MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks

John Burgess Brian Gallagher David Jensen Brian Neil Levine
Dept. of Computer Science, Univ. of Massachusetts, Amherst, USA 01003
{jburgess, bgallag, jensen, brian}@cs.umass.edu

***Abstract*— Disruption-tolerant networks (DTNs) attempt to route network messages via intermittently connected nodes. Routing in such environments is difficult because peers have little information about the state of the partitioned network and transfer opportunities between peers are of limited duration. In this paper, we propose MaxProp, a protocol for effective routing of DTN messages. MaxProp is based on prioritizing both the schedule of packets transmitted to other peers and the schedule of packets to be dropped. These priorities are based on the path likelihoods to peers according to historical data and also on several complementary mechanisms, including acknowledgments, a head-start for new packets, and lists of previous intermediaries. Our evaluations show that MaxProp performs better than protocols that have access to an oracle that knows the schedule of meetings between peers. Our evaluations are based on 60 days of traces from a real DTN network we have deployed on 30 buses. Our network, called UMassDieselNet, serves a large geographic area between five colleges. We also evaluate MaxProp on simulated topologies and show it performs well in a wide variety of DTN environments.**

I. Introduction

Disruption tolerant networks (DTNs) allow for routing in networks where contemporaneous end-to-end paths are unstable or unlikely. Unstable paths can be the result of several challenges at the link layer, for example: high node mobility, low node density, and short radio range; intermittent power from energy management schemes; environmental interference and obstruction; and denial-of-service attacks. Such environments can exist in undeveloped areas or when a stable infrastructure is destroyed by natural disaster or military efforts. DTNs are useful when the information being routed retains its value longer than the disrupted connectivity delays delivery.

DTNs can be based on moving nodes such as vehicles or pedestrians. Vehicles can provide substantial electrical supplies and transport bulky hardware, which may be inappropriate for use by non-mechanized peers. The disadvantage of a vehicle-based network is that the nodes move more quickly, reducing the amount of time they are in radio range of one another. Accordingly, one limited resource in a vehicle-based DTN is the duration of time that nodes are able to transfer data between one another as they pass. Storage can be a limited resource as well.

This research was supported in part by DARPA contract C-36-B82-S1 and in part by National Science Foundation awards CNS-0519881 and EIA-0080199. The contents of our work are solely the responsibility of the authors and do not necessarily represent the official views of the sponsors. This work has been approved by DARPA for public release; distribution is unlimited.

We offer several contributions in this paper using our deployed DTN as well as simulation environments. First, we propose a DTN routing protocol, called MaxProp, that performs significantly better than previous approaches. Our protocol addresses scenarios in which either transfer duration or storage is a limited resource in the network. MaxProp extends our previous routing work [1] to address several problems that we have observed in our real network topology. Existing approaches have a bias towards short-distance destinations, which MaxProp addresses by using hop counts in packets as a measure of network resource fairness. Additionally, existing approaches fail to remove stale data from network buffers. MaxProp uses acknowledgments that are propagated network-wide, and not just to the source. Finally, MaxProp stores a list of previous intermediaries to prevent data from propagating twice to the same node. While these ideas are simple, our experiments show they significantly raise the delivery rate and lower latency in a wide variety of scenarios as compared to previous approaches.

Our experiments are based on the real mobility and real transfers of the bus-based DTN testbed that we have built, called *UMassDieselNet*. Our network operates daily from the UMass Amherst campus and covers the surrounding county. UMassDieselNet is composed of 30 buses that each contain an HaCom Open Brick computer (P6-compatible 577Mhz CPU, 256MB RAM) powered by the bus's 24V supply. An 802.11b Access Point (AP) is attached to each Brick to provide DHCP access to passengers and passersby. A second USB-based 802.11b interface constantly scans the surrounding area for DHCP offers and other buses. Each bus also has a GPS device attached to the brick. Each brick runs Linux on a 40GB notebook hard drive.

Additionally, we have constructed a simulator that produces simple trace-based synthetic models, which allows us to extrapolate our results. Finally, we have evaluated our protocol with a third synthetic mobility model to ensure comparison in a variety of environments.

This paper is organized around those contributions. In Section II we summarize related work. In Section III we define our protocol. In Section IV we describe our deployed DTN. In Section V we describe the network traces and models that we use in our evaluations, which are presented in Section VI. We conclude in Section VII.

II. Background

Previous work on DTNs has been based on various assumptions regarding connectivity and the availability of environmental knowledge and control.

A number of the proposed routing algorithms for DTNs make few assumptions and are therefore widely applicable. In general, these algorithms are based solely on deciding which messages to forward during a meeting with a given peer and which messages to drop when buffers reach capacity. One epidemic routing algorithm manages finite buffers as first-in-first-out (FIFO) queues [13]. Our Drop Least Encountered (DLE) algorithm [2] proposed dropping messages with the lowest likelihood of delivery, and other papers have followed on this idea [1], [3], [9], [12]. All of these algorithms approximate delivery probability as the likelihood of a delivery path existing. Others have taken a more proactive approach to routing in DTNs, made possible by stronger assumptions such as knowledge of geographic location, prior knowledge of connectivity patterns, and control over peer movement [1], [3], [4], [8], [12], [16], [17], [18], [19], [10], [15].

A few similar deployments have been created for DTN research. Many projects have been focused on bringing Internet connectivity to developing and rural communities. DakNet in Calcutta [11] and the Wizzy Digital Courier in South Africa [14] are two examples. Zebranet [5] provided sensor collection and routing for a herd of zebras and is perhaps the closest work to ours.

III. The MaxProp Protocol

In this section, we present our assumptions, and we detail the protocol itself. There are many environments in which a DTN can operate: on vehicles, pedestrians, zebras, or underwater sensors. The assumptions we make are based on our existing DTN network (see Section IV) composed of buses and desktop computing hardware.

A. Model

We assume that each peer has an effectively unlimited buffer for messages that they originate, but a fixed-size buffer for carrying messages originated by others. We assume that transfer opportunities are limited both in duration and bandwidth. We assume peers have no *a priori* knowledge of network connectivity, no control over their movement, no knowledge of geographic location, and there are no always-on stationary peers in the environment.

In a real network, DTN operations proceeds roughly in three stages.

1) **Neighbor Discovery.** Peers must discover one another before a transfer opportunity can begin; they do not know when the next opportunity will begin.
2) **Data Transfer.** When two peers meet, the amount of data they can transfer is limited. Peers do not know the duration of each opportunity.
3) **Storage management.** As packets are received from a neighbor, each peer must manage its finite local buffer space by selecting packets to delete according to some algorithm. Messages that are destined for a receiving peer are passed up to the application layer and removed from the buffer.

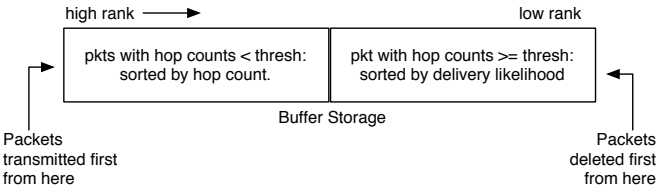

Fig. 1. The MaxProp routing strategy.

