2006

Capacity Enhancement using Throwboxes in DTNs

Wenrui Zhao
Georgia Institute of Technology

Yang Chen
Georgia Institute of Technology

Mostafa Ammar
Georgia Institute of Technology

Mark Corner
University of Massachusetts - Amherst

Brain Levine
University of Massachusetts - Amherst

See next page for additional authors

Capacity Enhancement using Throwboxes in DTNs

Wenrui Zhao† Yang Chen† Mostafa Ammar† Mark Corner* Brian Levine* Ellen Zegura†

†College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332,
*Department of Computer Science, University of Massachusetts, Amherst, MA 01003
{wrzhao, yangchen, ammar, ewz}@cc.gatech.edu {mcorner,brian}@cs.umass.edu

Abstract— **Disruption Tolerant Networks (DTNs) are designed to overcome limitations in connectivity due to conditions such as mobility, poor infrastructure, and short range radios. DTNs rely on the inherent mobility in the network to deliver packets around frequent and extended network partitions using a *store-carry-and-forward* paradigm. However, missed contact opportunities decrease throughput and increase delay in the network. We propose the use of *throwboxes* in mobile DTNs to create a greater number of contact opportunities, consequently improving the performance of the network. Throwboxes are wireless nodes that act as relays, creating additional contact opportunities in the DTN. We propose algorithms to deploy stationary throwboxes in the network that simultaneously consider routing as well as placement. We also present placement algorithms that use more limited knowledge about the network structure. We perform an extensive evaluation of our algorithms by varying both the underlying routing and mobility models. Our results suggest several findings to guide the design and operation of throwbox-augmented DTNs.**

I. Introduction

Emergency workers, scientists, and developing nations continually push networks beyond the edges of reliable communication infrastructure and into highly challenged environments. Often it is the case that workers responding to a disaster lack the stability of the electrical power grid, scientists monitor wildlife and phenomena in remote areas with a low density of nodes, and developing nations deploy networks in regions that lack infrastructure. Due to the limitations of power, connectivity, density of nodes, cost, and maintenance, devices may not be able to form a fully connected network for routing data, such as a MANET or a mesh network [2].

An emerging class of networks, commonly referred to as Disruption Tolerant Networks (DTNs) [10], [11], are explicitly designed to overcome such limitations. DTNs rely on the inherent mobility in the network to deliver packets around frequent and extended network partitions using a *store-carry-and-forward* paradigm [6], [7], [13], [22], [26]. While such networks can only support delay-insensitive applications, such as messaging, file transfer, and data dissemination, they enable communication where otherwise there may be none.

This work was supported in part by the NSF Grant CNS-0519784/CNS-0519881 and DARPA Grant W15P7T-05-C-P213.

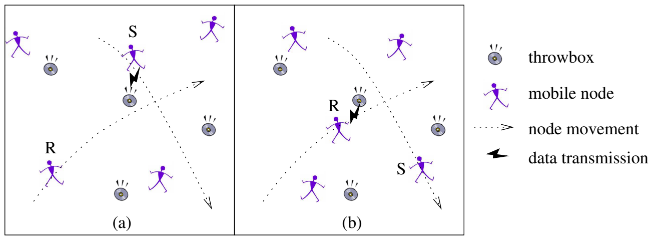


Fig. 1. An example of DTNs using throwboxes.

Even though DTNs are highly robust to poor connectivity, their performance is highly dependent on chance encounters between nodes. For the network to route a packet between two nodes that never meet, a transitive series of meetings, each of sufficient duration, must occur. When opportunities are missed, there is a decrease in throughput and an increase in delay in the network.

Consequently, if a network designer can engineer a greater number of contact opportunities then the performance of the network can be greatly enhanced. In this paper we propose the use of *throwboxes* to accomplish this goal. Throwboxes are small and inexpensive devices equipped with wireless interfaces and storage. Throwboxes are stationary, thus when two nodes pass by the same location at different times, the throwbox acts as a relay, creating a contact opportunity where none existed before. Fig. 1 shows an example of using throwboxes in a mobile DTN. In Fig. 1(a), node *S* sends data to a throwbox. At a later time when node *R* moves close to the throwbox, it receives *S*'s data from the throwbox, shown in Fig. 1(b).

To demonstrate how throwboxes improve data delivery in DTNs, we perform a simple simulation based on traces from our vehicular DTN testbed named DieselNet [6]. In Table I we show that the number of communication opportunities and the average capacity, i.e., the maximum data rate that can be sent in the long term, between two buses before and after the deployment of a *single* throwbox. The results show a dramatic improvement: the average capacity between the vehicles increases by a

Number of throwboxes	None	One
Total Contact Duration (sec)	631	11,927
Average Capacity (Kbps)	3.5	66.3
Delay (sec)	63,012	3,120

TABLE I
PERFORMANCE BETWEEN TWO BUSES WITH AND WITHOUT ONE THROWBOX. THE RADIO BANDWIDTH IS 2 MBPS.

factor of 19 and the delivery delay decreases by a factor of 20.

In this simulation, we place the throwbox by using our intuition about mobility in the network. While these results are encouraging, a larger network with more nodes and a larger number of throwboxes necessitates the use of algorithms to automatically place the throwboxes. This paper investigates such algorithms for adding throwboxes to a running DTN. Placing throwboxes into an operational network also creates an opportunity to modify the routing to utilize the throwboxes effectively. However, the addition of the throwboxes affects the flow of data through the network, which consequently affects the placement of throwboxes. While placement can be considered in isolation, we show that it is most effective to consider the routing algorithm *simultaneously* with the placement of throwboxes, thus maximizing the overall effectiveness of the throwboxes after they are deployed.

