# DTN Routing as a Resource Allocation Problem

Aruna Balasubramanian Brian Neil Levine Arun Venkataramani
Department of Computer Science, University of Massachusetts, Amherst, USA 01003
{arunab, brian, arun}@cs.umass.edu

*Abstract*—**Existing DTN routing protocols use a variety of mechanisms, including discovering the meeting probabilities among peers, packet replication, or network coding. The primary focus of these mechanisms is to increase the likelihood of finding a path with limited information. So, these approaches have only an incidental effect on various routing metrics.**

**In this paper, we approach DTN routing as a resource allocation problem. Our protocol, RAPID, is designed to intentionally optimize a routing metric specified by the network administrator, which RAPID translates to a utility maximization objective. Unlike previous work, RAPID replicates a packet at a transfer opportunity only when the marginal utility of replication outweighs the corresponding resource cost. By tracking the number of packets replicas that already exist in the network, RAPID more efficiently uses constrained resources. We evaluate three metrics: minimizing average delay, minimizing worse case delay, and maximizing the number of packets that are delivered before a deadline. Our evaluations are based on traces of real mobility and data transfer between vehicular nodes in the DieselNet DTN, as well as synthetic exponential and powerlaw mobility models. For all metrics and scenarios, RAPID outperforms three other current protocols by significant margins.**

## I. Introduction

Delay tolerant networks (DTNs) allow routing when mobile nodes are sparsely populated and connect only intermittently. These include networks that attempt to survive large-scale natural disasters, sensor deployments for ecological monitoring [1, 2], ocean sensor networks [3, 4], and vehicular networks [5].

Existing DTN routing protocols use a variety of mechanisms, including discovering the meeting probabilities among peers, packet replication, network coding, adding stationary waypoint stores to the network, or using prior knowledge of mobility patterns to route data. The primary focus of these mechanisms is to increase the likelihood of finding a path with limited information. As a result, these approaches have an *incidental* effect on routing metrics such as worst-case delivery latency, average throughput, or percentage of packets delivered. This disconnect between application needs and routing protocols hinders deployment of DTN applications. Currently, it is difficult to drive the routing layer of a DTN by specifying priorities, deadlines, or cost constraints.

In this paper, we present an *intentional* DTN routing protocol, *Resource Allocation Protocol for Intentional DTN routing* (RAPID), that is explicitly designed to optimize a routing metric specified by the network administrator. RAPID works by translating the objective to a utility maximization objective. Unlike previous work, RAPID replicates a packet at a transfer opportunity only when the marginal utility of replication outweighs the corresponding resource cost. We evaluate three metrics in the context of RAPID: minimizing average delay, minimizing worse case delay, and maximizing the number of packets that are delivered before a deadline.

This work was supported in part by NSF award CNS-0519881.

Previous work [5], including our own, has shown that flooding acknowledgments of packet delivery is an effective method of improving delivery rates by removing useless packets from the system. RAPID extends this notion by reporting on the status of all packets in a node's buffer — by loosely keeping track of the number of replicas that already exist in the network, RAPID more efficiently uses transmission bandwidth and node storage. Our experiments show that the large increase in performance RAPID achieves easily justifies the small increase in bandwidth used to exchange this data.

To show the broad appeal of our RAPID approach, we evaluate our protocol quantitatively against a broad range of DTN scenarios that vary in the choice of mobility models and routing metrics. In particular, one of the mobility models is based on traces from our real DTN, called DieselNet, deployed among 40 buses in Amherst, MA. The buses transfer data when they pass one another and therefore provide traces of real node mobility and 802.11 transfers. We compare RAPID against *MaxProp* [5], *Spray and Wait* [6], and *PROPHET* [7] protocols, in addition to comparisons against *Random* and *Optimal* routing schemes.

Our evaluations show that RAPID performs better than previous protocols for different scenarios. For example, in trace-driven evaluations, delivered packets in RAPID have an average delay of 112 minutes, compared to 151 minutes using *MaxProp* and 230 minutes with *Spray and Wait*; RAPID delivers a greater percentage of packets as well. Similarly, RAPID delivers 15% more packets than *MaxProp* ahead of a scheduled deadline, and it has a 14% lower maximum delay compared to *MaxProp*. We also are able to compare RAPID to an ILP optimal solution for low packets loads and show empirically that RAPID is between 14–29% greater than optimal performance for average delay, compared to 28–51% for *MaxProp*.

## II. Related Work

*Epidemic* routing protocols replicate packets at transfer opportunities hoping to find a path to a destination. However, naive flooding wastes resources and can severely degrade performance. Proposed protocols attempt to limit replication or otherwise clear useless packets in various ways: using historic meeting information to avoid replication along paths with little chance of reaching the destination [10, 2, 12, 5, 8, 7]; removing useless packets using acknowledgments of delivered data [5]; using probabilistic mobility information to infer delivery [11]; replicating packets with a small probability [14]; using network coding [15]; and limiting the number of replicas of a packet [6, 11, 16, 3].

In contrast, *forwarding* routing protocols maintain at most one copy of a packet in the network [9, 13, 17, 18]. The consensus [9, 19, 3] appears to be that replicating packets can po-

| Problem | Storage | Bandwidth | Routing | Previous work (and mobility model) |
|---|---|---|---|---|
| P1 | Unlimited | Unlimited | Replication | MobiSpace [8] (AP traces) |
| P2 | Unlimited | Unlimited | Forwarding | Modified Djikstra's algorithm Jain et al. [9] (Any) |
| P3 | Finite | Unlimited | Replication | Davis et al. [10] (Simple partitioning synthetic), SWIM [11] (Exponential), Spray and Wait [6] (Exponential), MV [12] (Community-based synthetic), Prophet [7] (Community-based synthetic ) |
| P4 | Finite | Finite | Forwarding | Jones et al. [13] (AP traces), Jain et al. [9] (Synthetic DTN topology) |
| P5 | Finite | Finite | Replication | This paper (Vehicular DTN traces, exponential, and powerlaw meeting probabilities), MaxProp [5] (Vehicular DTN traces, community-based synthetic) |

Table I. A classification of some related work into DTN routing scenarios.

tentially improve performance over forwarding, but at a greater risk of degrading performance.

We see several major differences with the approach we propose in this paper and related work in DTN routing.

Most importantly, past work proposes different schemes that have only an *incidental* effect on desired performance metrics, including commonly evaluated metrics like average delay or delivery probability. Their theoretical intractability in general makes the effect of a particular mechanism or a protocol design decision on a given performance metric unclear. The distinguishing feature of RAPID is that the routing is *intentional* with respect to a given performance metric. Our approach systematically improves a given metric by explicitly calculating the effect of replication on available resources and the metric.