Each peer carries all messages until the next meeting occurs. A peer will continue to forward a message to any number of other peers until its copy of the message times out, it is notified of delivery by an ack, or the message is dropped due to a full buffer.

Bandwidth can be a limited resource when the transfer opportunities are of short duration (which may be the result of a slow neighbor discovery protocol) or when the radio used has low bandwidth. Storage can be limited when such devices as motes, PDAs, or cell phone are used. Offered message load can affect both storage and bytes transfered.

B. Protocol Definition

The MaxProp protocol uses several mechanisms in concert to increase the delivery rate and lower latency of delivered packets.

MaxProp uses several mechanisms to define the order in which packets are transmitted and deleted. Figure 1 illustrates these mechanisms. At the core of the MaxProp protocol is a ranked list of the peer's stored packets based on a cost assigned to each destination. The cost is an estimate of *delivery likelihood*. In addition, MaxProp uses acknowledgments sent to all peers to notify them of packet deliveries. MaxProp assigns a higher priority to new packets, and it also attempts to prevent reception of the same packet twice. The remainder of this section presents the details of destination cost estimation, our other mechanisms, and buffer management.

1) Estimating Delivery Likelihood: Previous work has demonstrated that optimal delivery paths in a DTN can be discovered by constructing a directed graph of nodes connected by edges representing traversals through time and space [4]. A variation of Dijkstra's algorithm can determine the shortest path, if one exists. In practice, no oracle is available to reveal future connections. MaxProp therefore assigns link weights as follows.

Let the set of nodes in the network be s. Each node, $i \in s$, keeps track of a probability of meeting peer $j \in s$. We estimate this probability f_j^i as the likelihood that the identity of the node we connect to next will be j. For all nodes, f_j^i is initially set to $1/(|s|-1)$. When node j is encountered, the value of f_j^i is incremented by 1, and then all values of f are re-normalized. Using this method, often called *incremental averaging*, nodes that are seen infrequently obtain lower values over time. In

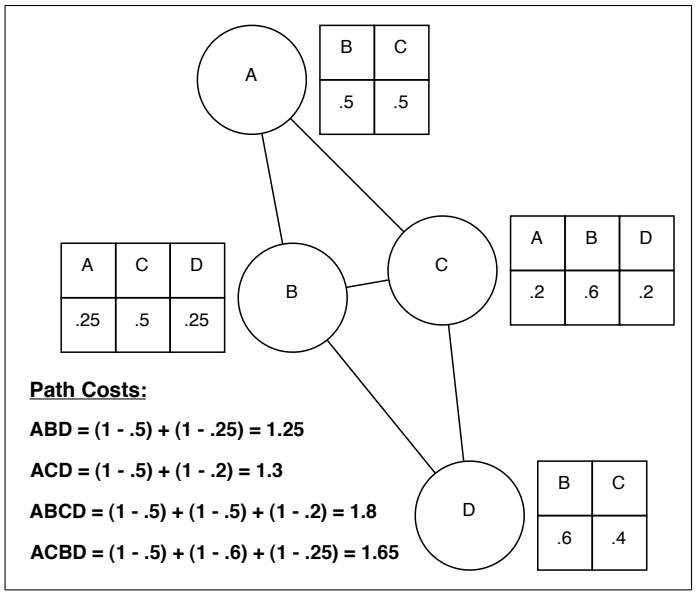

Fig. 2. MaxProp Path Cost Calculation

MaxProp, each time two peers meet, they exchange these values with one another.

For example, for a DTN with four other nodes, a peer j has values for $f_1^i = f_2^i = f_3^i = f_4^i = 0.25$. Upon encountering node 3, the peer sets $f_3^i = 1.25$ and re-normalizes all values so that they sum to 1 again: $f_1^i = f_2^i = f_4^i = 0.125$ and $f_3^i = 0.625$.

With other nodes' values in hand, a local node calculates a cost, $c(i, i+1, \ldots, d)$, for each path possible to the destination d, up to n hops long. The cost for a path using nodes $i, i+1, \ldots, d$ is the sum of the probabilities that each connection on the path does not occur, estimated as one minus the probability that each link does occur:

$$c(i, i+1, \ldots, d) = \sum_{x=i}^{d-1} [1 - (f_{x+1}^x)].$$

The cost for a destination is the lowest path cost among all possible paths. Figure 2 illustrates an example of this policy where the cost from A to D is determined as the minimum value, 1.25.

In practice, this calculation among all possible paths is fast because paths monotonically increase in cost during a depth-first search. Once the cost for a path is worse than the current best path, the search can stop. In our evaluations, we set the maximum path length to search as 10.

Packets that are ranked with highest priority are the first to be transmitted during a transfer opportunity. Packets ranked with lowest priority are the first to be deleted to make room for an incoming packet. When two packets have destinations with the same cost, the tie broken by giving the packet that has traveled fewer hops higher priority.

2) Complementary Mechanisms: Unlike other protocols, MaxProp involves several other mechanisms beyond this core that increase the delivery rate and reduce latency, as our evaluations show in Section VI.

When two peers discover each other, MaxProp exchanges packets in a specific priority order.

- First, all messages destined to the neighbor peer are transferred.
- Second, routing information is passed between peers; specifically, a vector listing estimations of the probability of meeting every other node, as explained above.
- Third, acknowledgments of delivered data are transfered, regardless of source and destination. An acknowledgment consists of the cryptographic hash of the content, source, and destination of each message, and is therefore about 128 bits. This mechanism serves to clear out buffers in the network of old data at little cost if the acknowledgment are small compared to data packets. In our evaluations, peers do not spend more than 1% of the historical average connection duration on sending acknowledgments.
- Fourth, packets that have not traversed far in the network are given priority. We found in simulations that estimating delivery likelihood can favor packets that have a high chance of reaching a destination, causing some packets to never get a chance at being propagated. Therefore, MaxProp attempts to give new packets a head start in the network by placing them at a higher priority. The effect of this approach is that newer packets are transmitted at several transfer opportunities when they are first generated, increasing their chance of reaching the destination. To implement this strategy, MaxProp logically splits the buffer in two according to whether the packets have a hop count less than a threshold t hops. Packets below the threshold are sorted by hopcount; packets above are sorted by the scoring mechanism described above.

  Setting the threshold statically would be arbitrary and would not work for all environments. MaxProp takes an adaptive approach to setting the threshold.

  In environments where the average number of bytes transfered per transfer opportunity, x, is much smaller than the byte size of buffer, b, we prioritize low-hopcount packets. As x grows, we slowly reduce the threshold to the difference between the two values. When x is larger than the buffer size, then we remove the threshold completely — it is no longer needed.

  Specifically, after each transfer opportunity, we re-evaluate the threshold by first choosing a portion of the buffer p as follows:

  - If $x < b/2$, then $p = x$
  - If $b/2 \leq x < b$ then $p = \min(x, b - x)$
  - If $b < x$ then $p = 0$.

  If we use p as a threshold, we would be arbitrarily partitioning packets of the same hop count; therefore we set t to be the minimum hop count that selects packets included in p (and perhaps more). We evaluate this algorithm in Section VI.
- Fifth, the remaining, untransmitted packets are sent in

an order based on the scores described in Section III-B.1.

