A Machine Learning Concept for DTN Routing

Rachel Dudukovich[1], Alan Hylton[1], Dr. Christos Papachristou[2]

[1]*NASA Glenn Research Center, Cleveland, OH, USA*

[2]*Case Western Reserve University, Cleveland, OH, USA*

***Abstract*— This paper discusses the concept and architecture of a machine learning based router for delay tolerant space networks. The techniques of reinforcement learning and Bayesian learning are used to supplement the routing decisions of the popular Contact Graph Routing algorithm. An introduction to the concepts of Contact Graph Routing, Q-routing and Naïve Bayes classification are given. The development of an architecture for a cross-layer feedback framework for DTN protocols is discussed. Finally, initial simulation setup and results are given.**

I. Introduction

This paper focuses on the development of a machine learning based routing algorithm for interplanetary delay tolerant networks. Many routing algorithms have been developed for both opportunistic and deterministic delay tolerant networks. Resource Allocation Protocol for Intentional DTN (RAPID)[1], Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET) [2], Spray-and-Wait [3], and Delay Tolerant Link State Routing (DTLSR)[4] are among the most well-known for opportunistic networking scenarios. Within the realm of interplanetary networking, one of the most popular routing algorithms is Contact Graph Routing [5] or CGR, which uses the known contact times and distances of scheduled network assets to determine an efficient route. Routing algorithms intended for delay tolerant networks and in particular space networks must address several issues. Flight hardware is often limited in terms of processing capability and memory resources, so the algorithm must be efficient and not use excessive computations or require a large amount of data storage. In deep space, there may be a significant propagation delay between network nodes, so the trading of network status data becomes costly and may not reflect the current state of the network. In addition, communication links are often asymmetric, so that a large amount of data may be sent from a network node, but there may be limited bandwidth to receive acknowledgments or other status information.

The work here is an attempt to take a small step towards a more cognitive communications paradigm by studying popular machine learning algorithms which may be used to enhance the functionality of existing routing algorithms. Two methods which readily adapt themselves to cognitive routing are reinforcement learning and Bayesian learning. We present a hybridized approach which would apply both to the basic Contact Graph Routing algorithm to allow it to more readily adapt to changes in the network, while still utilizing many of the strengths of CGR.

Delay tolerant networking is an overall architecture and set of protocols that have been developed to improve networking capabilities in a variety of scenarios in which an end-to-end path may not always exist and latency between nodes may be prohibitively long for some conventional protocols. DTN techniques have been applied to networks in rural developing regions, mobile networks in which an end-to-end path may not exist and space networking. This paper focuses on techniques that have been applied to space networking in particular. Space networks have several defining features which impact their communication operations. Each node in the network typically has predictable periods of contact with other nodes, due to the fact that orbital characteristics impact when communication assets will be in sight of one another. Deep space communications are characterized by long propagation delays which make frequent handshaking or trading of feedback signals between nodes costly, and algorithms which rely on current information from other nodes may make decisions based on old information, resulting in poor performance. In addition, high error rates may result in multiple retransmissions and require that data are stored until it can be received successfully. Data must also be stored between contact opportunities, thus an effective plan of managing data storage space must be developed or it is possible that the node will have to either stop accepting incoming data or begin to delete existing data if its capacity has become exhausted.

Delay tolerant networking uses a store and forward approach to mitigate the effects of long delays and disruptions. Bundles [6], the protocol data unit used in DTNs, are kept in longterm storage until there is an opportunity for contact with another node. If this neighboring node is the bundle's destination or if it leads to a path to the destination, the bundle may be transmitted to the neighboring node. Bundle protocol is an overlay protocol which can be used to transfer data across heterogeneous networks. As such, a variety of lower level protocols including TCP/IP, UDP, and LTP [7] among others, may be used.

R. Dudukovich is with the Flight Software Branch of NASA Glenn Research Center, Cleveland, OH 44135 and is a PhD candidate at Case Western Reserve University (e-mail: rachel.m.dudukovich@nasa.gov). A. Hylton is with the Architectures, Networks and System Integration Branch of NASA Glenn Research Center, Cleveland, OH 44135 (e-mail: alan.g.hylton@nasa.gov). C. Papachristou is with the Computer Engineering department of Case Western Reserve University, Cleveland, OH 44106 (e-mail: cap2@case.edu).