Jain et al. [9] propose an algorithm to optimize a specific routing metric using oracles with varying degrees of global system information and use only forwarding to route packets. Their work provide interesting insights but relying on global knowledge (details in Section III) makes a decentralized implementation impractical. Moreover,

RAPID also differs from most previous work in its assumptions regarding resource constraints, routing policy, and mobility patterns. Table I shows a taxonomy of several (but not all) existing DTN routing protocols. Bandwidth restrictions limit the size of transfer opportunities and storage restrictions the buffer size at nodes; both are either *finite* or *unlimited*.

In this paper, we focus on a replication-based algorithm with constraints on both storage and bandwidth (P5) as it is the most challenging and typical problem space for several reasons.

Problems P1 and P2 are interesting to examine for their theoretical tractability, but protocols that are to be deployed on testbeds or in real scenarios need to address resource constraints directly.

A series of works [6, 7, 10, 12, 11] have analyzed the case where storage at nodes is limited, but bandwidth is unlimited (P3). This scenario may happen when the radios used and the duration of contacts allow transmission of more data than can be stored by the node. However, we expect this scenario to be uncommon — typically storage is inexpensive and energy efficient. Trends suggest that high bitrate radios will remain more expensive and consumes more energy than storage [20]. Moreover, for mobile DTNs, and especially vehicular DTNs, transfer opportunities will remain short-lived.

Our related work [21] suggests that replication in DTNs is more effective than forwarding and also more robust to attack; hence we focus on replication over forwarding. Our past work *MaxProp* also makes this assumption. However, *MaxProp* is an incidental routing protocol unlike RAPID, and we show in Section VI that RAPID performs significantly better than *MaxProp*.

Finally, we note that some DTN routing schemes use ferries [22], stationary [23], or mobile agents [12, 24] that assist nodes in delivering packets but do not generate packets themselves. Our model (Section III) naturally incorporates stationary access points or ferries whose mobility schedule does not depend on the state of the network, but we leave for future work the discussion of ferries that move to relieve congestion hotspots.

## III. System Model

We model a DTN as a set of mobile nodes. Two nodes can exchange data when within communication range. A node can deliver data to a destination node directly or via intermediate nodes. We place constraints on both the amount of data that a node can store and the bandwidth available between two nodes. As we discuss in Section II, this assumption is in contrast to several previous works (e.g., [10, 11, 6, 12, 7]) that assume that nodes can exchange an unlimited amount of data during a meeting with storage being only resource constraint.

We focus on settings where node meetings last for a short interval with a fixed bandwidth during that interval. We represent meetings involving node $n_i$ as a sequence of transfer opportunities $\{(n_1, X_1, t_1), (n_2, X_2, t_2), \ldots\}$ where the $j^{th}$ element is a tuple consisting of the identity of the other node, the size of the transfer opportunity in bytes, and the time at which the meeting occurred, in that order.

We assume that destination nodes have sufficient capacity to store delivered data and that only storage for in-transit data is limited. We assume that nodes may replicate packets resulting in multiple in-transit copies of a packet in the system. We show in Section VI how RAPID protocol accomodates storage constraints for in-transit data.

In RAPID, we model the solution as a resource allocation problem, where a replicated packet represents an allocated resource. RAPID attempts to optimize a particular performance metric by *intentionally* allocating resources to packets within the constraints of the network. More specifically, a packet is allocated resource only when its benefit or expected performance improvement for the packet outweighs the cost or expected performance degradation to other packets. When the mobility pattern is unknown, nodes must learn the mobility from history of meetings and adjust the level of packet replication using a local view of system resources using limited high-delay feedback.

| | | |
|---|---|---|
| $B(i)$: Benefit | Difference in value of $i$ due to replication | $\Delta V(i)$ |
| $C(i)$: Cost | Difference in value of others due to replication | $\Delta\ (\Sigma_{j \in S} V(j))$ |
| $U(i)$: Utility | Marginal benefit of replication | $B(i) - C(i)$ |

Table II. Definitions to calculate utility

## IV. The RAPID Protocol

In this section, we present RAPID in detail. RAPID makes routing decisions to optimize a *utility function* associated with a specific *routing metric*. We define three routing metrics to exemplify the use of RAPID, namely: (*i*) minimizing the number of packets that miss delivery deadlines; (*ii*) minimizing the maximum delay across all packets; and (*iii*) minimizing the average delay of packets in the system.

For all metrics, the crucial question is — how can we balance the benefit of replication against the cost of overloading the system in an environment with limited high-delay feedback? Moreover, given limited bandwidth, which packets should we replicate during a transfer opportunity? RAPID replicates a packet only when it estimates an improvement in the performance metric for that packet and when this benefit outweighs the cost to other packets.

### A. Definitions

We define the *value* $V(i)$ of a packet $i$ as the contribution of the packet to the routing metric. Let $S$ denote the set of all packets in a remote node's buffer and let $i$ be the packet to be copied to the remote node. Table II describes the definition of the utility of replicating packet $i$.

For example, when the objective is to minimize average delay of packets, the value $V(i)$ of packet $i$ is defined to be the negative of the total time the packet is expected to reside in the system or its expected delay, as that is its contribution to the system objective. Intuitively, replicating a packet decreases the expected delay of the packet and increases its value. (Note that it is more natural to speak of larger values as more desirable than smaller values, hence the definition as negative expected delay.)

We define the cost $C(i)$ as the sum change in value to other packets *in the remote buffer* This definition of cost ensures that it is beneficial to replicate a packet only when the replication increases the value of that packet more than the aggregate decrease in the value of all other packets. i.e., RAPID attempts to improve the value of packets by replicating packets with low value first, but only when the utility of replication is positive.

RAPID has two core components: a *selection* algorithm and an *inference* algorithm. The selection algorithm is used to determine *which* packets to replicate at a transfer opportunity, given the value of a packet and the cost of replication. The inference algorithm is used to estimate the *value* and the cost given the routing metric.

### B. The Selection Algorithm: Canonical RAPID

The key steps of RAPID take place when two nodes are within radio range and have discovered one another. The protocol is symmetric; without loss of generality, we describe how node $Y$ determines which packets to transfer to node $X$.

| | |
|---|---|
| $D(i)$ | Expected delay = $T(i) + A(i)$ |
| $T(i)$ | Time since creation of $i$ |
| $A(i)$ | Expected delivery time by at least one of the replicas of $i$ = E$[a(i)]$ |
| $k$ | Estimated number of replicas of $i$ existing in network |
| $n_1, \ldots, n_k$ | Nodes known to have a copy of $i$ |
| $m_{n_j}(i)$ | Inter-meeting time between $n_j$ and the destination of $i$ |
| $f_{n_j}(i)$ | Random variable representing the time taken for delivering $i$ by node $n_j$ |
| $F_{n_j}(i)$ | Expected delivery time by $n_j$, E$[f_{n_j}(i)]$ |
| $a(i)$ | Random variable representing the time taken for delivering $i$ by at least one of the $k$ replicas |

Table III. Variables used by routing metric descriptions and by derivation of estimated delivery delay of packet $i$.

