

Routing in Intermittent Network Topologies

Padma Mundur*, Sookyoung Lee*, and Matthew Seligman[§]

* Department of Computer Science and
Electrical Engineering
University of Maryland, Baltimore County
1000 Hilltop Circle, MD 21250
{pmundur, slee22}@cs.umbc.edu

[§]Laboratory for Telecommunication Sciences
8080 Greenmead Rd. College Park,
MD 20742
mattseligman@gmail.com

ABSTRACT

The topic of this paper is the algorithmic development of routing techniques for Delay Tolerant Networks (DTNs). Assuming a store and forward type of network transfers, our main objective in designing routing algorithms is to minimize the delay and maximize delivery subject to storage constraints on intermediate nodes connected by intermittent links. We present a novel modification to the breadth-first search algorithm to find the quickest route between a given source and any destination node in a delay-tolerant network. This is done without flooding the network – at any one time we maintain only one copy of the message in the network. We implement the proposed routing algorithm in NS2 and present an extensive performance analysis using metrics such as delivery ratio, incomplete transfers with no routes and dropped messages.

Categories and Subject Descriptors

C.2.1 [Network Architecture and Design]: Network Topology, Store and Forward Networks, Routing Algorithms

General Terms

Algorithms, Performance, Design

Keywords

Delay Tolerant Network (DTN), Routing Algorithms, Performance Evaluation.

1. INTRODUCTION

The topic of this paper is the efficient data delivery mechanisms in dynamic network topologies with intermittent links. The class of networks under consideration has been referred to as *delay tolerant or disruption tolerant networks (DTNs)*. Applications for these networks range from military combat situations to civilian applications of vehicle-based mobile data centers; disaster relief situations where fixed infrastructure may have been destroyed; a commuter bus as it moves through rural areas provides connectivity by acting as a store and forward switch.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MSWiM'06, October 2–6, 2006, Torremolinos, Malaga, Spain.
Copyright 2006 ACM 1-59593-477-4/06/0010...\$5.00.

Some unique challenges arise as we move away from the underlying assumptions for traditional TCP/IP-based networks. For TCP/IP protocols to work, there must be an end-to-end path between the source and the destination and the round trip delays must be small enough that there can be a “conversation” about the data transfer between the source and the destination. Neither of these assumptions is valid in a DTN -- intermittent connectivity makes it difficult to guarantee an end-to-end path for an ongoing data transfer and long round trip delays make it impossible to provide acknowledgements and retransmissions. The proposed DTN architecture (see [10] for details) offers a set of choices to counter these challenges: messages versus stream of packets; hop-by-hop delivery with optional in-network storage versus end-to-end routing. Given these new operational semantics, efficient data delivery becomes an important design issue with the objective of minimizing delivery time, minimizing buffer/storage usage, and maximizing delivery.

One of the main contributions of this paper is an approach where traditional routing algorithms (the simplest being the breadth-first search) can be adopted for DTNs by routing messages through successive changes in the network topology over a period of time. We propose and evaluate such an algorithm using extensive simulation and performance metrics such as delivery ratio, incomplete transfers with no routes and dropped messages.

Rest of the paper is organized as follows: Section 2 presents a brief description of related work. In Section 3, we present the proposed DTN routing algorithm with details of performance evaluation in Section 4. A brief conclusion follows in Section 5.

2. RELATED WORK

DTNs are overlays over heterogeneous networks providing network services and interoperability (For architectural details see [3] and [10]). Authors in [1, 4] propose several routing algorithms specifically for delay tolerant networks that consider intermittent connectivity. They modify Dijkstra’s shortest path algorithm by including link weights that take into account the waiting time at nodes because of disconnected links. Different variations of this modified algorithm based on the knowledge of the dynamic network topology are presented. However, none of these variations consider the all important storage constraints on DTN nodes. While they propose an LP formulation that takes into account the storage constraints for the store and forward DTN network, the performance of the LP-based algorithm is lacking, in their own words.

