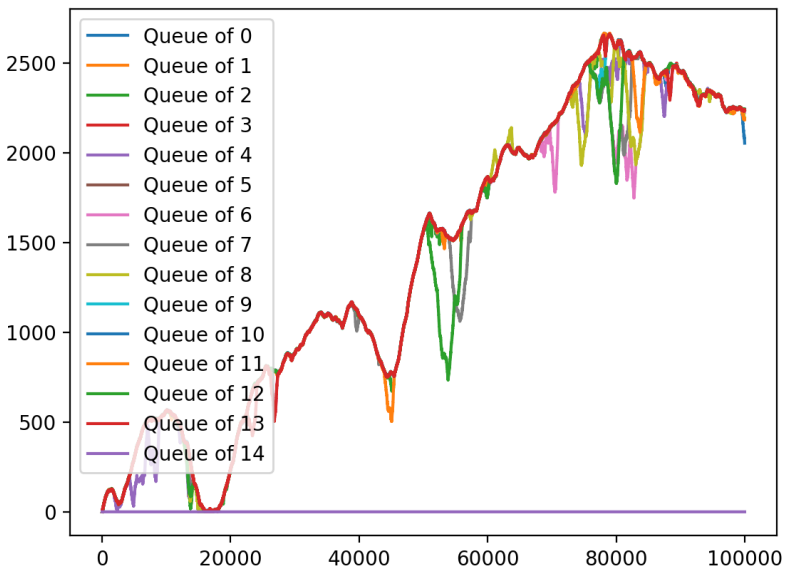
In [5] it is said that if all queues are mean rate stable then constraints are satisfied. We can validate this algorithm on the hourglass shape using lambda = 4/15. Here the queue values stabilize.



We see that we have mean rate stable queue sizes using our algorithm. Each node has no information about other nodes apart from other nodes queue sizes. If each node only knew adjacent nodes' queue sizes this would make the problem even easier by letting each node know who his neighbors are. Thus we can say that stability can be achieved with a 1/3 < threshold < 4/15. We have thus validated a distributed backpressure algorithm with very complex constraints.

B. Random Walk

This scheme is interesting because it allows nodes to adapt to situations in dynamic situations. As such, it is interesting to consider how queues stabilize when nodes can move randomly. For this experiment defined a random walk as follows. A node with stay in one position for time geometrically distributed with parameter Mu (fixed at 1/1000). Then the node moves in one of 8 directions with equal probability. This was simulated with varying parameters of lambda for Poisson arrivals.

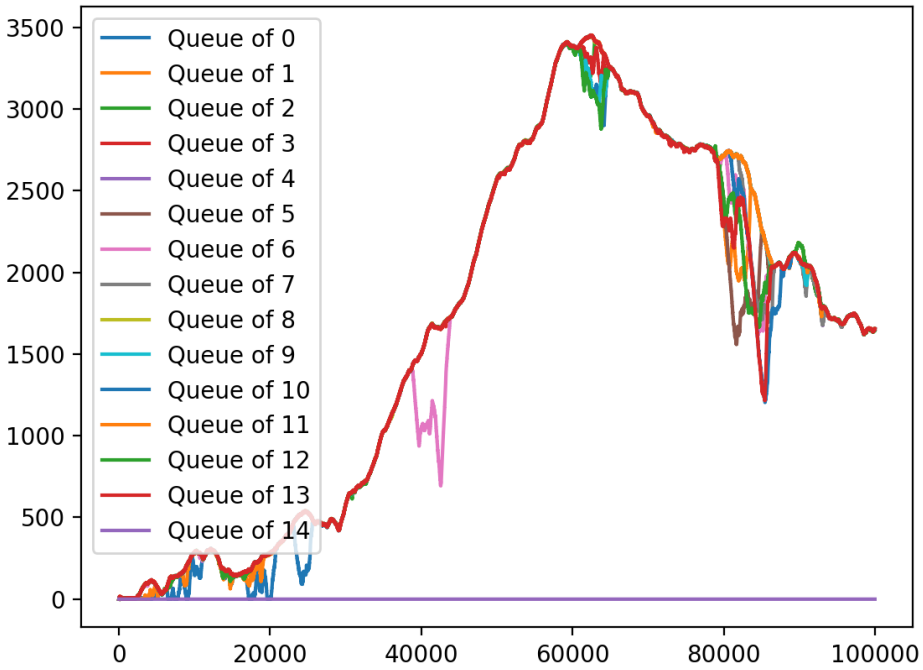


As expected, the parameter 4/15, which produced stable queues for the hourglass shape was not stable when introducing this random walk. This seems intuitive because random walks not only cause challenges by requiring new optimal scheduling configurations, but also sometime have optimal scheduling which still does not allow for very many arrivals. One example is when a node is out of range of all other nodes, or if the sink because out of range of all other nodes. The figure on the right is a graph of queue sizes using an arrival rate of 15/4. This is clearly not mean rate stable.