- Finally, we note that in all cases, packets that have already been sent to the node are not sent again. A *hop list* in each packet stores peers that the packet has already traversed, including peers to which the current node has sent the packet. (A similar algorithm is used in Usenet/NNTP to limit flooding [6].)

MaxProp keeps copies of packets that it has passed on to other peers. In environments where intermediaries are fully reliable and all routes are known — such as the interplanetary transmissions handled by DTNRG for NASA — this is wasteful. But for UMassDieselNet, and similar DTNs, this is a strategy we find effective in our evaluations.

3) Managing Buffers: The difference between managing limited storage and limited transmission is that packets that are sent in one transfer opportunity may be sent in the next opportunity. In contrast, if a packet is dropped from a buffer, it may never be delivered.

There are only three reasons a peer p can drop a packet, m, without reducing the overall delivery rate of the network unnecessarily.

- Criteria 1: A copy of message m has already been delivered to its destination.
- Criteria 2: No route with sufficient bandwidth will exist between p and m's destination during the lifetime of message m.
- Criteria 3: No copy of m has been delivered, but some copy of m will be delivered even if peer p drops its copy.

It is easy to show that these three criteria are necessary and sufficient. First, the three criteria are mutually exclusive; it is not possible for any message to meet more than one of them. Second, the only possibility not covered is that m has not been delivered but can only be delivered if p holds on to m. Clearly, dropping this type of message will affect the overall delivery rate.

Since the propagation of information in a DTN is relatively slow, a peer will generally not know the values of the three criteria with certainty.

To estimate whether Criteria 1 has been satisfied, we use acknowledgments sent from the destination and propagated to all peers in the network. Although this information can be delayed, it will never be inaccurate once received.

To estimate Criteria 2, we use the scoring mechanism from Section III-B.1.

Criteria 3 is the most difficult to estimate. As a weak estimator, we use hop count. Because packets are copied from one peer to another (without being deleted from the first), packets that have propagated further within the network are given lower priority for routing and are dropped first. These packets are most likely to have already been delivered; as we show in Section VI, this is a good approach.

In sum, MaxProp deletes acknowledged packets immediately, followed by packets that have reached the threshold of t hops with poor scores assigned to the packet's destination, followed by packets with the most hops below threshold t.

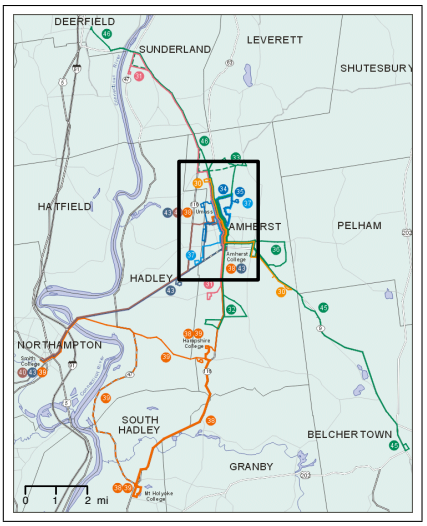

Fig. 3. Routes traveled by the UMassDieselNet buses. The black rectangle shows the area covered by the photo in Fig. 5.

IV. UMassDieselNet

The end goal of any systems project is a deployed implementation. Simulations and analysis can predict trends of performance, but real systems expose problems that may not be obvious on paper. For example, models of node movement and connectivity are only approximations of what can happen in real life systems. This is critical as the movement and communication patterns of peers is likely to affect the relative performance of protocols. In fact, mobility research to date has been criticized for having too many oversimplifying assumptions [7].

We have constructed a DTN testbed composed of 30 buses operated by the UMass Amherst branch of the Pioneer Valley Transport Authority (PVTA) that we have fitted with a custom package of off-the-shelf hardware. This testbed is called *UMassDieselNet*. The transit buses service an area sparsely covering approximately 150 square miles. Figure 3 shows the exact geographical coverage of the buses. All routes intersect in Amherst, Massachusetts; Fig. 5 shows the density of bus-to-bus connection opportunities during a one month period in that geographic area.

In this section, we describe our setup and present the characteristics of our testbed from which we took traces to evaluate our protocol and also to construct a synthetic trace generator.

A. Current Backbone Testbed

The testbed began operating in May 2004 with five buses. Each bus carries a HaCom Open Brick computer (P6-compatible 577Mhz CPU, 256MB RAM). An 802.11b Access Point (AP) is attached to each brick to provide DHCP access to passengers and passersby. A second USB-based 802.11b interface constantly scans the surrounding area for DHCP offers and other buses. Each bus also has a GPS device attached to the brick. Each brick runs Linux on a 40GB

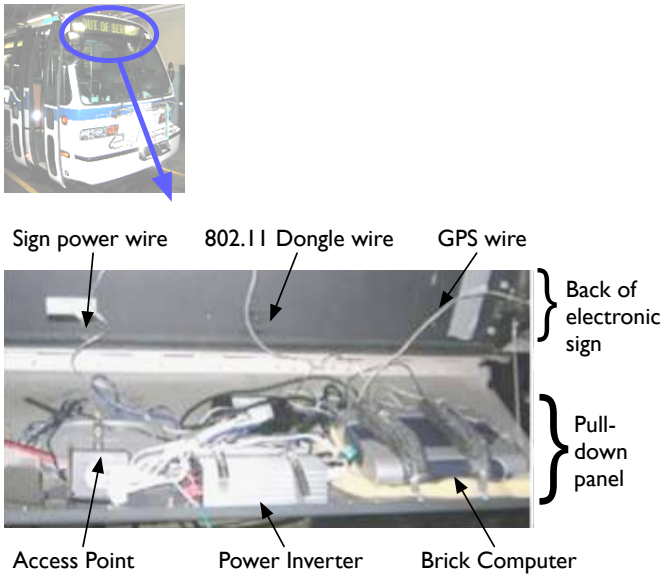

Fig. 4. Photos of a deployed bus peer.

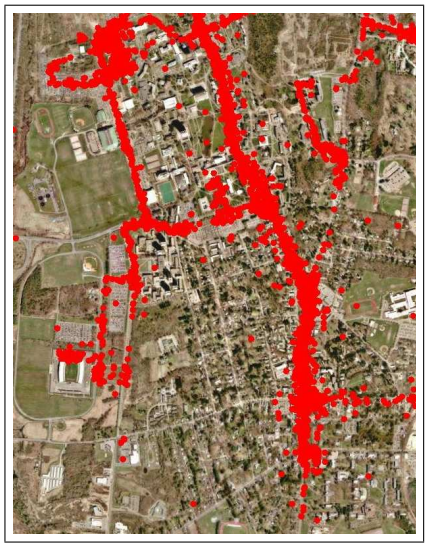

Fig. 5. Each points represents one in-motion bus-to-bus transfer during March 23–April 23 overlaid on an aerial photo (roughly 4 sq. mi.). (Only viewable in color.)

notebook hard drive. Figure 4 shows a photo of a system, which are installed behind the electric sign.