There has been much previous work done regarding machine learning routing and extensive work done on the topic of probabilistic routing in opportunistic networks. The approach taken in this paper differs in several ways. Probabilistic approaches such as PRoPHET [2] and Opportunistic CGR (OCGR) [8] tend to try to predict the likelihood that two nodes will encounter one another. Furthermore, machine learning methods such as Q-routing [9] have not addressed the challenges specifically found in space networks, such as the potentially limited bandwidth and long one-way times for feedback data. The method presented here attempts to use the attributes of a given contact period to classify its degree of reliability among a set of potential routing paths. The algorithm will use data already available from the lower protocol layers to provide feedback to its decision making process.

II. Interplanetary Scheduling and Routing

A. Scenario

The DTN routing problem best lends itself to a network of multiple nodes with multiple potential paths as might be found in a deep space scenario. An example of such a network might consist of multiple surface assets such as Martian rovers that perform science data collection on the planet's surface. The resulting data are transmitted to relaying satellites such as the Mars Reconnaissance Orbiter, among others. The orbiter then sends the data to the earth ground station. As the Deep Space Network expands, more nodes and more potential route selections will exist. Figure 1 shows the conceptual network of nodes this work focuses on. The network consists of three deep space surface assets such as rovers, three potential relay satellites and three earth ground stations. Each rover (nodes 0-2) may transmit and receive from each relay satellite (nodes 3-5) which then may forward the data to any of the 3 earth ground stations (nodes 6-8). The ground stations are interconnected by high bandwidth links.

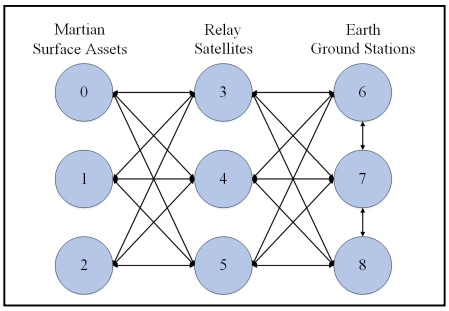


Figure 1. Conceptual Deep Space Scenario

B. DTN Routing Algorithms

Contact Graph Routing [5] (CGR) is a popular DTN routing algorithm, particularly for space networks which have highly periodic contacts between nodes. It is typical in such networks that communication will be scheduled (manually) days, weeks or months in advance. In addition to human scheduling constraints such as operator availability and sharing resources among multiple users, orbital constraints on the communication assets make it relatively straight forward to know when and for how long two nodes will be physically able to contact one another. Contact Graph Routing uses this upfront knowledge to determine suitable routes based on contact times. To do this, CGR uses a contact plan as input to the CGR routing algorithm. The information in the contact plan is entered by users either through update commands in a DTN administration interface program or as configuration files.

CGR begins with basic network information which is obtained from user supplied configuration files. These files define a set of contact messages and a set of range messages. The contact messages contain the start and stop time that a given contact opportunity pertains to, the transmitting node number, the receiving node number and the planned data rate between the nodes in bytes per second. The range messages consist of the start and stop time that a given range pertains to, the transmitting node, the receiving node and the anticipated distance between the two nodes in light seconds. Upon initialization, destination variable D is set to the bundle's final destination and deadline variable X is set to the bundle's expiration time. Bundle forfeit time is set to infinity and the best-case delivery time is set to zero. The list of proximate (neighboring) nodes is empty. A list of excluded nodes is populated with the node from which the bundle was received and all excluded neighbors for the destination node.

When a new bundle arrives to be forwarded to another node the CGR algorithm begins with the contact review procedure as shown in Figure 2.

```
CGR Contact Review Procedure:
Append D to the list of excluded nodes
For each xmit m in node D's xmit list:
    If m's start time is after the deadline X:
        Skip xmit m
    Else:
        If D is a neighbor of the local node S:
            Compute Estimated Capacity Consumption of
            bundle for tx from local node to D
                If m's residual capacity < Estimated Capacity
                Consumption:
                    Skip xmit m
                Else:
                    If D is already in the list of proximate nodes:
                        Skip xmit m
                    Else:
                        Is m's stop time < forfeit time:
                            Set forfeit time to m's stop time
                            Add D to list of proximate nodes
                            Compute forfeit and best case delivery
                            times to tx to D
                            Remove D from excluded node, revert
                            forfeit and best case times
        Else:
            If node S is already in the list of excluded nodes:
                Skip xmit m
            Else:
                If m's stop time < forfeit time
                    Set forfeit time to m's stop time
                    If m's start time > best-case delivery time
                        Set best-case delivery time to m's start time
                        Compute estimated forwarding latency
                        Invoke Contact Review Procedure
                        recursively with D=S and X =min(T,L)
                        Remove D from list of excluded nodes,
                        revert forfeit and best case times
```

Figure 2. CGR Contact Review Procedure [5]

For simplicity, we focus on the original algorithm given in [5] and do not include the several updates that have been made to the algorithm such as ETO-CGR (Earliest Transmission Opportunity CGR) [10] and overbooking management [10], though it is recognized that these are very relevant improvements.