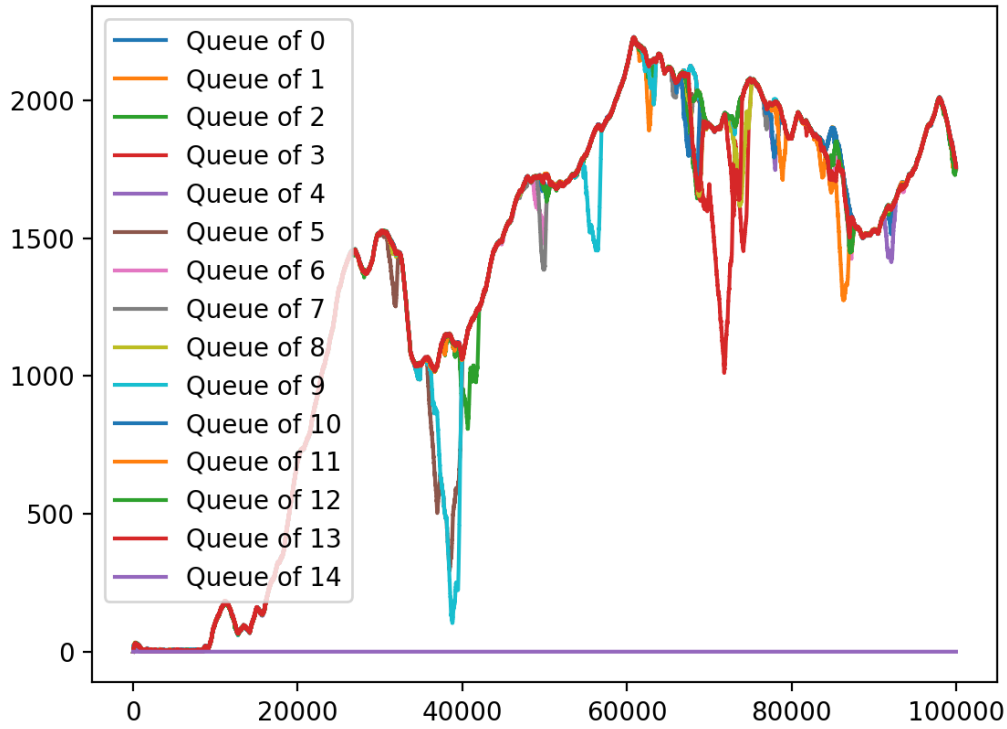
In the following three graphs, we relax the arrival rate lambda by .02.