To enable bus-to-bus transfers, the buses beacon on a single channel once every 100ms. We programmed the bricks to transfer the largest amount of data possible using TCP at each transfer opportunity. The figures show the results from our testing of bus-to-bus in-motion transfers, including the PDFs of data transfered (Fig. 6), transfer opportunity duration (Fig. 7), and inter-transfer opportunity time (Fig. 8), from February 1 until May 10, 2005. The best fit to log-normal are shown only for comparison in Figures 6 and 7. These statistics are not simple because this is a real system that operates according to many factors that affect bandwidth,

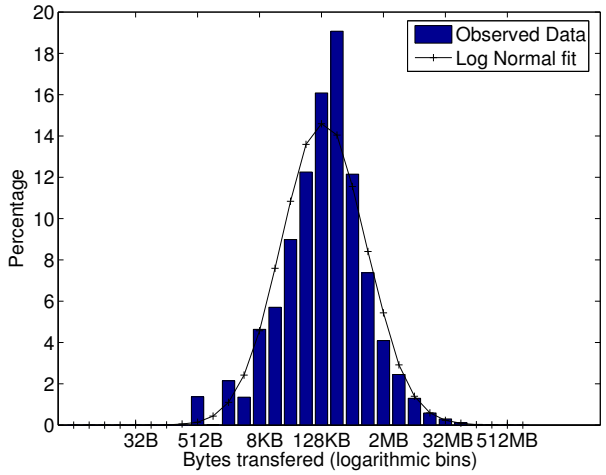

Fig. 6. PMF of bytes transfered at each transfer opportunity (Feb 1–May 10, 2005). Avg: 1.2 MB.

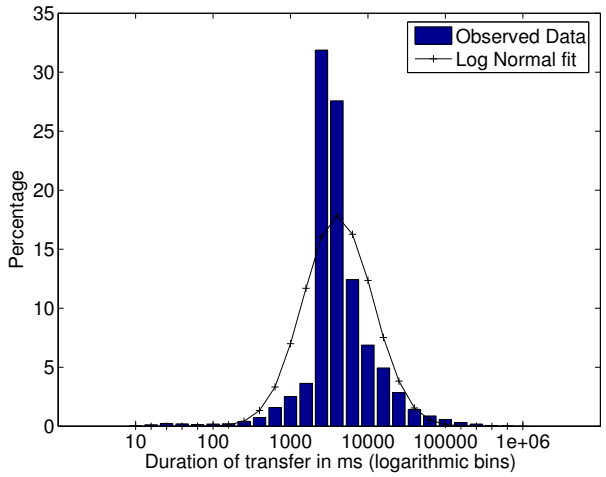

Fig. 7. PMF of transfer opportunity duration (Feb 1–May 10, 2005). Avg: 10,209 ms.

movement, density, and other characteristics. Because they are based on a real system, they give us confidence in our trace-drive evaluations, presented in the next sections.

To maintain and monitor our network, we use numerous external APs that offer free service along the bus routes hosted by third parties. We have installed only two APs — one on campus and one at the bus garage. Whenever the buses have web access, they retrieve software updates from a central server. At that time a bus provides its current GPS location and MAC address, and it uploads logs of its performance during the day, including the throughput of bus-to-bus transfer opportunities, APs contacted, a record of movement, and application records.

V. Network Traces and Models

Real networks change topologies, link latencies, and available bandwidth throughout the course of their usage. DTN routing protocols must be adaptable enough to continue operating effectively under these evolving network conditions. Our

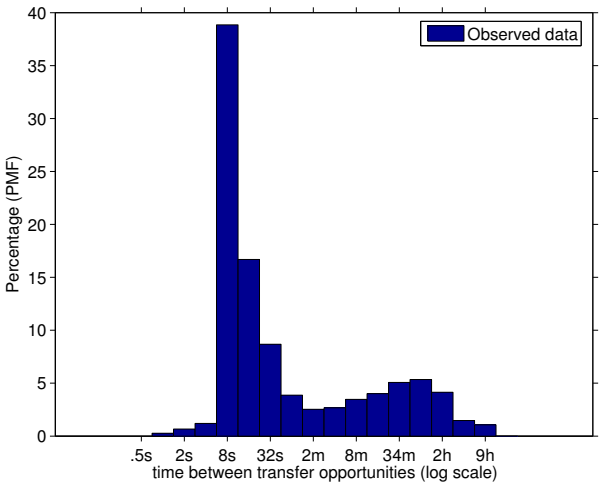

Fig. 8. PMF of the time between transfer opportunities (Feb 1–May 10, 2005). median: 11 sec.

main goal is to design an algorithm that works best with the bus-based DTN we have deployed in Amherst. However, we do endeavor to design the most general algorithm.

Accordingly, our protocol evaluations make use of the traces as well as two synthetic topologies of intermittently connected peers. The first is a program that simulates the movements and transmissions of buses but allows us to increase the number of buses appearing on each route. The second is completely synthetic, based on a simple algorithm governing connections. This section describes the properties of each of these three models.

A. UMassDieselNet Traces

To allow us to easily test different routing algorithms in a real DTN environment, we set the UMassDieselNet buses to transmit random data to one another whenever they are within range and record the time, transmission size, and buses involved. Our testbed is subject to the real schedule of the UMass campus — many fewer buses run on weekends and holidays. To realize some uniformity, we used 60 traces from weekdays between January 25, 2005 to May 6, 2005; we excluded holidays and other occasions causing buses to run infrequently.

On average, about 28 buses are active each day, and each days lasts from about 7AM to 7PM. In all, the traces consist of over 720 hours of recorded data for each of the 30 buses (about 20,000 bus-hours total). Each data point of our graphs in the next section represents a single protocol evaluated using this amount of trace data three times, each with a different seed for randomly generated packets.

The route each bus is placed on each day is chosen by the garage dispatcher and can change during the day. Unlike synthetic simulations, buses can leave the network at any time. We did not try to automatically determine the routes of buses, though this is possible with some significant effort. We decided against this approach after finding GPS data often inconsistent or containing gaps where line-of-sight to satellites was lost.

In our simulations, we generated packets from each bus between 2 and 18 packets per hour. Packets are destined for other buses that are currently in the network, but there is never a guarantee that the destination bus will remain for any period of time — we treat such packets as failures. We varied buffer storage available in each bus between 50 and 480,000 packets of 10K each (i.e., 500K to 4.8G drives). Finally, we varied the size of packets from 10K to 100K. For transmission bandwidth at each opportunity, we used the real trace data.

B. Synthetic Bus Traces

Routing experiments usually require that certain operational parameters remain constant as others are varied to reveal the true causes of observed phenomena. The inconsistent operation of UMassDieselNet nodes makes such experiments a challenge because few parameters can be held constant from day to day.

To this end, we developed a simple trace generator using GPS data to reconstruct the movement and timing patterns of nine major bus routes around the university. Each route is based on one representative trace day that had accurate GPS data. This simulator allows us to have all buses operating in a predictable fashion each simulation run while varying such factors as radio range, transmission bandwidth, buses per route, and schedule variance.

C. Virtual DTN Networks

UMassDieselNet effectively demonstrates routing performance in one actual operational environment, however, because of UMassDieselNet's topology and specific operational environment, its routing performance may not be characteristic of other DTNs.

We created a virtual DTN that models the connectivity between peers according to a simple algorithm. In the model, each peer p has a small set of preferred peers that it meets with often and fairly regularly over the course of a simulation. Peer p meets with the remaining set of peers infrequently or not at all. These networks are intended to produce varied patterns of linkage over different pairs of peers that remain relatively consistent over time. The expectation is that this will give the protocols a regular pattern to learn from.