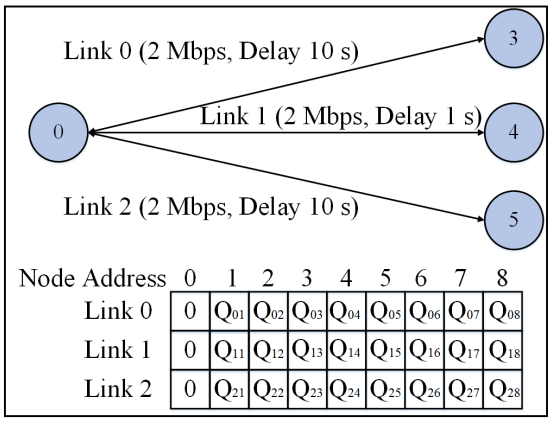
III. Machine learning

As a new approach to routing within a network which may adhere to deterministic schedules as well as be subject to uncertain disruptions, we propose a machine learning based routing framework. This solution will provide the adaptive benefits found in opportunistic routing strategies, while still adhering to user specified constraints which often exist within interplanetary networking. To accomplish this, our intelligent router uses a hybrid approach of both Bayesian and reinforcement learning. In addition, we leverage many of the benefits of the CGR algorithm, in that nodes are not required to transmit additional hand-shaking or status update packets, since this algorithm uses link state knowledge from the lower levels of the network stack.

A. Reinforcement Learning

Reinforcement learning is a commonly used machine learning algorithm in which the learner discovers how to achieve a desired outcome by maximizing a numerical reward [11]. Reinforcement learning systems typically consist of four elements: a policy, a reward function, a value function, and in some cases, a model of the environment [12]. Q-routing is an adaptation of the Q-learning algorithm developed for packet routing [9]. Q-routing uses the estimated end-to-end packet delivery time for the basis of its reward table, or Q-table. The table contains a row for each neighbor that a node has. Each column corresponds to a destination node. The entry for the row-column pairs in the table is the estimated time required for a packet to be received at the destination if it was sent from one of the possible neighboring node choices.

Figure 3 shows a single node in the network and links to each of its neighboring nodes. Its Q-table contains a row for each link to a neighbor and a column for every possible destination in the network. The index of each column corresponds to each node address. The entry corresponding to the node's own address is given a value of 0 (or some other indication of an invalid value), since it will not transmit data to itself. There are several approaches that can be taken for the initial estimate. All entries may be initialized to zero or a random value. Alternatively, a method can be developed to try to calculate an initial estimate of the end-to-end delays. The learner will determine what neighboring node to send a packet to based on which node minimizes the delivery time. Once the packet has been sent to the chosen neighboring node, the neighbor will reply back with what it believes the remaining time will be to deliver the packet to its final destination. This response will be used by the first node to update its Q-table. Each update should incrementally improve the accuracy of the Q-table, since nodes closer to the destination should have a more accurate idea of the remaining delivery time [9].



Node Address	0	1	2	3	4	5	6	7	8
Link 0	0	Q_{01}	Q_{02}	Q_{03}	Q_{04}	Q_{05}	Q_{06}	Q_{07}	Q_{08}
Link 1	0	Q_{11}	Q_{12}	Q_{13}	Q_{14}	Q_{15}	Q_{16}	Q_{17}	Q_{18}
Link 2	0	Q_{21}	Q_{22}	Q_{23}	Q_{24}	Q_{25}	Q_{26}	Q_{27}	Q_{28}

Figure 3. Node with 3 Links and Corresponding Q-table

Pseudo-code for the Q-routing algorithm is shown in Figure 4. The variable η represents the learning rate, t is the transmission delay over the link from node x to node y, and q is the queuing delay at node y. When a packet arrives it enters the node's inbound queue. The Q-routing algorithm will compare the Q-values (delivery time estimates) for transmitting the packet to its destination via each neighboring node and select the neighbor with the smallest Q value. When the packet arrives at its destination, the receiver responds back with its own estimated delivery time. This is then used to update the Q-table entry corresponding to that destination.