More recently, many researchers have turned their attention to the routing problem in sparse ad hoc networks where network partitions can occur frequently and data needs to be delivered to

nodes across partitions asynchronously. Most of the algorithms however, rely entirely on node mobility to move data in the event of network partitions such as epidemic routing in [9], or a refinement of the same algorithm in [2]. The authors in [8] propose a routing scheme called Spray and Wait that aims to reduce the overhead of flooding by spraying a limited number of copies into the network and waiting to see if that will suffice to reach the destination node. Zhao and others [11] employ special mobile nodes called message ferries in the deployment area that move in a predictable manner among the nodes to help collect and deliver the data. Musolesi and others in [6] consider asynchronous communication between nodes in separate clouds by computing delivery probabilities for each host and using the host with the highest delivery probability to actually deliver the data across the cloud. In the same spirit, authors in [5] present five strategies for opportunistic forwarding of messages when two mobile routers are within transmission range but where mobility of vehicles is not controlled. Our paper is different from these works in that we do not use flooding or mobility for data delivery. Unlike these other works, we assume that link states are predictable – we provide a justification for that in the next section. In addition, we evaluate the effect of storage limitation in intermediate nodes on data delivery. This paper along with the work in [7] on alternative custodians is part of the early research that consider storage constrained DTN.

3. QUICKEST DELIVERY ROUTING

In this section, we present a novel modification to the breadth-first search algorithm to find the quickest route between a given source and any destination node in a delay tolerant network. We assume that the link state changes are predictable. We assume that the links, when up, have sufficient bandwidth to carry the messages needed and that the intermediate nodes have sufficient storage. We also assume that network and transmission delays are negligible compared to the delays due to parts of the network being unreachable. Our algorithm determines the path in its entirety at the time of message origination.

The assumption of predictable link state changes is justified and similar to the situation presented in other works in this area. In our model, we do not use an agent to bring about link state changes but leave it as implementation dependent. For instance, Zhao and others in [11] employ special mobile nodes called message ferries in the deployment area that move in a predictable manner among the nodes to help collect and deliver the data. Their main idea is to make the node movement non-random so that data delivery is planned and more efficient. Given a pre-determined ferry route the nodes can either be static or pro-actively move closer to a ferry. With this type of set up, an event list for link state changes of the type we use in our formulation can be easily generated.

3.1 mBFS Algorithm Description

We adapt the breadth-first search algorithm for graphs to find the “quickest” route from a single source node to all other nodes in the graph. We assume an undirected graph $G = (V, E)$ where V is a set of vertices (or nodes) and E , edges. We assume an adjacency list representation of G , consisting of an array A of $|V|$ lists, one for each node in V . One of the nodes $s \in G$ is the source node. With delay tolerant networks, any edge (u, v) , $u, v \in G$ may be added or deleted at any time. Thus in a delay tolerant network, G changes with each addition and deletion. In general we call them *events*. We assume that events are *predictable*, in that we assume

that we know in advance which edge will be added to or deleted from the graph and at what time. For convenience, we will call fixed edges *static*. In contrast, edges which get added or deleted would be *dynamic*.

In our analysis we will assume a starting configuration for G at time t_o . We propose a linked list $Evts(u, v, t_e, a)$ representing events, where $(u, v) \in G$ is the edge that is added to or deleted from E at time t_e . a denotes the action, which could be either ADD or DELETE. The list is sorted by the event time t_e . We propose a time limit $T > t_o$ within which we are to restrict our search. This is to avoid potentially endless event lists where edges are added and deleted regularly. Thus the list $Evts$ must contain all events which occur between times t_o and T .

3.2 Single Source Quickest Delivery Algorithm

Given a graph $G = (V, E)$ at start time t_o , end time T , events list $Evts$, and a start node $s \in G$, the algorithm computes the earliest time at which a message originating at s can be delivered to any other node in G if at all possible within time T . The pseudo-code for the proposed algorithm (mBFS) is presented in Figure 1.

Our approach is to conduct a breadth-first search initially to discover the nodes reachable immediately from s . All nodes reachable during the initial search are assumed to be discovered at time t_o and are marked such. We process each event in the list $Evts$. As edges are added and deleted, we update E . If an addition of the edge leads to discovery of new nodes, breadth-first search is conducted with the newly discovered node as the new source node. All new nodes discovered during the search are marked with the time of the event that led to their discovery, t_e . In general, ADD events may lead to new nodes being discovered and therefore extending the route. On the other hand, the effect of DELETE events is limited to just a topology change. The algorithm terminates when all events up to the time T have been processed. It is possible that some nodes in G are not discovered.