**Algorithm 1.**

1. *Initialization*: Nodes $X$ and $Y$ exchange meta-data including the list of packets in their buffer and the current values of the packets. (Detailed in Section V-E.)
2. *Per-packet Operation*: For each packet $i$ in node $Y$'s buffer, sorted in order of increasing values:
   (a) $Y$ estimates the benefit of replication $B(i)$ locally and the cost $C(i)$ to the packets in $X$'s buffer.
   (b) $Y$ computes the utility $U(i) = B(i) - C(i)$ and transmits $i$ to $X$ if $U(i) > 0$
3. *Termination*: Transfer ends when the other node is out of radio range or all eligible packets are transmitted.

RAPID also adapts to storage restrictions for in-transit data. If a node exhausts all available storage, packets with the highest value are deleted first. RAPID removes packets with high value to make room for new packets that have not been allocated enough resources.

### C. Inference algorithm

Next, we describe how this canonical version can support specific metrics using an inference algorithm. All metrics defined here use a variation of the expected delay as value. In Section V, we detail an algorithm to calculate both the expected delay $D(i)$ of a packet $i$ and the change in delay due to replication. Table III summarizes all variables used in these two sections.

#### C.1 Example 1: Minimizing average delay

To minimize the average delay of packets in the network we set the value of a packet as

$$V(i) = -D(i) \tag{1}$$

since the packet's expected delay is its contribution to the performance metric. Essentially, the protocol attempts to greedily replicate the packet with the largest delay, but only if the decrease in its delay due to replication outweighs the increase in delay to other packets.

### C.2 Example 2: Minimizing missed deadlines

RAPID can also be set to minimize the number of packets that miss delivery deadlines. Let $L(i)$ be the deadline of packet $i$ and $T(i)$ be the time since the packet was created. Then value is defined as

$$V(i) = \begin{cases} -(L(i) - D(i)), & L(i) < T(i) \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

A packet that has missed its deadline can no longer improve performance and is thus assigned a value of 0. The protocol replicates packets that are closest to missing their deadline first if cost does not outweigh benefit.

### C.3 Example 3: Minimizing maximum delay

To minimize the maximum delay of packets in the network, we define the value $V(i)$ as follows.

$$V(i) = \begin{cases} -D(i), & D(i) \geq D(j) \quad \forall j \in S_Y \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where $S_Y$ denote the the set of packets in the buffer of $Y$. Thus, $V(i)$ as defined above is the negative expected delay if $i$ is a packet with the maximum expected delay among all packets held by $Y$. By this definition, the cost is the change in maximum delay of packets in the buffer. The utility of a replication is negative if a packet is replicated to a buffer and the replication increases the delay of another packet in the buffer beyond the maximum delay.

Unlike the previous example, the value $V(i)$ is not exactly the contribution of packet $i$ to the performance metric. The reason is that the packet with the maximum expected delay in the system may not be in either $X$ or $Y$'s buffer. The above definition of $V(i)$ makes the exchange protocol work conserving, i.e., if there is a transfer opportunity, then a node will replicate a packet to improve local performance even if the improvement to the system performance metric is zero.

## V. Estimating utilities

In the routing metrics described in the previous section, a node estimates the value of packets and the utility of replication using an estimate of the delivery delay of the packet. Recall that the variables used in this section are summarized in Table III.

To estimate $D(i)$, a node calculates $T(i)$, the time since packet creation, and $A(i)$, the expected delivery time by one of the replicas of $i$. $T(i)$ is calculated simply as the difference between the current time and the creation timestamp carried in the packet. (Given the delays of real DTNs, we require minimal clock synchronization.) Estimating $A(i)$ is more challenging. The exact answer requires knowledge of the minimum expected time until any node with the replica of the packet delivers the packet. Thus, a node must know which other nodes carry replicas of the packet as well as nodes that may come to possess a replica of the packet in the future.

RAPID estimates $A(i)$ using control information passed between nodes (Step 1 in Section IV-B), which includes information about the estimated number of replicas that exist for a specific packet, and by making a simplifying assumption about node mobility as follows: RAPID omits higher-order terms in calculating expected delay by considering only direct delivery, ignoring routing via intermediaries, for extant replicas of a packet in the system. In Section VI, we show quantitatively that this assumption (and possibly incorrect information about the number of replicas) provides excellent performance even against traces of our real DTN.

In this section, we first present an algorithm to estimate $A(i)$ that uses all control information for a packet, including the number of replications $k$ and the nodes that carry the replica $n_1, n_2 ... n_k$. Secondly, we describe the sequence of meta-data exchanges between nodes to obtain the control information necessary to estimate $A(i)$.

### A. Estimating delivery time

In this section, we present a canonical algorithm for calculating the expected delivery time, $A(i)$. Subsequently, we show how the algorithm can be used under three different scenarios: exponentially distributed meeting times, powerlaw distributed meeting times, and traces of UMass DieselNet.

**Algorithm 2.** Node $n_j$ storing a set of packets $S$ to destination $n_x$ performs the following steps to estimate the time until packet $i \in S$ is delivered

1. $n_j$ sorts all packets $s \in S$ in the descending order of $m_{n_j}(s) + T(s)$.
2. Let $b_j(i)$ be the sum size of packets that precede $i$ in the sorted list of $n_j$. Figure 1 illustrates a sorted buffer containing packet $i$.
3. Let $B_j$ be the expected transfer opportunity in bytes between $n_j$ and $i$'s destination. (For the sake of readability, we drop the subscript $i$.) Nodes locally compute the expected transfer opportunity with every other node as a moving average of the past transfers.
4. If $n_j$ could deliver the packet only directly to the destination $n_x$, it would require $\lceil b_j(i)/B_j \rceil$ meetings with that node.
   Let $r$ be a distribution that models the inter-meeting times between nodes, and let $r_{jx}$ be the random variable that represents the time taken for $n_j$ and $n_x$ to meet. We transform $r_{jx}$ to random variable $r'_{jx}$ that represents the time until $n_j$ and $n_x$ meet $\lceil b_j(i)/B_j \rceil$ times. Then, by definition and the direct delivery assumption
   $$f_{n_j}(i) = r'_{jx} \quad (4)$$
5. The probability of delivering the packet within time $t$ given $k$ replicas is the minimum of the $k$ random variables $f_{n_y}$, $y \in [1, k]$
   $$\mathrm{P}(a(i) < t) = 1 - \prod_{y=1}^{k} (1 - \mathrm{P}(f_{n_y}(i) < t) \quad (5)$$
6. Accordingly: $A(i) = \mathrm{E}[a(i)]$ (6)

The algorithm above quantitatively defines the relationship between the number of replicas and the expected delivery time in the presence of bandwidth constraints. Prior DTN analyses [11] present a relationship between replication and delay but only for the case of a uniform mobility distribution and in the absence of bandwidth constraints.