We create these small-diameter networks as follows. Each pair of peers in the network has a link connecting them. For each link, we draw a meeting count, c, at random from an exponential distribution with mean λ. For any links with $c < \lambda$, we reset $c = 0$. This creates fewer direct meetings between peers and reduces the number of one-hop deliveries in the network. Let s represent the duration of the simulation. We then calculate an inter-meeting time, $i = s/c$, for each link independently. The actual times between meetings during the simulation are drawn randomly from a Poisson distribution with a mean of i.

To determine the bandwidth transfered between the peers in this model, we selected a value from the bus traces uniformly at random.

VI. Evaluation

The primary goals of any DTN routing protocol is to maximize the delivery rate of offered packets and to minimize the latency between source and destination. We evaluated our protocol in the context of other routing algorithms and in three topological environments. We found in evaluations varying offered load, buffer capacity, and individual packet size that our protocol delivered more packets with lower latency than other protocols.

A. Protocol Comparisons

We compared MaxProp against several other approaches: an oracle-based Dijkstra algorithm [4], an expanded version of our previous work called Drop-Least Encountered [1], [2], and random routing as a baseline for performance. We detail each algorithm below.

• **Oracle-based Dijkstra [4].** This algorithm cannot be reproduced in practice because of the use of an oracle; however, MaxProp was able to perform better in our trace-based evaluations. The algorithm is an adapted version of Dijkstra's shortest-path algorithm that works with time and space. It determines which packets are likely to route through a peer with the lowest latency. The algorithm uses the exact times of the future meetings with other peers to construct a directed network graph.[1] Vertices in the graph represent connection events, and direct links between edges order the events. Links are weighted with the delay before the event occurs. Peers may buffer a packet until any future link event, giving the graph a highest branching factor closest to the present node. The standard Dijkstra algorithm then determines shortest delay paths by traversing the graph.

This strategy algorithm still falls short of being globally optimal because the oracle does not have knowledge of the buffer space and link bandwidth allocations of all peers at all times in the future. Effectively, each node makes routing decisions independently, without consideration of other peers packets which will be traversing the links and buffers simultaneously. With this schedule coordinating information, computational complexity for optimal route determination of all packets becomes an NP-complete problem which can be solved via a dynamic programming algorithm [4].

• **Most Encountered/Drop Least Encountered** This protocol consists of only the scoring mechanism described in Section III-B.1 to order packets. Packets with the highest scores have destinations that are Most Encountered are the first to be transmitted, and packets with the lowest scores are Dropped because they are Least Encountered — hence the name ME/DLE. This protocol does not use complementary mechanisms described in Section III, including acknowledgments, hop lists, and higher priority for young packets. It can therefore be used as a measure of how well our complementary mechanisms perform. From another viewpoint, it is an extension of our previous work, the Drop Least Encountered [1], [2] protocol to include a mechanism for scheduling packets at transfer opportunities. It therefore shows how well our newest protocol performs in comparison.

Random. This protocol chooses messages uniformly at random to be transmitted or discarded as necessary at each connection event.

[1]Not surprisingly, using the planned bus schedule instead of exact times performs considerably worse than the oracle; those results are not shown here.

B. Experiments

Our simulations produced three sets of results using the traces of the buses for determining when transfer opportunities occur and for how much bandwidth is transfered at each.

First, we analyzed three specific scenarios:

1) Delivery rate and latency when offered load varies from 2 to 18 pkts/hour on each bus (with fixed buffer size and packet size). These values represent the means of exponentially distributed packet loads.
2) Delivery rate and latency when buffer size varies from 500KB to 5MB on each bus (with fixed offered load and packet size).
3) Delivery rate and latency when packet size increases from 10KB to 100KB (with fixed buffer size and offered load).

Then using synthetic traces, we analyzed how the protocols performed when the the number of buses in the network varied, and how the protocols performed with different radio ranges.

Finally, we analyze performance again using the virtual DTN topology described in the previous section.

For all simulations, we calculated a paired t-test of MaxProp against each other protocol. Since the packet load and buses were the same in each simulation run, we paired each node for comparisons. (Standard deviation is not appropriate since each node achieves vastly different values depending on their route in the simulation.) Our results, though not plotted, show that the means and medians for MaxProp that are plotted in our graphs are significantly different than the other protocols.

C. Results

Figures 9 and 10 show the delivery rate and median latency, respectively, of each protocol as offered load in the system increases from 2 pkts/hour to 18 pkts/hour. With this increase in load, we can see a drop in delivery rate for all protocols. However, for all scenarios, MaxProp delivers more packets and maintains smallest latency. What is striking is that MaxProp outperforms oracular Dijkstra even though the latter protocol knows exactly when the next meeting takes place. Note in these simulations the large 40GB buffer on the peers (the same as the real buses) result in no dropping at the nodes due to filled buffers.

Figures 11 and 12 show the delivery rate and median latency, respectively, of each protocol for storage resources that are much less than what we have on the buses. In these experiments, we allows peers to carry between 50 and about 500 packets, which is 500KB to 5MB for the 10k-sized packets we used. MaxProp shows its versatility in these experiments as it maintains its performance advantage over oracular Dijkstra after a buffer size of just 1MB, and always performs better

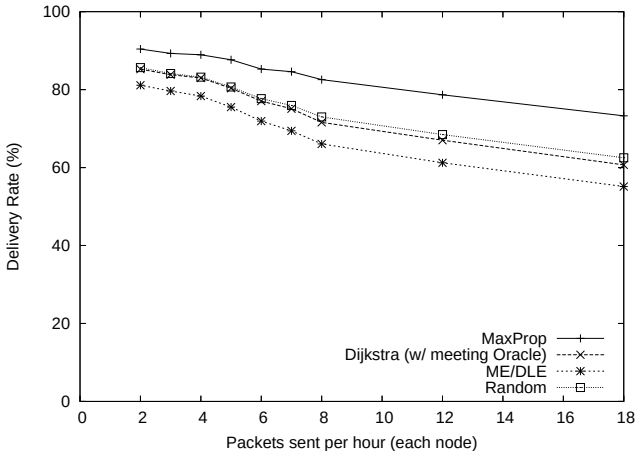

Fig. 9. Delivery rate.

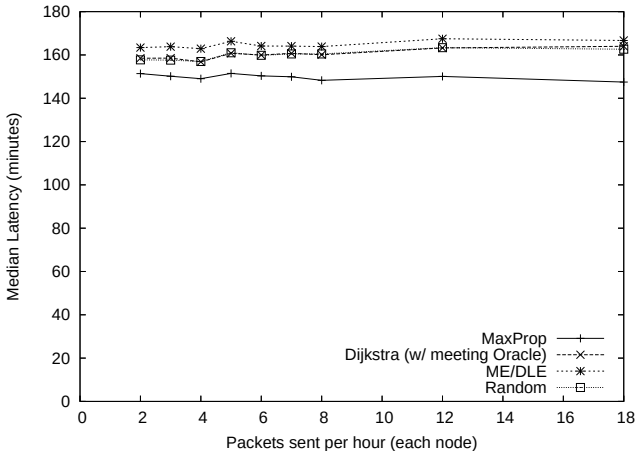

Fig. 10. Median latency of delivered packets.