Q-Routing Algorithm
While(true):
Select a packet from queue
Select node y' from neighboring nodes with minimal Q(y,d)
Wait for response from y'
Update Q(y',d) in the current node using the new estimate from y'
$Q_x(y', d)=Q_x(y', d)+\eta[Q_{y'}(z', d)+t+q-Q_x(y', d)]$
end while
If packet received: interrupt while and do:
Receive packet p from node s
Select z' with a minimal Q(z,d)
Send the value of Q(z,d) back to node s

Figure 4. Q-Routing Algorithm [9]

B. Bayesian Learning

The concept of Bayesian machine learning is based on the conditional probability that a certain outcome has some likelihood given that it possesses a particular set of attributes. In particular, this paper focuses on the Naïve Bayes classifier. This learning method is used to classify a new instance or occurrence within a set of possible values based on previous training data. The learner will determine the probability that a certain set of attributes most likely correspond to a specific classification within the training data. When a new occurrence is presented to the learner, the training probabilities are used to determine the value v of the new instance from a finite set of values V based on its attribute vector $<a_1,a_2,...,a_n>$ [11].

Bayesian learning is based on calculating the most probable outcome, often called the maximum a posteriori or MAP

hypothesis. The Naïve Bayes classifier attempts to find the most probable value for a current instance V_{MAP} given its known attributes $<a_1, a_2,...,a_n>$ [11]. Equation 3 calculates the MAP hypothesis as the probabilities of observing the value v_j in conjunction with the attributes $<a_1, a_2,...,a_n>$. This is easily found by taking the product of the conditional probabilities of observing v_j given each individual attribute. Naïve Bayes classifier becomes [11]:

$$V_{MAP} = \max_{v_j \in V} P(v_j | a_1, a_2, \dots, a_n) \quad (1)$$

Bayes rule can be used to write Eq. 1 as:

$$V_{MAP} = \max_{v_j \in V} \frac{P(a_1, a_2, \dots, a_n | v_j) P(v_j)}{P(a_1, a_2, \dots, a_n)} \quad (2)$$

This simplifies to:

$$V_{MAP} = P(a_1, a_2, \dots, a_n | v_j) P(v_j) \quad (3)$$

$$V_{NB} = \max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j). \quad (4)$$

Naïve Bayes classification and decision tree learning have been proposed in [13] to opportunistically select available communication channels for cognitive radio sensor networks. Context information such as neighboring nodes, sink nodes, current time slots and the currently available channel set are used to predict link connectivity. An optimal routing path is obtained from the consideration of two classifiers which predict link stability.

IV. Learning Architecture

A. Cross-Layer Information

In order to implement a more cognitive routing application, some type of feedback is needed to allow the learner to improve its selection decisions. Many DTN algorithms trade status vectors [1], [2], [4] to inform other nodes in the network about its buffer contents, link status and other information. This is not always desirable considering that transmitting this status information uses additional network resources and may become stale due to long distances and link asymmetry (the feedback may be transmitted on a much slower link). For this reason, this work proposes the use of convergence layer protocol report segments to be used as the feedback mechanism. This type of approach was first suggested in [14] to implement an end-to-end retransmission framework. It is following this train of thought that this work follows to implement a machine learning based router.

An example of this approach could be a sending node using LTP as a transport protocol. The node will receive reception reports when red data (reliably sent) segments are lost and will also have knowledge of the LTP checkpoint timer expiration. This information can simply be stored in a database and used by an intelligent router to better understand packet losses and delays within the network. This imposes no changes to the LTP protocol and can make use of a very simply logging mechanism that can also be used for network troubleshooting by operators on the ground. Furthermore, this approach does not consume any additional bandwidth as no new packets are generated that would not otherwise be used. While this does require the data to be stored, the size is rather minimal and as stated before, can serve multiple purposes such as network administration.