We first modify the standard breadth-first search algorithm to store the time of discovery $t_{discovered}$ for each node discovered. We call this *modified BFS* procedure in our algorithm. In the pseudo-code in Figure 1, F is a first-in-first-out queue. d is used to store the *distance* from the source node (number of hops), and π , the *predecessor* node. We use WHITE to denote nodes which are not yet discovered, and BLACK, for nodes discovered and explored. GRAY nodes represent the discovered nodes which are being explored. At the end of the execution of the *modified BFS* algorithm, all nodes discovered are fully explored and are colored BLACK.

We next introduce the constraint that the amount of storage available at any node is limited. If S_u is the total storage available on node u , m is the size of the message, and s_u is the amount of storage in use at node u at a given time, we must ensure that:

$$s_u + m \leq S_u$$

To enforce storage constraint, we implement a drop policy in each node. Using the route found by mBFS algorithm, a message as it is being transmitted could be dropped due to storage constraint along the path. For a drop policy, we propose that the message with the longest life time in a queue would be dropped when there is no available storage.

4. PERFORMANCE EVALUATION

4.1 Simulation Setup

We evaluate the proposed DTN routing algorithm using NS2. Our DTN network topology includes 15 nodes with intermittent links between pairs of nodes. There are several parameters that are applied to affect the basic 15 node network topology and its performance. These are listed below with a description of their effect on the network performance:

Link probability: This parameter is related to topology construction and defines the number of neighbors any node will have in the DTN topology. For instance, 0.1 link probability gives you a sparsely connected network than a link probability of 0.5. Therefore, higher link probabilities result in a better delivery ratio.

Disconnection periods – Downtime and Uptime of each link: the downtime and uptimes for each link are generated using exponential distributions with a certain mean. For instance, 200/50 sec indicates a downtime mean of 200 sec and uptime mean of 50 sec. A network topology with lower downtime or higher uptime results in a better delivery ratio.

Look Ahead Time (LAT): This parameter is the result of the proposed routing algorithm. Longer look ahead times mean better delivery ratio and fewer transfers with incomplete routes.

Storage capacity per node: In the proposed data delivery mechanism, we use a drop policy whenever there is no storage on the intermediate node. Higher storage means that we can reduce message drops in intermediate nodes.

Workload: In our simulation the workload is expressed in terms of messages per second (mps). Each node generates a Poisson mean number of mps with exponential interarrival times. The destination for the message is randomly picked. Each data transfer is affected by link disconnections along the path, the LAT, and the storage available on intermediate nodes. Each transfer can result in one of three different outcomes: 1) failure to find a complete route because mBFS failed to discover the destination node within the given LAT; or 2) once the quickest path is found between the source and the destination using the proposed mBFS, the message may be dropped due to unavailable storage on intermediate nodes; or 3) the message gets transferred successfully.

Message size: In the current simulation message size is based on exponential distribution.

In the simulation, we assume that the network bandwidth on each link is unlimited. The propagation delay is a trivial amount compared to the link disconnection periods. We choose UDP protocol for each hop transmission in NS2. In our simulation, the DTN environment is devoid of fragmentation caused by a network partition occurring *during* transmission. Therefore, we implement a message to be processed in its entirety as it arrives at each intermediate hop.