Performance when offered load varies. (10k packets, 40GB Buffer; real buses bandwidth and mobility.)

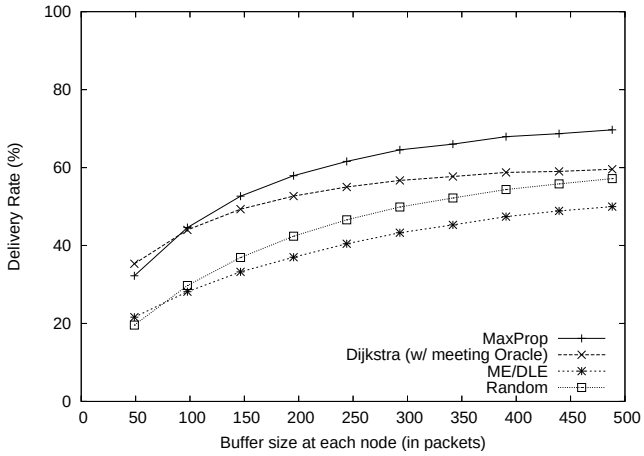

Fig. 11. Delivery rate.

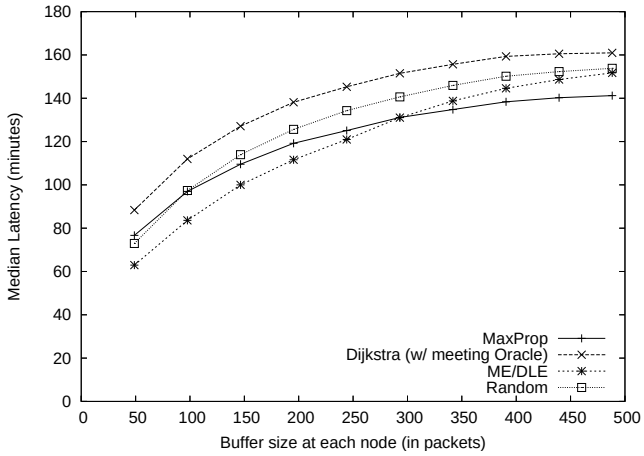

Fig. 12. Median latency of delivered packets.

Performance for small storage sizes. (18pkts/h, 10k packets, real bus bandwidth and mobility.)

than random and ME/DLE. Recall the ME/DLE is equivalent to MaxProp without the complementary mechanisms described in Section III-B.2 — this experiment demonstrates their effectiveness. Again we see that latency of MaxProp is either best or second to ME/DLE; however, the latter protocol delivers a significantly smaller percentage of packets.

Noticeably, ME/DLE performs better than random only for small buffers. In our previous work on the DLE protocol [2], [1] we evaluated only buffer sizes below 50 packets (with infinite bandwidth at each transfer opportunity). However, these experiments show that performance of such an algorithm is poor for more resourceful peers.

In our evaluations we found that Dijkstra and ME/DLE overly favor certain links, causing congestion in the network. Random is able to perform better because it does not send all data over a small number of links. MaxProp is also able to take advantage of a wider set of opportunities in the network by favoring new packets, sending them along links that other protocols would not use. Additionally, MaxProp's use of acks and hoplists also makes better use of buffers and transfer opportunities.

Figures 13 and 14 show the delivery rate and median latency, respectively, of each protocol as the size of the packets increases from 10KB to about 100KB. Increasing packet size models different applications (e.g., small email messages versus large document and image transfers). Larger packets make it difficult for peers to make use of all transfer opportunities, some of which are too short in duration. Here again we see that MaxProp is able to perform better than other protocols.

D. Evaluations of MaxProp Components

There are several components to the MaxProp protocol. Figure 15 separates the performance of each. As above, Random denotes a strategy where packets are transmitted and deleted in a random order. Random with Hoplists adds only the mechanism to thwart duplicate reception of the same packet by a peer. Random with Acks adds only the acknowledgments to random, allowing deletion of delivered packets. ME/DLE is equivalent to the scoring mechanism of MaxProp. Finally,

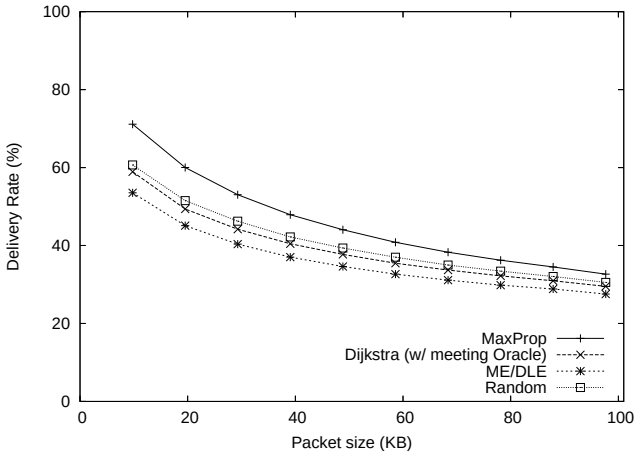

Fig. 13. Delivery rate.

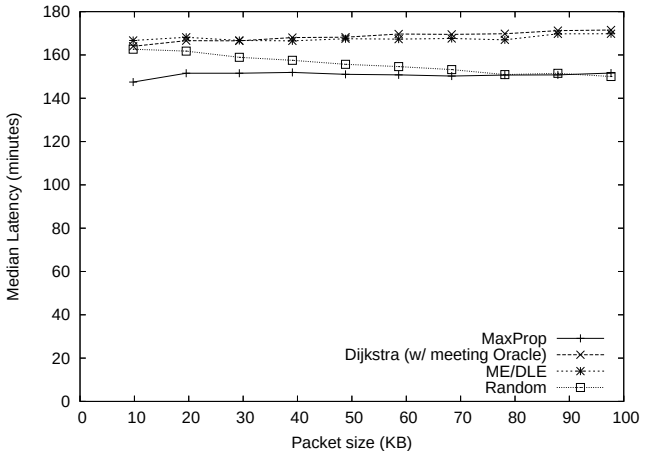

Fig. 14. Median latency of delivered packets.

Performance as packets increase in size. (18pkts/h, 40GB buffer, real bus bandwidth and mobility.)

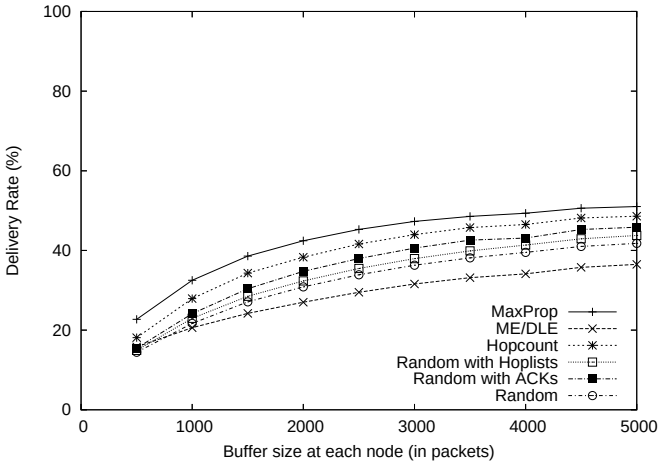

Fig. 15. The components of MaxProp evaluated separately.