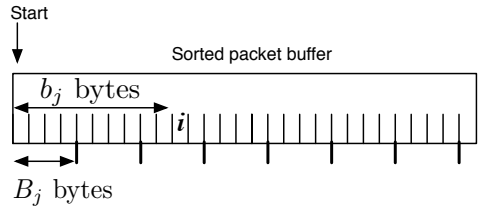


Fig. 1. A sample sorted list

### B. Known Exponential and Powerlaw Distributions

In this section, we specify the values of Algorithm 2 for a scenario where nodes meet with a known exponentially or powerlaw distributed meeting times. Few scenarios will have such exact characteristics, but examining these situations gives us insight for cases like UMass DieselNet, discussed subsequently.

Let the meeting time between nodes is described by a uniform exponential distribution with mean $1/\lambda$. From Algorithm 2, if there were no bandwidth restrictions, $B_j$ the transfer opportunity is infinity and $\lim_{B_j \to \infty} r'_{jx} = r_{jx}$,. We calculate $A(i)$ from Eqs. 4–6 as

$$\begin{aligned} \mathrm{P}(f_{n_j}(i) < t) &= 1 - e^{-\frac{1}{\lambda}t} \\ \mathrm{P}(a(i) < t) &= 1 - e^{-\frac{k}{\lambda}t} \\ A(i) &= \frac{\lambda}{k} \end{aligned} \tag{7}$$

In the absence of bandwidth restrictions, the expected delivery time is the mean meeting time factored by the number of copies of the packet. However, when transfer opportunities are limited, the expected delivery time depends on the contents of $n_j$'s buffer. From Step 4 of Algorithm 2, let $b_j(i)$ be the size of packets ahead of a packet $i$ in a node's buffer and let the size of the transfer opportunity available be $B_j$. Then the distribution for meeting the destination $\lceil b_j(i)/B_j \rceil$ times is described by a gamma distribution, or more specifically, the Erlang distribution. The mean of the Erlang distribution is $\frac{1}{\lambda} \cdot \lceil b_j(i)/B_j \rceil$. Then, the random variable $a(i)$ is described by the minimum of $k$ Erlang distributions.

Calculating an accurate estimate of the expected delivery time becomes complex when such bandwidth restrictions are introduced. Accordingly, we estimate the delay by making the following approximations and ignoring second order terms in our calculations. We assume that the time taken for a node to meet the destination $b_j(i)/B_j$ times is exponential with mean $\frac{1}{\lambda} \cdot \lceil b_j(i)/B_j \rceil$. Let

$$s_y(i) = \lceil b_y(i)/B_y \rceil, \ \ \forall y \in [1, k] \tag{8}$$

Then $A(i)$ is calculated from Eqns. 4 and 5 as

$$\begin{aligned} \mathrm{P}(a(i) < t) &= 1 - (e^{\frac{-s_1(i)}{\lambda} + \frac{-s_2(i)}{\lambda} + \ldots + \frac{-s_k(i)}{\lambda}}) \\ A(i) &= \frac{1}{\frac{s_1(i)}{\lambda} + \frac{s_2(i)}{\lambda} + \ldots + \frac{s_k(i)}{\lambda}} \end{aligned} \tag{9}$$

When the meeting time between nodes is described by a powerlaw distribution, the result is similar. The meetings times are drawn from an exponential distribution, but the mean meeting times between each pair of nodes is drawn from a powerlaw distribution. Assume that there are $k$ nodes with the copy of the packet and the mean meeting time between the $k$ nodes and the destination is $\mu_1, \mu_2 \ldots \mu_k$. Then $A(i)$ is calculated as

$$\begin{aligned} \mathrm{P}(a(i) < t) &= 1 - (e^{\frac{-s_1(i)}{\mu_1} + \frac{-s_2(i)}{\mu_2} + \ldots + \frac{-s_k(i)}{\mu_k}}) \\ A(i) &= \frac{1}{\frac{s_1(i)}{\mu_1} + \frac{s_2(i)}{\mu_2} + \ldots + \frac{s_k(i)}{\mu_k}} \end{aligned} \tag{10}$$

### C. When mobility distribution is unknown: Using Traces

For meeting times obtained from traces of our vehicular network, the inter-node meetings times are not described by a well-defined distribution (see [5]). To estimate inter-node meeting times, every node tabulates the average time to meet every other node using history. Nodes exchange this table as part of meta-data exchanges (Step 1 in Section IV-B) . A node combines the meta-data into a meeting-time matrix and the information is updated after each transfer opportunity. The $(x, y)$ entry of the meeting-time matrix $M_{x,y}$ is the expected time for $x$ to meet $y$ directly, calculated as the average of past meetings.

Node $n_j$ calculates $m_{n_j}(i)$, the expected time for $n_j$ to meet $i$'s destination, using the meeting-time matrix. $m_{n_j}(i)$ is the minimum time taken for $n_j$ to meet the destination of $i$ in at most $h$ hops. (Unlike exponential mobility models, some nodes in the trace never meet directly.) For example, if a node $A$ meets another node $C$ via an intermediary $B$, the expected meeting time is $M_{A,B} + M_{B,C}$ for $h = 2$. When two nodes never meet, the inter-meeting time is infinity. In our implementation we restrict $h = 3$. To calculate $A(i)$, we let $n_j$ approximate the trace meeting times as an exponential distribution. $A(i)$ is calculated using Eq. 9 as

$$m'_{n_j}(i) = \lceil b_j(i)/B_j \rceil \cdot m_{n_j}(i) \tag{11}$$

$$A(i) = (\sum_{y=1}^{k} \frac{1}{m'_{n_y}(i)})^{-1} \tag{12}$$

By making certain assumptions, the expected delivery time reduces to the harmonic mean of the expected meeting time between nodes $n_y, \ y \in [1, k]$ and the destination of the packet, taking into account bandwidth restrictions.

### D. Estimating change in value due to replication

Recall that RAPID assigns a utility $U(i) = B(i) - C(i)$. Replication increases the value of the packet and may decrease the value of the other packets in the buffer. A node quantitatively estimates the increase in the value of a replicated packet using the change in expected delivery time; For example, from Eq. 7, in the absence of bandwidth restrictions, the value of the packet (for the average delay metric) reduces from $\frac{\lambda}{k}$ to $\frac{\lambda}{k+1}$ due to replication.