Table 1 summarizes the various values of each parameter we used for the simulation.

```

Modified BFS( $G, x, t_{discovered}$ )
1   $F \leftarrow \{x\}$ 
2  While  $F \neq \emptyset$ 
3  Do  $u \leftarrow \text{head}(F)$ 
4  For each  $v \in A[u]$ 
5  Do if  $\text{color}[v] == \text{WHITE}$ 
6  Then  $\text{color}[v] \leftarrow \text{GRAY}$ 
7   $d[v] \leftarrow d[u] + 1$ 
8   $\pi[v] \leftarrow u$ 
9   $d_t[v] \leftarrow t_{discovered}$ 
10  $\text{ENQUEUE}(F, v)$ 
11  $\text{DEQUEUE}(F)$ 
12  $\text{Color}[u] \leftarrow \text{BLACK}$ 

Single-Source Quickest Delivery ( $G, s, t_o, T, \text{Evts}$ )
1  For each vertex  $u \in V[G] - \{s\}$ 
2  Do  $\text{color}[u] \leftarrow \text{WHITE}$ 
3   $d[u] \leftarrow \infty$ 
4   $\pi[u] \leftarrow \text{NIL}$ 
5   $d_t[u] \leftarrow \text{NEVER}$ 
6   $\text{Color}[s] \leftarrow \text{GRAY}$ 
7   $d_t[s] \leftarrow t_o$ 
8  Modified BFS( $G, s, t_o$ )
9  While  $\text{Evts} \neq \emptyset$ 
10 Do  $\text{Evt} \leftarrow \text{DEQUEUE}(\text{Evts})$ 
11  $u \leftarrow u(\text{Evt});$ 
12  $v \leftarrow v(\text{Evt});$ 
13  $t_e \leftarrow t_e(\text{Evt});$ 
14 If  $a(\text{Evt}) == \text{DELETE}$  then
15  $E \leftarrow E - (u, v)$ 
16 Else Do  $E \leftarrow E \cup (u, v)$ 
17 if  $\text{color}[u] \neq \text{color}[v]$ 
18 Then do
19 if  $\text{color}[u] \neq \text{BLACK}$ 
20 then  $\text{swap}(u, v)$ 
21  $d[v] \leftarrow d[u] + 1$ 
22  $\pi[v] \leftarrow u$ 
23  $d_t[v] \leftarrow t_e$ 
24  $\text{Color}[v] \leftarrow \text{GRAY}$ 
25 Modified BFS( $G, v, t_e$ )
    
```

Figure 1. Pseudo-code for the quickest delivery algorithm

Table 1. Parameter values used in the simulation

Simulation parameter	Value
Number of nodes (Nodes)	15
Probability of link connection between any two nodes (LinkProb)	0.1 (low) 0.25 (medium) 0.5 (high)
Number of messages generated per second on each node (mps)	5 10 15
Simulation time (SimTime)	1500sec
Look-ahead-time (LAT)	500sec
Link down interval/Link up interval (DownT/UpT)	200/50sec 150/50sec 100/50sec 50/50sec
Message size (msize)	10KB
Storage amount on each node (Storage)	5MB, 10MB 15MB, 20MB

The performance metrics used in the simulation are:

Deliver Ratio (DR): is defined as the ratio of successful transfers to number of overall transfers. Only complete transfers are taken into consideration for the denominator. Because of the long link disconnections, many transfers will still be in progress within the network at the end of simulation time. We discount those in our calculation of DR.

$$DR = S / (S + N + D)$$

where S is the number of successful transfers, N is the number of no routes, D is the message drops

Number of successful transfers (S): this metric defines the number of complete transfers with storage on intermediate nodes.

Number of No Routes (N): this metric defines the number of transfers that result in incomplete paths to the destination because the mBFS algorithm fails to find a path within the given LAT.

Number of message drops (D): this metric defines the number of transfers failing to complete because of storage unavailability at intermediate nodes.

4.2 Simulation Results

We simulated the network bottleneck environment using workload parameters of messages per second (mps) and storage constraint with some combinations of values described in Table 2. Each simulation run is executed for 1500 seconds. To make the simulation environment stable, we introduced a warm up period. For the proposed algorithm the warm up period was defined as the duration with no message drops. We computed the results presented using 3 to 5 trials with each trial run for 1500 seconds. The associated 95% confidence intervals show that the results are statistically significant. A sample of those is presented in Table 2.