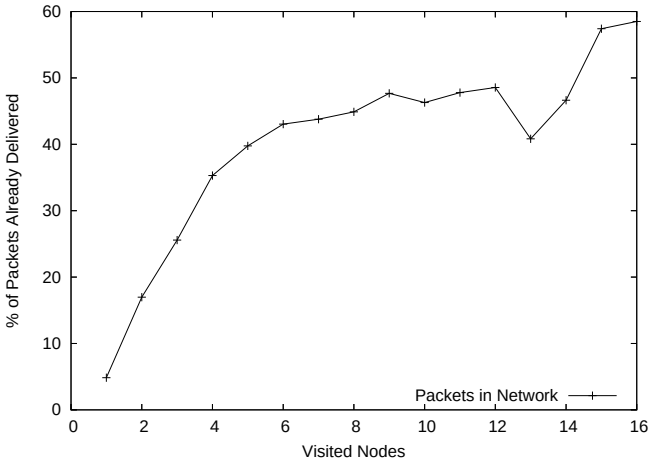

Fig. 16. Observed packets delivery status and hop counts.

Hopcount sorts the buffer by distance traveled for prioritizing transmission and deletion. Packets with the same count are chosen randomly. The figure shows a simulation where 10KB-size packets are used with mean load of 18pkts/hour on each bus. MaxProp performs best across a range of buffer sizes since these mechanisms are complementary.

We were also interested in the quality of hop count as estimator of whether a packet has been delivered by another node, as we postulated in Section III-B.3. Figure 16 shows the hop counts of packets at each node sampled at each connection event. The percentage of packets which have already been delivered are displayed on the y-axis. These results help demonstrate that MaxProp is correctly exploiting hopcount information to better prioritize packets that have likely not yet been delivered.

E. Examining Other Parameters

As we discussed above, using synthetic bus traces we are able vary other network parameters. Figures 17 and 18 show the delivery rate and latency, respectively, of simulated traces when the number of buses servicing each route varies. We can observe from these figures that delivery probability remains roughly constant after enough peers are introduced for frequent connection events. This balance reflects increasing load (produced by new peers) offset by increasing numbers of peers to facilitate packet delivery. Latency sharply decreases as available delivery paths increase. The relative performance of the protocols is exactly that of Figures 11 and 12. Even though the experiments are from multiple synthetic traces, characteristics specific to our bus routes manifest themselves in performance results; specifically, median latency jumps by 10% when each of the nine routes has five buses.

Another experiment we could not perform with the real UMassDieselNet was varying radio transmission range. Figures 19 and 20 show the delivery rate and latency, respectively, of simulated traces with varying radio transmission ranges. In a very simple approach, we determined bandwidth multiplying the length of time peers are in radio range by the mean transfer rate reported in the bus data, which is 120 KB/s.

Increasing the peers' radio range shows no gain in delivery

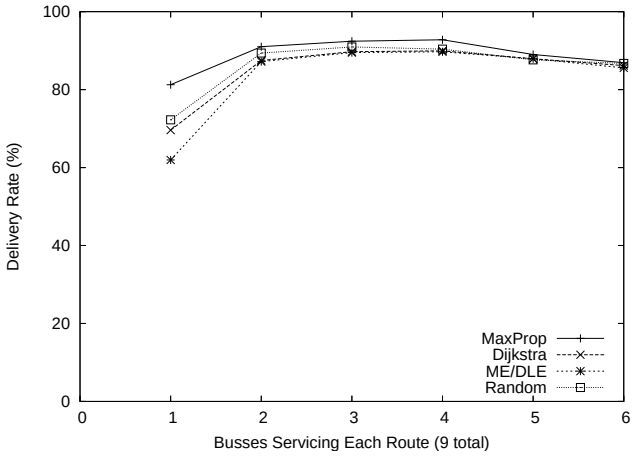

Fig. 17. Delivery rate.

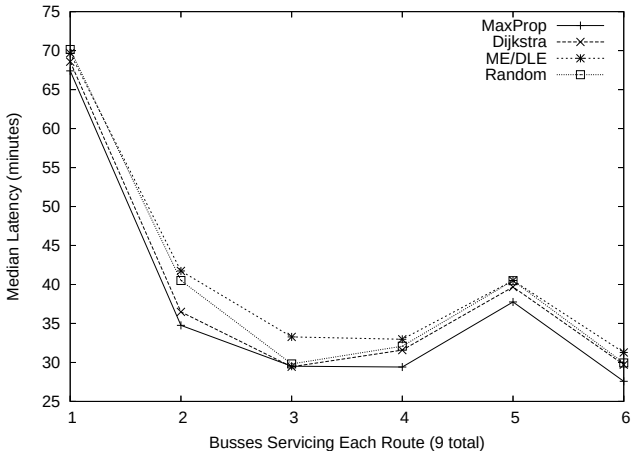

Fig. 18. Median latency of delivered packets.

Performance as buses servicing each route increases. (18pkts/h, 40GB buffer, simulated bus bandwidth and mobility.)

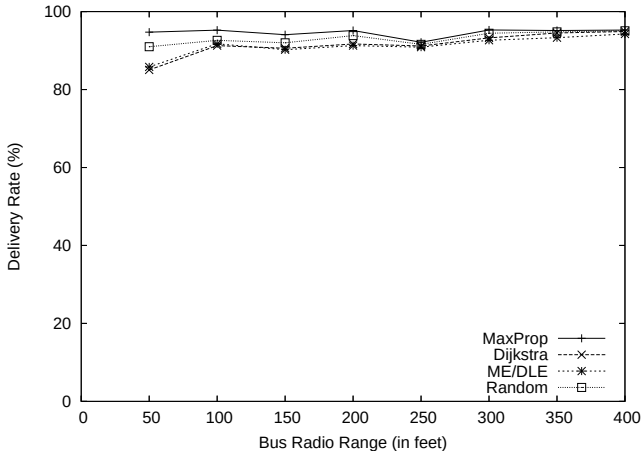

Fig. 19. Delivery rate.

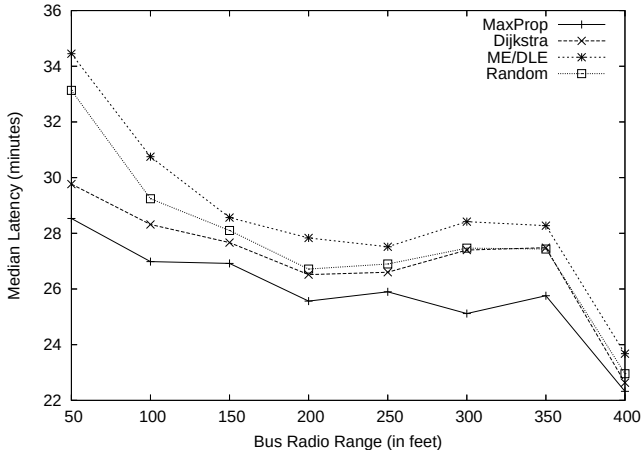

Fig. 20. Median latency of delivered packets.