Charles Bennett
EE550

Lambda = .24

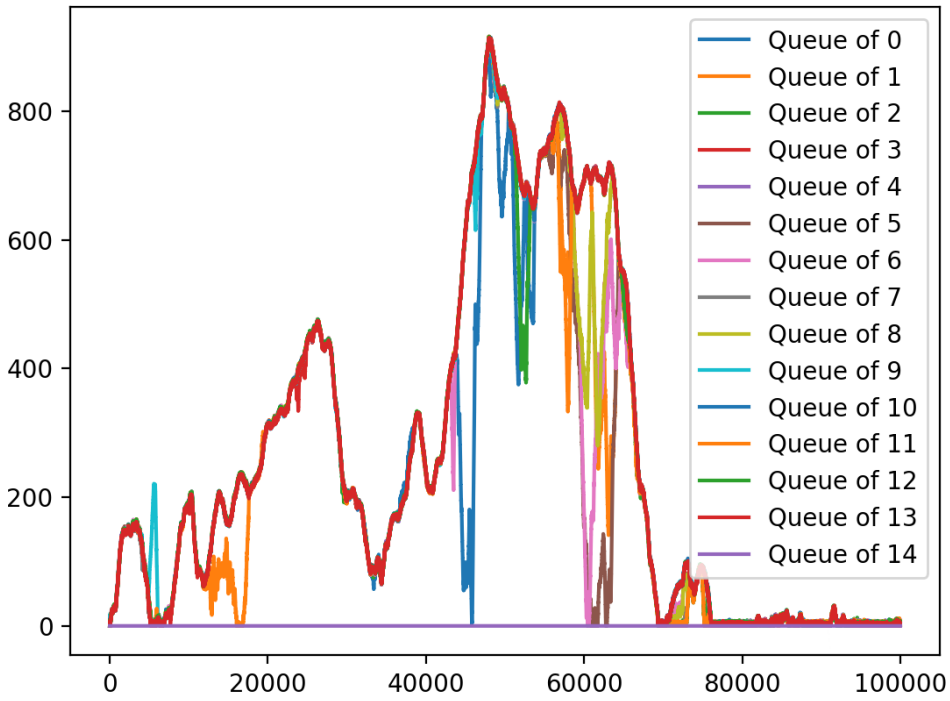


Lambda = .22

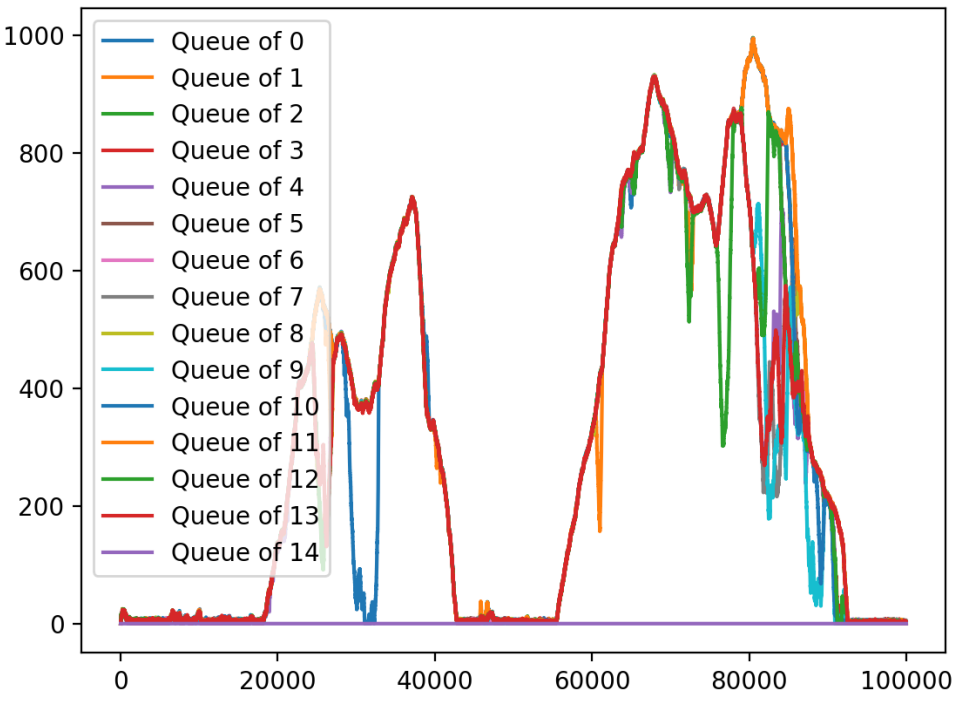


Charles Bennett
EE550

Lambda = .2



Lambda = .18



Charles Bennett
EE550

It appears from these graphs that the system becomes mean rate stable perhaps as early as when lambda is relaxed to .24.

CONCLUSION

By combining a Q-Learning with the Backpressure algorithm we have found a way for nodes to communicate in a collision free and optimal scheduling and routing. This can be done on a distributed set of nodes.
Future analysis could see how Q-Learning can be further combined with backpressure for nodes to learn to share in even more complicated environments.

ATTACHEMENTS

Please see the attached video of the routing in the hourglass shape and simulation of routing for a random walk. A lot of time was spent developing code for these simulations. To run yourself, you may use the code attached and edit the following parameters…
RNG : How far can packets be sent
Random_Walk: If a random walk is performed (positions are default in hour glass shape)
Mu: If Random_Walk is set to one then Mu dictates the amount of time spent in each location before a random step is taken.
Lambda: The parameter for the arrival rate to each node.

REFERENCES

[1] L. A. Maglaras and D. Katsaros, “Delay Efficient Backpressure Routing in Wireless Ad Hoc Networks,” *EAI Endorsed Transactions on Mobile Communications and Applications*, 29-Sep-2014. [Online]. Available: https://eudl.eu/doi/10.4108/mca.1.4.e5. [Accessed: 07-Dec-2020].

[2] Juntao Gao, Yulong Shen, Minoru Ito, and Norio Shiratori, *Multi-Agent Q-Learning Aided Backpressure Routing Algorithm for Delay Reduction*. [Online]. Available: https://arxiv.org/pdf/1708.06926.pdf.

[3] M. Neely, “Notes on Backpressure Routing,” *USC*. [Online]. Available: http://ee.usc.edu/stochastic-nets/docs/backpressure.pdf.