Table 2. Sample Confidence Intervals for Average Delivery Ratio

	5mps	10mps
5MB	0.543 +/- 0.002	0.397 +/- 0.003
10MB	0.721 +/- 0.001	0.529 +/- 0.001

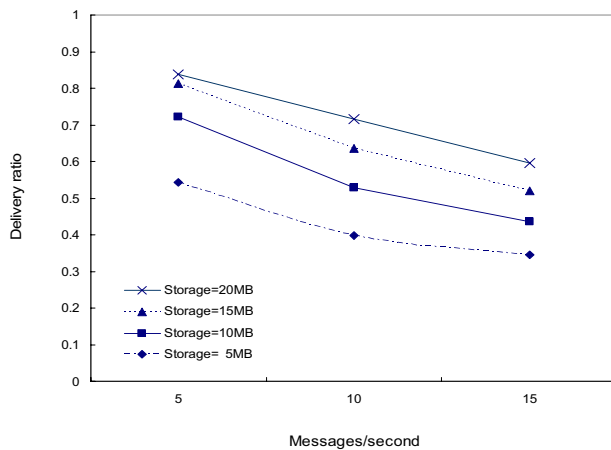


Figure 2. Delivery ratio of the quickest delivery routing algorithm with varying traffic; Nodes=15, LinkProb=0.25, DownT/UpT=200/50second

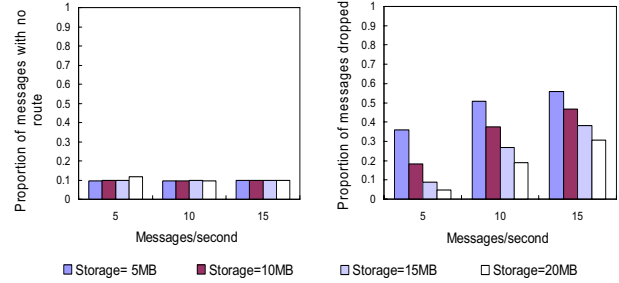


Figure 3. Proportion of messages with no route (left side) or dropped (right side) for varying storage on the traffic of 5, 10 and 15messages/second; Nodes=15, LinkProb=0.25, DownT/UpT=200/50second

The quickest delivery routing performance measured in terms of delivery ratio is affected by the workload, and the storage capacity. We present results of this experiment in Figure 2 where the x-axis shows the messages per second and the y-axis shows the DR for several storage capacities. As expected, we see increased DR as we increase storage capacity because of fewer message drops. The topology related parameters and the LAT are fixed for the four scenarios. The drop policy adopted in this experiment is to drop the oldest message in the queue.

In Figure 3, we present the percentage of transfers that result in “no routes” (graph on the left) and message drops (graph on the right). As can be expected the number of “no routes” remains the same even as we increase the workload (mps) because this metric depends on the LAT and the topology as defined by the link probability and the disconnection periods. As anticipated, we see message drops decreasing with increases in storage.

In Figure 4, we show that the level of link connectivity can be an important factor to achieve a higher delivery rate in our quickest delivery routing algorithm. Bottom portion of each bar shows the

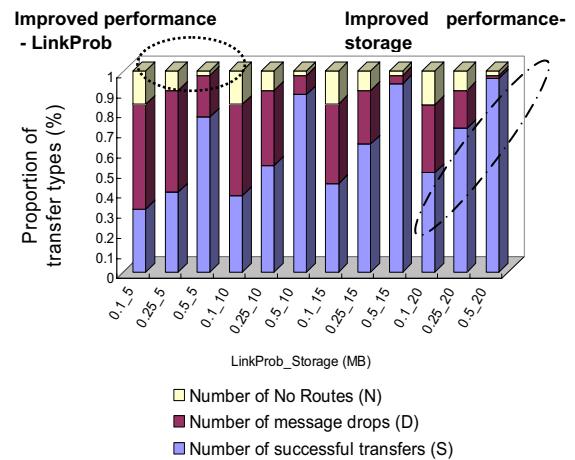


Figure 4. Performance of the quickest delivery algorithm regarding link connectivity; Nodes=15, mps=10messages/second, DownT/UpT=200/50second