Performance radio range of peers increase. (18pkts/h, 40GB buffer, simulated bus bandwidth and mobility.)

rate as buses have enough buffer available to store and forward most packets they receive. Latency, on the other hand, steadily decreases as buses are able to contact additional peers and transfer additional data at each opportunity.

F. Virtual DTN Simulation

Figures 21 and 22 show the delivery rate and latency, respectively, of a simulated trace of 30 peers using the virtual DTN described in Section V-C. Each of 20 virtual topologies was simulated for 1,440 minutes through 3 simulations, resulting in 1,400 hours simulated per data point. Dijkstra's foreknowledge of events in these unusual topologies give it an edge over MaxProp as statistical prediction of peering events works less reliably. Despite operation outside a vehicle-based DTN, MaxProp still maintains good performance.

VII. Conclusion

We have proposed MaxProp as an effective protocol for DTN routing, particularly for the context of our real DTN deployment. MaxProp unifies the problem of scheduling packets for transmission to other peers and determining which packets should be deleted when buffers are low on space. Additionally, we have identified several complementary mechanisms for improving the performance of path-likelihood based routing, including: system-wide acknowledgments, hoplists denoting previous intermediate recipients, and priority for new packets using an adaptive threshold.

Unlike many other previous works, our evaluations are based on traces of the actual movements and TCP transfers over 802.11 of a network of 30 buses. UMassDieselNet operates everyday from our campus garage has provided us with months of trace data.

Our evaluations show that MaxProp performs better than even protocols that have access to an oracle that knows the schedule of meetings between peers. Our evaluations also examine MaxProp in simulated topologies to show it performs well in a varied DTN environments.

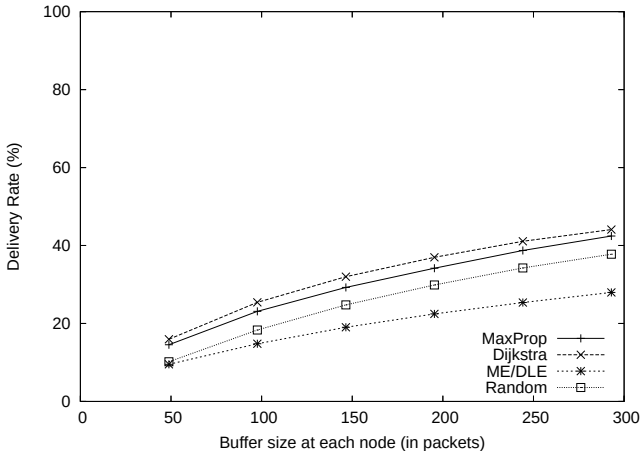

Fig. 21. Delivery rate.

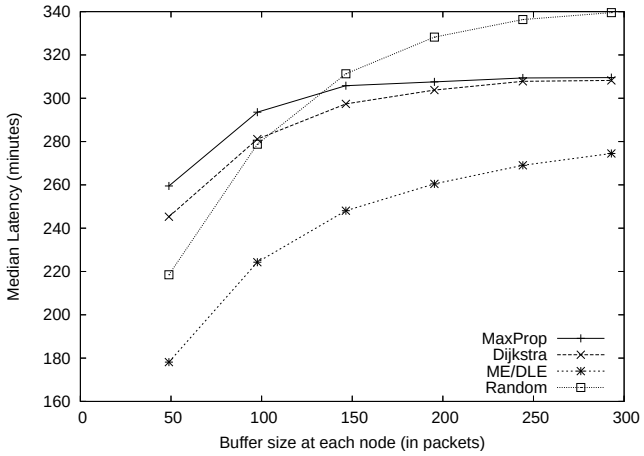

Fig. 22. Median latency of delivered packets.

Performance within an artificial DTN topology. (18pkts/h, 40GB buffer, real bus bandwidth and mobility.)

VIII. Acknowledgments

We are grateful to Tom Caron and Adam Sherson of the UMass Transit branch of the Pioneer Valley Transportation Authority for assisting in the creation of UMassDieselNet.

References

[1] B. Burns, O. Brock, and B.N. Levine. *MV* routing and capacity building in disruption tolerant networks. In *Proc. IEEE INFOCOM*, pages 398–408, March 2005.

[2] J. Davis, A. Fagg, and B.N. Levine. Wearable Computers and Packet Transport Mechanisms in Highly Partitioned Ad hoc Networks. In *Proc. IEEE Intl. Symposium on Wearable Computers*, pages 141–148, October 2001.

[3] M. Grossglauser and M. Vetterli. Locating Peers With Ease: Mobility Diffusion Of Last Encounters In Ad hoc Networks. In *Proc. IEEE Infocom*, April 2003.

[4] S. Jain, K. Fall, and R. Patra. Routing in a Delay Tolerant Network. In *Proc. ACM SIGCOMM*, pages 145–158, August 2004.

[5] P. Juang et al. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. *SIGOPS Oper. Syst. Rev.*, 36(5):96–107, 2002.

[6] B. Kantor and P. Lapsley. RFC 977: Network News Transfer Protocol, February 1986.

[7] D. Kotz, C. Newport, R. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *Proc. ACM/IEEE Intl Symp on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 78–82, October 2004.

[8] Q. Li and D. Rus. Sending Messages to Mobile Users in Disconnected Ad hoc Wireless Networks. In *Proc. MobiCom*, pages 44–55, August 2000.

[9] A. Lindgren, A. Doria, and O. Scheln. Probabilistic Routing in Intermittently Connected Networks. In *Proc. Workshop on Service Assurance with Partial and Intermittent Resources*, August 2004.

[10] S. Merugu, M. Ammar, and E. Zegura. Space-time routing in wireless networks with preictable mobility. Technical Report GIT-CC-04-07, College of Computing, Georgia Institute of Technology, March 2004.

[11] A. Pentland, R. Fletcher, and A. Hasson. Daknet: Rethinking connectivity in developing nations. *IEEE Computer*, 37(1):78–83, Jan 2004.

[12] N. Sarafijanovic-Djukic and M. Grossglauser. Last Encounter Routing under Random Waypoint Mobility. In *Proc. IFIP-TC6 NETWORKING*, pages 974–988, 2004.

[13] A. Vahdat and D. Becker. Epidemic Routing for Partially-Connected Ad Hoc Networks. Technical Report CS-2000-06, Duke University, July 2000.

[14] Wizzy digital courier. http://www.wizzy.org.za.

[15] J. Yang and C.-K. Lee Y. Chen, M. Ammar. "ferry replacement protocols in sparse manet message ferrying systems". In *Proc. IEEE Wireless Communications and Networking (WCNC)*, March 2005.

[16] W. Zhao and M. Ammar. Message Ferrying: Proactive Routing In Highly-Partitioned Wireless Ad Hoc Networks. In *Proc. IEEE Workshop on Future Trends in Distributed Computing Systems*, May 2003.

[17] W. Zhao, M. Ammar, and E. Zegura. A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad hoc Networks. In *Proc. ACM Mobihoc*, May 2004.

[18] W. Zhao, M. Ammar, and E. Zegura. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In *IEEE INFOCOM*, 2005.

[19] W. Zhao, M. Ammar, and E. Zegura. Multicast routing in delay tolerant networks: Semantic models and routing algorithms. In *Proceedings of the SIGCOMM DTN Workshop*, August, 2005.