RAPID estimates $C(i)$ the cost of replicating packet $i$ to other packets by examining the displacement of other packets in sorted list computed by Algorithm 2 (Step 1). For example, let $i$ be replicated to a buffer $S$ of node $n_j$ that, as a result, shifts several other packets towards the bottom of the list. For some subset $S' \subset S$ of packets in the list, replicating $i$ will increase

| Parameter | Exponential | Powerlaw | Trace |
|---|---|---|---|
| Number of nodes | 20 | 20 | 40 |
| Buffer size | 10 MB | 10 MB | 40 GB |
| Average transfer opp. size (variance 30%) | 2000 KB | 2000 KB | N/A |
| Duration | 10 min | 20 min | 19 hours |
| Size of a packet | 10 KB | 10 KB | 10 KB |
| Packet generation rate (exponential) | 20 sec | 1 min | 15 min |

Table IV. Experiment parameters

$\lceil b_j(x)/B_j \rceil$ for $x \in S'$ . From Eqns. 8, 9, 10, and 12, we see that an increase in $b_j(x)$ may increase the expected delivery time.

### E. Control channels to exchange meta-data

RAPID estimates the value of a packet by exchanging meta-data, which we distinguish as a *control channel*. In our previous work [5], we show that flooding of delivery acknowledgments during transfer opportunities is an effective method of removing already-delivered packets from the system. This approach can be generalized to propagate both acknowledgments and other control information. The control information needed to estimate the expected delay as described in Section V are the number of replicas of a packet, nodes that have a copy of the packet, contents of the nodes' buffers. We discuss two methods of propagating the control information: a global channel and causal channel. The choice of channel depends on the infrastructure and resource constraints of the DTN.

- A *global channel* is an idealized system that broadcasts acknowledgments for delivered packets as well as all meta-data instantaneously. More realistically, this information could be broadcast with negligible delay using a low bit-rate, long-range channel such as WiMax or cell phone networks.
- A *causal channel* exchanges during transfer opportunities acknowledgments for delivered packets as well as all control information learnt from past exchanges.

In the absence of a global channel, it is difficult to have accurate values for all the control information needed to estimate the expected delay. Nodes exchange the following information during a transfer opportunity: the aggregate number of replicas of a packet, nodes that have a copy of the packet and (hashes of) the contents of nodes' buffers as gathered from all the nodes met in the past.

Although causal information may not be current, RAPID uses it to estimate the expected delay. We have found in practice that the causal channel is sufficient for good performance as shown in the next section. Moreover, in our evaluations, we did not discount the costs of this data during transfer opportunities. Despite the increased meta-data exchanged, RAPID significantly outperforms other protocols.

## VI. Evaluation

The goal of our evaluations is to show that, unlike existing work, RAPID can improve performance for customizable metrics. We evaluate using RAPID to minimize maximum delay, minimize average delay, and minimize missed deadlines. We also compare the performance of RAPID to an optimal solution.

### A. Assumptions

Our evaluations are based on a custom event-driven simulator. The simulator takes as input a schedule of node meetings, the bandwidth available at each meeting, and a routing algorithm. We compare RAPID (using only the causal channel for exchanging control information) to five other routing protocols: *MaxProp* [5], *PROPHET* [7], *Spray and Wait* [6], *Random*, and *Optimal*. In all evaluations, we account for the cost of exchanging meta-data in RAPID.

*MaxProp* operates in a storage- and bandwidth-constrained environments, allows for packet replication, and leverages delivery notifications to purge old replicas; of recent related work, it is closest to RAPID's objectives. *Random* replicates randomly chosen packets for the duration of the transfer opportunity. *Spray and Wait* restricts the number of replications of a packets to $L$, where $L$ is calculated based on the number of nodes in the network. For our simulations, we implement the binary *Spray and Wait* and set L to 10. *PROPHET* uses historical information to calculate meeting probabilities. We implemented *PROPHET* with parameters $P_{init} = 0.75$, $\beta = 0.25$ and $\gamma = 0.98$.

We also compare RAPID to *Optimal*, the optimal routing protocol that provides an upper bound on performance. *Optimal* is computed using an ILP formulation stated in our Appendix and solved using the CPLEX solver [25].

We took *traces* from UMass DieselNet, a real network of 40 buses equipped to take measurements of mobility and transfers. Each bus carries a small computer, two 802.11 wireless radios, 40GB of storage, and a GPS device. The buses are designed to search for one another 100 times a second; once found, they exchange random data with each other using TCP until they are out of wireless range. They send logs of these transfer back to a central server. We used 60 days of logs in our experiments that contain the schedule of bus meetings and the transfer opportunity size at each meeting. Note these traces were recorded in Spring 2006 (and are not the Spring 2005 traces of a slightly smaller network used in our previous work [5]).

To model stochastic mobility patterns, we use uniform exponential and the powerlaw distributions. Song et al. [26] suggests that meeting times between people in a busy public place follow a *uniform exponential model*, i.e., pairs of nodes meet each other once every $t$ seconds where $t$ is exponentially distributed with a mean that is common for all pairs of nodes. We evaluated the protocols over a power-law distribution as well since it is common in previous work (e.g., [8]).

The default parameters used for the experiments are tabulated in Table IV. Destinations are randomly chosen uniformly from all other nodes. Packets are generated with an exponential inter-arrival time.

### B. Results from DieselNet traces

In all experiments, *MaxProp*, RAPID and *Spray and Wait* performed significantly better than *PROPHET* (which is not shown in the graphs for clarity). We assumed that nodes have until the end of each day only to deliver packets, and we plot the average performance over 60 days of trace data.