Figure 5 shows a conceptual architecture for an intelligent router. A source will transmit data to the intelligent router node to be forwarded to a final destination. In a similar manner to traditional CGR, the router will consult its contact plan and range database to attempt to find a suitable path through the network. Additionally, the intelligent router will have a local database of stored network statistics consisting of retransmission requests between a given pair of nodes corresponding to a periodic contact opportunity, as well as LTP check point timer expiration data, and estimates of the amount of data already sent to the prospective nodes. In addition, if the destination node is an earth ground station, weather at the location of the ground station, historical and current for this contact can also be stored in a database and used to correlate failed or unreliable transmission attempts on this path. Once a neighboring node has been selected as the best candidate, data is transmitted in a similar manner to standard CGR. Any retransmission or timer expirations for this current contact are then stored to the statistics database for future routing decisions. It is in this way that the learner will begin to determine what historically have been the best routes for a specific destination.

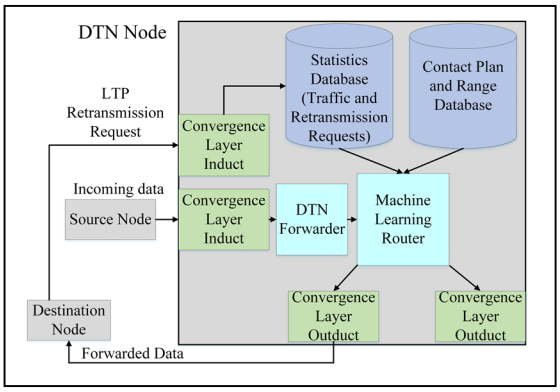


Figure 5. Intelligent Router Architecture

B. The Bayesian Model

Using the concept of a contact database presented in CGR, we will further describe the attributes of each contact to create a model of the network to be used with Naïve Bayes. The frequency of occurrence of each possible attribute value can be mapped to the probability that a given contact opportunity behaved in a reliable or unreliable manner. The statistics database can contain historical records of the contact attributes and the reliability observed for each contact opportunity. This will provide training data for the learner to determine the probability of reliability for future transmissions during similar contact opportunities.

In addition to the contact start and stop times, distance and data rates, we are interested in other factors which may cause a selected route to be a less desirable route in comparison to other opportunities. The historical percentage of retransmission requests between two nodes at a given time during its periodic contact opportunities may be a good indication of how reliable a link is. The amount of traffic sent to a neighboring node is also relevant to control congestion.

If the bit error rate of a given link is known, this would also be an important factor. Furthermore, if knowledge of the weather in the vicinity of a ground station is known, this could also be an attribute of the model. While knowledge of weather conditions may not be possible or even desirable for all nodes, one of the strengths of Naïve Bayes is that its attributes can easily be omitted without much effect to the algorithm. Table I summarizes potential attributes and sources for how this data could easily be obtained.

TABLE I. CONTACT ATTRIBUTES

Attribute	***Source of Attribute***
Retransmission requests	Convergence layer, stored in database
Average incoming data rate	Network statistics, stored in database
Average outgoing data rate	Network statistics, stored in database
Distance	Range information from CGR
Bit Error Rate	Network statistics or user supplied
Weather patterns	User supplied, system telemetry

Our routing algorithm begins at the end of the CGR contact review procedure in the forwarding decision procedure. It is here that the list of proximate nodes has been computed and now CGR will determine the best node to select. We will consider the prediction of reliability of each candidate neighboring node determined by Naïve Bayes and the contact attributes pertaining to each node. This probability can be part of a multi-criteria decision consisting of the standard selection produced by CGR as well as the reliability prediction of Naïve Bayes. Weights can be assigned to both values, so that users can select which of the criteria, CGR or Naïve Bayes, they would like to give preference to.

Once the selection of the best contact opportunity has been made and the data has been transmitted to the neighboring node, if a reliable transport protocol has been used for this link, our reinforcement learning strategy can be used. Retransmission attempts associated with this contact can be stored and examined by the learning algorithm. If this decision has been shown to be a reliable choice, more weight can be given to this contact for future data bound to the same destination. Further, a forgetting parameter can be added such that one node may not begin to become the dominate choice, thereby creating potentially undesirable congestion in the network.

V. SIMULATION

We have begun the development of the machine learning based router by first implementing a simple Q-routing based approach. This serves as a simple starting point as well as providing a separate algorithm to be used to compare our future results. The simulator used was OMNeT++ [15]. OMNeT++ is an open-source discrete event simulator that has a generic framework which can be used to develop simulations of communication networks, multiprocessors and distributed hardware systems and protocol modeling. OMNeT++ has a component based architecture that can be used to develop a wide variety of models, but it is very often used for communication network modeling. The behavior of the components or modules are defined in C++ classes, some of which are provided as base classes by OMNeT++, and some of which are developed by the user. OMNeT++ uses a message class to pass information between modules. The messages can be further defined by the user to represent network packets, status and timing internal messages or any other type of message the user requires.