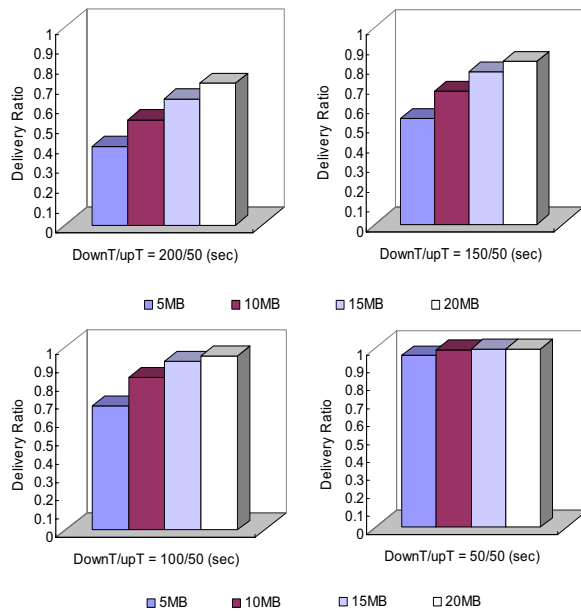


Figure 5. Delivery ratio of the quickest delivery algorithm according to the level of link availability; Nodes=15, LinkProb=0.25, mps=10messages/second

number of messages successfully delivered; the middle portion is the number of dropped messages in the network; the top portion is the number of messages with no route. We simulated three levels of link connectivity such as low (LinkProb=0.1), medium (LinkProb=0.25) and high (LinkProb=0.5). A high link distribution produces a higher delivery ratio because the probability of finding a path in DTN is higher with more connections.

An important factor to consider for the performance of the quickest delivery algorithm is link availability. We evaluated the effect of link availability with downtime (DownT) and uptime (UpT) parameters. The delivery ratio in each plot of Figure 5 is increasing with increased storage as it should. But the absolute amount is getting higher as the duration of link deactivation is getting shorter relative to the interval of link activation. If DownT is equal to UpT as in 50/50sec, delivery rates for each amount of storage are all close to 1.0. This is because the traffic generated during link down time is relatively small that most messages can be transferred during the next link up interval at intermediate nodes.

5. CONCLUSION

In this paper, we presented an algorithmic approach for developing a routing algorithm for DTNs by modifying the well known breadth-first search algorithm. We made simplifying assumptions with justification, such as predictability of link state changes, to bring out the essence of our approach in its simplest form. Through simulation we analyzed the performance of the proposed routing algorithm using appropriate metrics.

6. REFERENCES

- [1] Alonso, J. and Fall, K. A linear programming formulation of flows over time with piecewise constant capacity and transit times. Technical report IRB-TR-03-007, Intel Research Berkeley, July 2003.
- [2] Chen, X., and Murphy, A. L., Enabling disconnected transitive communication in mobile ad hoc networks. In Proceedings of the *Workshop on Principles of Mobile Computing (POMC'01)*, August 2001.
- [3] Delay Tolerant Networking Research Group <http://www.dtnrg.org/>
- [4] Jain, S., Fall, K., and Patra, R. Routing in a delay tolerant network. *ACM SIGCOMM*, Portland, OR, August 2004.
- [5] LeBrun, J., Chuah, C-N, Ghosal, D. Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks. *IEEE 61st Semiannual Vehicular Technology Conference (VTC Spring)*, Stockholm, Sweden, May 2005.
- [6] Musolesi, M., Hailes, S., and Mascolo, C. Adaptive routing for intermittently connected mobile ad hoc networks. *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Taormina, Italy, June 2005.
- [7] Seligman, M., Fall, K., Mundur, P. Alternative custodians for congestion control in Delay Tolerant Networks. In *ACM SIGCOMM Workshops, CHANTS*, Pisa, Italy, September 2006.
- [8] Spyropoulos, T., Psounis, K., and Raghavendra, C. S. Spray and Wait: An efficient routing scheme for intermittently connected mobile networks. In *ACM SIGCOMM Workshops, WDTN*, Philadelphia, PA, August 2005.
- [9] Vahdat, A., and Becker, D. Epidemic routing for partially connected ad hoc networks. Technical Report CS-2000-06, UCSD, July 2000.
- [10] Warthman, F. Delay-Tolerant Networks (DTNs) A Tutorial. <http://www.dtnrg.org/>
- [11] Zhao, W., Ammar, M., and Zegura, E. Message ferrying approach for data delivery in sparse mobile ad hoc networks. In Proceedings of the *3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, Tokyo, Japan, May 2004.