Figure 2 shows the average delay of delivered packets using

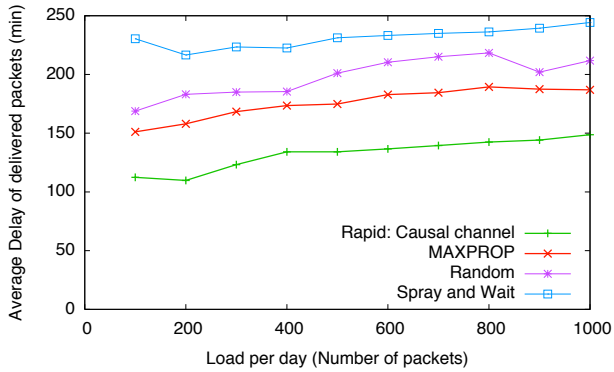


Fig. 2. **Trace data:** Load vs Avg delivery delay

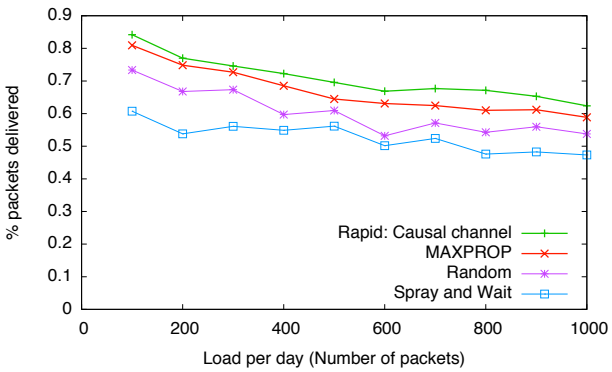


Fig. 3. **Trace data:** Load vs Delivery rate

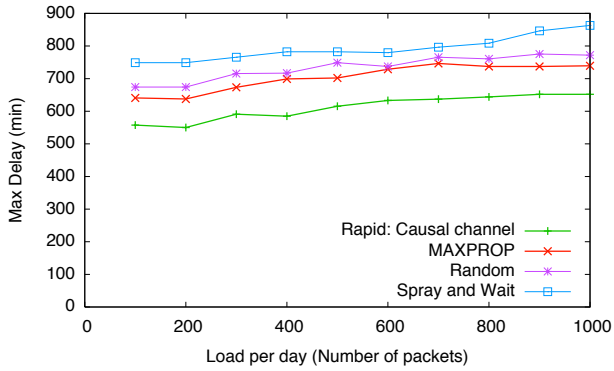


Fig. 4. **Trace data:** Load vs Max delay

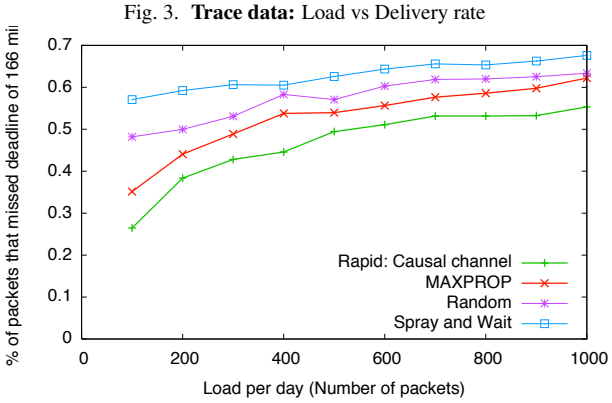


Fig. 5. **Trace data:** Load vs % missed delivery deadline

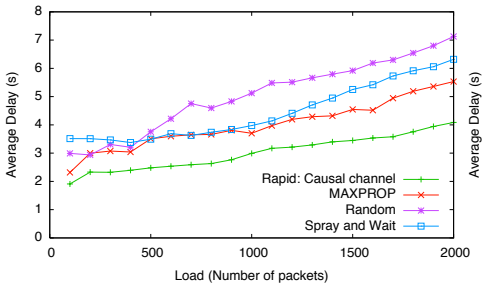


Fig. 6. Exponential: Load vs Avg delay

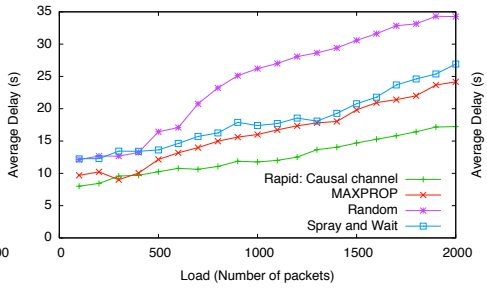


Fig. 7. Powerlaw: Load vs Avg delay

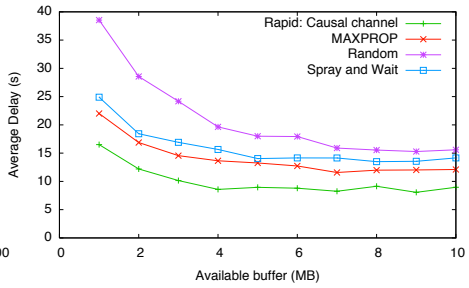


Fig. 8. Powerlaw: Buffer space vs Avg delay (similar results for exponential)

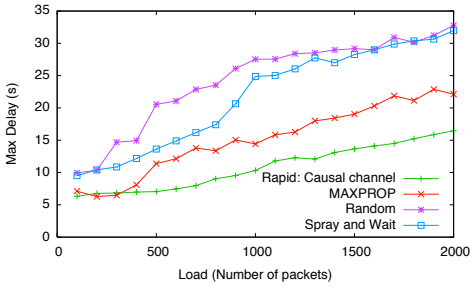


Fig. 9. Exponential: Load vs Max delay (similar results for powerlaw)

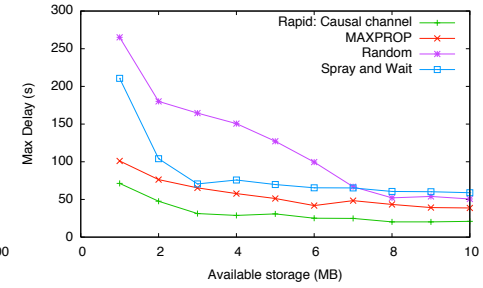


Fig. 10. Powerlaw: Buffer space vs Max delay (similar results for exponential)

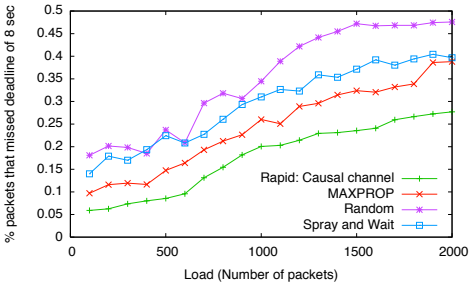


Fig. 11. Powerlaw: Load vs % missed deadline (similar results for exponential)

the four protocols under varied load when RAPID's routing metric is minimizing average delay. On average, when using RAPID packets are delivered 40 minutes (20%) earlier than *MaxProp*, 60 minutes (30%) earlier than *Random*, and 90 minutes (40%) earlier than *Spray and Wait*. Moreover, the fraction of packets delivered by RAPID is consistently greater than these protocols (shown in Figure 3) even though RAPID does not intentionally improve delivery rate.

Figure 4 shows that RAPID reduces the maximum delay of packets by an average of 80 minutes compared to *MaxProp* and

*Random* and about 190 minutes compared to *Spray and Wait*. Figure 5 shows the results when RAPID is set to minimize the percentage packets that missed a deadline. RAPID reduces the number of nodes that miss deadline by up to 15% compared to *MaxProp*, 18% compared to *Random* and 28% compared to *Spray and Wait*.

Because each bus takes a different route, standard deviation and similar measures of variance are not appropriate for comparing the means. Accordingly, we performed a paired $t$-test [27] to compare the average delay of every source/destination pair using RAPID to the average delay of the same source-destination pair using *MaxProp*. In our tests, we found $P$-values always less than 0.0005, indicating the differences between the means reported in these figures are statistically significant.