The routing sample project provided with the OMNeT++ installation was used as the basis for our Q-routing simulation. Within the simulation model, the network is implemented as a set of nodes. Each node consists of three modules: an application module, a routing module and a queue module. The application generates and receives network packets, just as a software application would generate network traffic. The routing module determines where to send the packets that are generated by the application module. The queue module implements a vector of queues for transmitting and receiving the packets. The connections between nodes are defined as bidirectional and each has a delay time and data rate associated with it.

Network packets are generated by the application module. The rate at which packets are generated is configurable. The packet format is very simple and consists of a source address field, a destination address field, the current hop count, the last hop taken and the first hop taken. These fields are used by the routing modules to help determine where to send the packet. For Q-routing, a feedback message is generated by the routing module of each node and sent as a reply to the node that last transmitted a packet to the current node. The feedback message contains the address of the node sending the feedback message, the final destination address of the packet that replying node received, the remaining time estimate until the packet reaches its destination and the creation time of the packet. This information is used by the router modules to update their Q-table time estimates.

The routing module will check an incoming packet's destination address and choose which of its neighboring nodes to send the packet to based on which one has a smaller delay associated with reaching the packet's destination. In addition to this policy, network exploration is encouraged by forcing the router to choose a random link on every 500th packet. This will ensure that all paths will continuously get updated once the receiving node sends its response with the improved time estimate.

Initial simulations have been conducted using a 9 node mesh as shown in Figure 1. Each node application generates 50 kB packets to transmit to a randomly selected node following an exponential distribution with a mean which was varied from 1 to 0.1 seconds. Each simulation executed for 2.7 hours in simulation time (10000 simulation seconds). Table II shows the link characteristics used. The link characteristics have been selected such that multiple possible paths to a destination will have the same number of hops but a longer propagation delay and/or slower data rate. This was done as a test to determine that the Q-routing algorithm can successfully choose the quicker path based the average end-to-end delays of packets sent on a particular route. Q-routing was found to react to the network load quite well and as

expected, performed similarly to the shortest path algorithm under low network load but out-performed the shortest path algorithm as the network load increased. This shows that previous end-to-end delays may be a good indication of congestion or link unreliability along a particular path.

TABLE II. NETWORK SIMULATION PARAMETERS

Link	*Propagation Delay (s)*	*Data Rate*
0-3	1 to 30	200 kbps
0-4	1	2 Mbps
0-5	1	200 kbps
1-3	1	2 Mbps
1-4	1 to 30	200 kbps
1-5	1 to 30	2 Mbps
2-3	1	200 kbps
2-4	1	200 kbps
2-5	1 to 30	200 kbps
3-6	1	200 kbps
3-7	1	2 Mbps
3-8	1 to 30	2 Mbps
4-6	1	200 kbps
4-7	1 to 30	2 Mbps
4-8	1	200 kbps
5-6	1	2 Mbps
5-7	1	200 kbps
5-8	1 to 30	2 Mbps