### C. Results from synthetic mobility models

In this section, we show results comparing RAPID to *MaxProp*, *Random*, and *Spray and Wait* using synthetic mobility models. We set the mean of exponentially distributed meeting times to 30 seconds, and we set the exponent of power law meeting times to 0.3 seconds. Each point on the graphs represents the average of 10 trial runs, each with a different random number generator seeds. The delivery rate for all protocols is nearly 100% since all nodes meet all other nodes relatively frequently.

Figure 9 shows the maximum delay of packets using the four routing protocols when the meeting times are exponential and the load is varied. RAPID reduces average delay by up to 30% compared to MaxProp, which is the next-best performing protocol. The reason *MaxProp* performs worse is that it prioritizes new packets; older, undelivered packets won't see service as load increases, which increases maximum delay in the system. This experiment shows that an intentional routing protocol can achieve a better allocation of resources even in a completely decentralized setting. In Figure 6, we see similar results when RAPID's metric is set to average delay.

We see the same benefits of RAPID over the other protocols for powerlaw meeting distributions in Fig. 7. *Because of space limitations, for the remainder of our experiments, we show results for only one of exponential or powerlaw distributions* — just as in Figs. 6 and 7, the results are similar in that the relative performance of the protocols is the same.

Figure 8 shows how constrained buffers varied from 1MB to 10MB affect the max delay metric (for powerlaw meeting times and fixed load of 1000 packets). RAPID is able to best manage limited buffers, though all protocols need very little buffer before delay is relatively constant. Figure 10 shows how constrained buffers affect the maximum delay of packets. We see RAPID reduces delay by an average of 30% compared to *MaxProp* and by greater than 50% compared to *Spray and Wait* and *Random* when the buffer is less than 5MB. We note that *Spray and Wait* and *Random* perform very poorly when the buffer space is less than 2MB while RAPID and *MaxProp* are stable even when the available buffer space is low, suggesting that propagating acknowledgements and removing delivered packets from the buffer is effective especially when resources are limited.

Fig. 11 show the performance of the different routing protocols with respect to minimizing the number of packets that missed deadline. RAPID performs significantly better than other protocols because, fundamentally, the latter protocols focus only on finding a path rather than efficiently allocating resources.

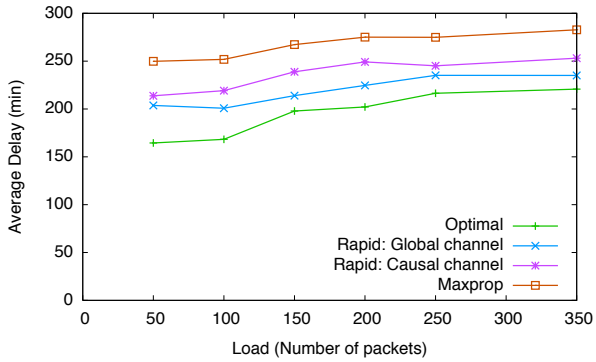


Fig. 12. **Trace:** A comparison of RAPID against *Optimal* (load versus average delay)

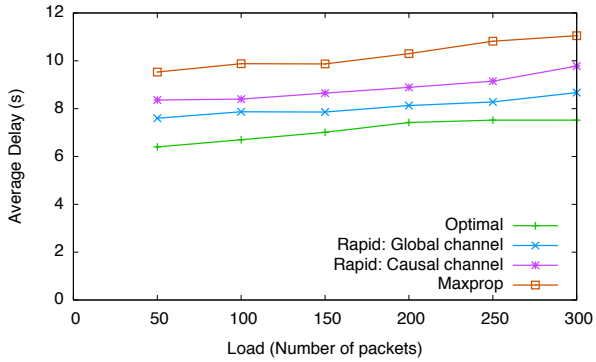


Fig. 13. **Powerlaw:** A comparison of RAPID against *Optimal* (load versus average delay)

#### C.1 Comparison with *Optimal*

In this section, we compare RAPID to *Optimal* which is an upper bound on the performance. Additionally, we show the performance of RAPID for both global and causal channels. These results show us how inaccuracies in meta-data reduce the performance of RAPID.

To obtain the optimal delay, we formulate the DTN routing problem as an Integer Linear Program (ILP) optimization problem when the meeting times between nodes are precisely know. The optimal solution does not use replication when there are no failures in the system, propagation delay of all links are equal, and when node meetings are known in advance. We present a formulation of this problem in the Appendix. Our evaluations use CPLEX solver [25] to show specific results for the *Optimal* algorithm. Because the solver grows in complexity with the number of packets, these simulations are limited to only 350 packets and are presented separately. Jain et al. [9] solve a similar DTN routing problem but allow packets to be fragmented across links and assigned non-zero propagation delays on the links. This severely limited the size of the network and the number of packets they could evaluate.

Figs. 13 and 12 present the comparisons in average delay performance between *Optimal*, RAPID, and *MaxProp* when the mobility model is powerlaw and trace-driven respectively. In the

trace-driven experiments, when computing average delay, the delay of an undelivered packet is taken as the delay since packet creation to the end of the bus day. (Due to space limitations we omit results for exponential distributions.)

The experiments show that RAPID with a global channel performs fairly close to optimal. This observation suggests that much of the performance improvements to RAPID with a causal channel can come from allowing control information to propagate more easily. Since control information is small compared to data, RAPID can make efficient use of a low-bandwidth, long-range radio to propagate control information, such as WiMax or XTend.

## VII. Conclusions

Previous work in DTN routing protocols has seen only incidental performance effects from various routing mechanisms and protocol design choices. In contrast, we have proposed a routing protocol for DTNs that intentionally maximizes the performance of a specific routing objective. Our protocol, RAPID, treats DTN routing as a resource allocation problem, and makes use of causal information that is passed between nodes as a control channel. We have shown that RAPID yields significant performance gains over incidental routing protocols.

The intentional routing approach opens up several research questions. The resource allocation formulation was inspired by the utility-theoretic framework [28] for Internet-like low-feedback-delay networks pioneered by Kelly. However, we have not shown the ability of RAPID's local utility maximization approach to achieve the global optima for different routing performance objectives; unlike [28], our benefit (cost) function is not always strictly concave (convex) and smooth. In future work, we will also investigate encoding other application-specific metrics, including consistency requirements and monetary costs of routing in heterogeneously priced networks as resource constraints in RAPID. To this end, we are encouraged by the feasibility of an intentional routing approach.

## References


[1] J. Sorber, A. Kostadinov, M. Brennan, M. Corner, and E. Berger, "eFlux: Simple Automatic Adaptation for Environmentally Powered Devices. (Poster/Demo)," in *Proc. IEEE workshop on Mobile Computing Systems and Applications (HotMobile/WMCSA)*, April 2006.

[2] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," in *ASPLOS*, Oct. 2002.