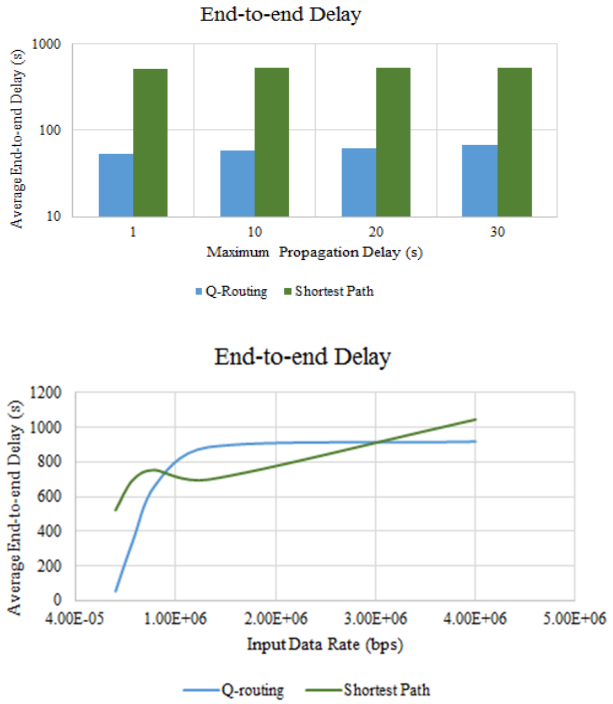


Figure 6. End-to-end Delays of Q-routing vs. Shortest Path Algorithms

VI. FUTURE WORK

While our machine learning based router is still very much a work in progress, we have developed a concept and architecture for an intelligent router that will not require additional status packets to provide feedback for the learner. To improve the fidelity of our network simulation we have investigated several simulators and emulators focused on delay tolerant and mobile networks. Particularly, the ONE (Opportunistic Network Environment) [16] simulator and CORE (Common Open Research Emulator) [17] have been used for modelling space DTNs. Going forward, it is our plan to use CORE to develop a more realistic scenario using Linux containers to run multiple instances of actual flight-like software with separate network stacks. CORE has already been released with an ION-based DTN development kit, which will give us access to a bundle protocol and LTP implementation to use as the basis for our development efforts. In addition, CORE supports custom mobility models which will allow us to more fully test our routing techniques during intermittent contact periods.

REFERENCES

[1] A. Balasubramanian, B. Levine, and A. Venkataramani. “Replication Routing in DTNs: A Resource Allocation Approach,” *IEEE/ACM Transactions on Networking*, vol. 18 pp. 596–609, April 2010.

[2] A. Lindgren and A. Doria, “Probabilistic Routing Protocol for Intermittently Connected Networks,” https://tools.ietf.org/html/draft-lindgren-dtnrg-prophet-02, 2006.

[3] T. Spyropoulos, K. Psounis, C. Raghavendra, “Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks,” SIGCOMM’05 Workshops, Philadelphia, PA, August, 2005.

[4] M. Demmer and K. Fall, “DTLSR: Delay Tolerant Routing for Developing Regions,” Proceedings of the 2007 Workshop on Networked Systems for Developing Regions, Kyoto, Japan, pp. 5-1 - 5-6, 2007.

[5] S. Burleigh, “Contact Graph Routing,” Internet Engineering Task Force, https://tools.ietf.org/html/draft-burleigh-dtnrg-cgr-00, December 2009.

[6] K. Scott and S. Burleigh, “Bundle Protocol Specification”, Internet Engineering Task Force, https://tools.ietf.org/html/rfc5050 , November 2007.

[7] M. Ramadas, S. Burleigh, and S. Farrell. Licklider Transmission Protocol-Specification, RFC 5326, https://tools.ietf.org/html/rfc5326, September 2008.

[8] S. Burleigh, C. Caini, J. Messina, and M. Rodolfi, “Toward a Unified Routing Framework for Delay-Tolerant Networking,” IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), Aachen, Germany, September 2016.

[9] M. Littman and J. Boyan, “A Distributed Reinforcement Learning Scheme for Network Routing,” Proceedings of the First International Workshop on Applications of Neural Networks to Telecommunications, pp. 45-51, 1993.

[10] N. Bezirgiannidis, C. Caini, D. Padalino, M. Ruggieri, and V. Tsaoussidis. “Contact Graph Routing Enhancements for Delay Tolerant Space Communications,” 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC), September 2014.

[11] T. Mitchell, *Machine Learning*, C. I. Liu, Ed.: McGraw-Hill, pp. 367-381(Reinforcement Learning) and 157-184 (Bayesian Learning), 1997.

[12] A. Barto and R. Sutton, *Reinforcement Learning: An Introduction*, Thomas Dietterich, Ed. Cambridge, MA, USA: The MIT Press, 1998.

[13] Z. Jin, D. Guan, J. Cho and B. Lee. “A Routing Algorithm Based on Semi-supervised Learning for Cognitive Radio Sensor Networks,” SENSORNETS, pp. 188-194, 2014.

[14] N. Bezirgiannidis, “Accurate Estimation of End-to-End Delivery Delay in Space Internets: Protocol Design and Implementation”, PhD thesis, Democritus University of Thrace, 2015.

[15] OMNet++ Discrete Event Simulator, https://omnetpp.org/.

[16] ONE Simulator, https://akeranen.github.io/the-one/

[17] CORE, https://www.nrl.navy.mil/itd/ncs/products/core