[3] M. Motani, V. Srinivasan, and P. Nuggehalli, "Peoplenet: engineering a wireless virtual social network.," in *ACM MOBICOM*, pp. 243–257, 2005.

[4] J. Partan, J. Kurose, and B. N. Levine, "A survey of practical issues in underwater networks," in *ACM International Workshop on UnderWater Networks (WUWNet)*, Sept. 2006.

[5] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks," in *Proc. IEEE INFOCOM*, April 2006.

[6] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *ACM WDTN*, pp. 252–259, 2005.

[7] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," in *Proc. Intl Wrkshp on Service Assurance with Partial and Intermittent Resources (SAPIR)*, 2004.

[8] J. Leguay, T. Friedman, and V. Conan, "DTN Routing in a Mobility Pattern Space," in *ACM WDTN*, pp. 276–283, 2005.

[9] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *ACM SIGCOMM*, pp. 145–158, 2004.

[10] J. Davis, A. Fagg, and B. N. Levine, "Wearable Computers and Packet Transport Mechanisms in Highly Partitioned Ad hoc Networks," in *Proc. IEEE Intl. Symp on Wearable Computers (ISWC)*, pp. 141–148, October 2001.

[11] T. Small and Z. Haas, "Resource and performance tradeoffs in delay-tolerant wireless networks," in *Proc. ACM WDTN*, pp. 260–267, 2005.

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

[13] E. P. C. Jones, L. Li, and P. A. S. Ward, "Practical routing in delay-tolerant networks," in *ACM WDTN*, pp. 237–243, 2005.

[14] Y.-C. Tseng and S.-Y. Ni and Y.-S. Chen and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wireless Networks*, vol. 8, no. 2/3, pp. 153–167, 2002.

[15] J. Widmer and J.-Y. L. Boudec, "Network coding for efficient communication in extreme networks," in *ACM WDTN*, pp. 284–291, 2005.

[16] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure-coding based routing for opportunistic networks," in *ACM WDTN*, pp. 229–236, 2005.

[17] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data Mules: Modeling a Three-tier Architecture for Sparse Sensor Networks," in *IEEE Intl. Wkshp on Sensor Network Protocols and Applications (SNPA)*, May 2003.

[18] T. Spyropoulos and K. Psounis and C. Raghavendra, "Single-copy Routing in Intermittently Connected Mobile Networks," in *IEEE SECON, (extended version)*, October 2004.

[19] T. Spyropoulos and K. Psounis and C. Raghavendra, "Efficient Routing in Intermittently Connected Mobile Networks: The Multi-Copy Case," in *submission to IEEE Transactions on Networking*, 2005.

[20] P. Desnoyers, D. Ganesan, H. Li, M. Li, and P. Shenoy, "Presto: A predictive storage architecture for sensor networks," in *HotOS X*, 2005.

[21] J. Burgess, M. Corner, and B. N. Levine, "Surviving Attacks on Disruption-Tolerant Networks without Authentication," tech. rep., UMass Amherst, 2006.

[22] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *ACM MobiHoc*, pp. 187–198, 2004.

[23] W. Zhao, Y. Chen, M. H. Ammar, M. Corner, B. N. Levine, and E. Zegura, "Capacity Enhancement using Throwboxes in DTNs," in *Proc. IEEE Intl Conf on Mobile Ad hoc and Sensor Systems (MASS)*, Oct 2006.

[24] B. Burns, O. Brock, and B. N. Levine, "Autonomous Enhancement of Disruption Tolerant Networks," in *Proc. IEEE International Conference on Robotics and Automation*, May 2006.

[25] CPLEX, "Linear programming solver, http://www.ilog.com/."

[26] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating location predictors with extensive Wi-Fi mobility data," in *Proc. IEEE INFOCOM*, pp. 1414–1424, March 2004.

[27] W. Navidi, *Statistis for Engineers and Scientists*. McGraw Hill, 2006.

[28] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," in *Journal of the Operational Research Society*, vol. 49, 1998.


## Appendix

**LP Formulation.** We divide time into discrete intervals so every node meets at most one other node in an interval. This idea is similar to the formulation presented in [9]. The inputs to the problem are

- The set of time intervals $I, = 1, 2...h$ . The function $b()$ returns the beginning of the interval. $e()$ returns the end of an interval and variable $h$ represents the last interval
- The set of nodes in the network $N$
- The set of edges $E$. An edge is defined when two nodes meeting in an interval. We define functions $f()$ and $s()$ to return the first and the second node that meet respectively, $d()$ returns the interval in which the edge is defined.
- The set of packets $P$. Function $src()$ return the source of the packet, $dest()$ return the destination of the packet, $c()$ returns the interval in which the packet was created, $t()$ returns time the packet was created and $size()$ returns the size of the packet.
- The bandwidth for each meeting is a constant and is B.

When two nodes $i$ and $j$ meet, they are represented by two edges, $e$ where the function $f(e)$ is $i$ and $s(e)$ is $j$ and $e'$ where $f(e')$ is $j$ and $s(e')$ is $i$.

The variables are

- $X(p \in P, e \in E) = 1$ if $j$ is forwarded over the edge $e$ and is 0 otherwise
- $N(p \in P, n \in N, i \in I)$ = 1 if node $n$ has packet $p$ in the interval $i$ and is 0 otherwise
- $D(p \in P, i \in I)$ = 1 if packet $p$ is before interval $i$ and is 0 otherwise

$X$ can be used to construct the optimal path taken by a packet. The value to $X$

is constrained by the $N$ and $D$ variables. The objective function is

$$min \sum_{p \in P} \sum_{i \in I} \sum_{e \in E \ : d(e)=i, s(e)=\text{dest(p)}} (s(i) - t(i)) \cdot X(p,e)$$
$$+ \sum_{p \in P} (1 - D(p, e(h)) \cdot (s(h) - c(p))$$

All constraints use notations $\forall p, n, i, e$ to mean $\forall p \in P, \forall n \in N, \forall i \in I$ and $\forall e \in E$. The constraints are

**Initialization constraints**

$N(p,n,i) = 0 \, if \; i < c(p) \; \forall p, n, i$

$N(p,n,i) = 1 \, if \; src(p) = n$ and $c(p) = i \; \forall p$

**Bandwidth constraint**

$$\sum_{p \in P} (X(p,e) * size(p) \leq B \; \forall e$$

**Transfer constraints**

$$N(p,n,i-1) - \sum_{e \in E : f(e)=n, d(e)=i} X(p,e) +$$
$$\sum_{e \in E : s(e)=n, d(e)=i} X(p,e) - N(p,n,i) = 0 \; \forall p, n, i$$

$N(p, f(e), d(e) - 1) - X(p,e) >= 0 \; \forall p, e$

**Conservation constraint**

$1 - \sum_{n \in N} N(p,n,i) = 0$ if $i >$ c(p) $\forall p, e$