We perform an extensive evaluation of our placement algorithms by varying both the underlying routing and mobility models. Our results suggest several findings to guide the design and operation of throwbox-augmented DTNs:

- Throwboxes are very effective in improving throughput and can also reduce data delivery delay. The improvement in throughput is generally more significant than improvement in delay.
- Throwboxes are most useful for routing algorithms that use multi-path routing and when nodes follow structured mobility patterns.
- Throwbox deployment that incorporates knowledge about contact opportunities performs better than deployment that ignores this knowledge. Additionally, if deployment is customized to existing traffic patterns, the algorithms are more effective than assuming that traffic is equally distributed.

The rest of this paper is structured as follows. In Section II, we describe throwbox characteristics and our network model. In Section II-D, we present a framework to systematically study the issues of throwbox deployment and routing. We study various routing approaches in Section III and IV, and we present the simulation results in Section V. We review related work in Section VI and conclude the paper in Section VII.

N	The set of nodes
n	The number of nodes
m	The number of throwboxes
L	The set of potential throwbox locations
P	The set of nodes and potential throwbox locations
λ	The total data rate
b_{ij}	The relative traffic demand from i to j, where $i, j \in N$
c_{ij}	The average capacity between i and j, where $i, j \in P$
x_i	The number of throwboxes deployed at location i, $i \in L$
f_{ij}^{sd}	The traffic load from S-D pair (s, d) that is forwarded from i to j, where $s, d \in N$ and $i, j \in P$

TABLE II
TABLE OF NOTATIONS.

II. NETWORK MODEL USING THROWBOXES

In this section, we detail our assumptions about throwbox characteristics, present our network model, define our performance objectives, and define our evaluation methodology.

A. Throwboxes

Throwboxes must be designed to have high availability, be able to transmit data at high data rates, have sufficient processing power to handle such data rates, and be energy efficient.

Several platforms exist that meet some or all of these requirements. Commercially, the Intel Stargate [23] is a reasonably efficient and powerful device. However, in our prior work, Banerjee et al. [4] created a prototype device that is closer to our assumptions. The Triage platform and software consist of a coupled Stargate board and MicaZ mote. The Triage software reduces the amount of time that the device must spend operating the high-power Stargate board, running tasks on the MicaZ whenever possible. Using a hailing radio, passing nodes can trigger the always-on MicaZ to power on the Stargate board and 802.11 CF network interface. In this way, Triage never misses a passing node but remains energy efficient. Solar panels recharge batteries to ensure long-term use. An energy efficient architecture for throwboxes is detailed in recent work [3]. Given the current technological advances in processor and storage, we expect the cost and size of such devices to continually decrease, and energy efficiency to improve.

B. Network Model

We assume a network composed of m throwboxes and n network nodes. (Table II lists all notations used in this paper.) All devices–nodes and throwboxes—communicate with each other through wireless interfaces (e.g., 802.11) and are equipped with storage that carries network data. Devices use periodic beacon messages to discover one another and to exchange buffered data. In DTNs, nodes transfer application data in units called *messages* (i.e., bundles [11]), which can be of varied sizes. Each message carries a timeout value that specifies when the message should be dropped if not delivered.

We assume that throwboxes are neither sources nor destinations and that throwboxes never interact with each other. In addition, we assume that throwboxes have sufficient energy to carry out all tasks. Details on a specific energy management algorithm can be found in our related work [4].

We assume that all other nodes are both sources and destinations of data communication. We represent the traffic demand by a matrix $\mathbf{b}_{\mathbf{n}\times\mathbf{n}}$ where b_{ij} specifies the relative long term traffic demand from node i to node j such that $\sum_{i,j} b_{ij} = 1$, where $1 \leq i,j \leq n$. This model is general in that it can capture different traffic patterns, e.g., uniform traffic among nodes that is typical in ad hoc networks or concentrated traffic to a single destination that is typical in sensor networks.

Transfer opportunities in a DTN occur as a time-varying process. We characterize transmission opportunities using *average capacity*, which is the maximum data rate that can be sent between two nodes in the long term. Let u_{ij} be the average contact duration and v_{ij} be the average inter-contact time between nodes i and j. We compute the average capacity as $c_{ij} = \frac{u_{ij}w}{u_{ij}+v_{ij}}$ where w is the transmission data rate when node i and j are in contact. This is because nodes i and j are only able to communicate with each other for a fraction $\frac{u_{ij}}{u_{ij}+v_{ij}}$ of time. The average capacity between a node and a throwbox is defined using the same notation. We denote C as the set of all c_{ij} values. Note that the average capacity is shared by traffic in both directions. In addition, wireless interference is ignored in this paper because of the sparse distribution of nodes.

C. Performance Objective

Throwboxes have the potential to improve both throughput and delay. In this paper, we focus on improving throughput using throwboxes. For a given number of throwboxes, we would like to maximize the total traffic demand that can be supported. Recall that b_{ij} specifies the relative traffic demand from node i to node j, and $\sum_{i,j} b_{ij} = 1$. The objective of throwbox usage is to maximize λ such that node i is able to send data to node j at rate λb_{ij}, $1 \leq i,j \leq n$. In other words, we try to maximize the total data rate, λ.

We choose to focus our efforts on optimizing throughput for several reasons. Given frequent partitions in DTNs, it is more important to deliver messages successfully than to minimize the delay. In addition, applications in DTNs are expected to already tolerate relatively large delays. Furthermore, improving throughput is often in accordance with reducing delay, as shown in Section I and our simulation results.

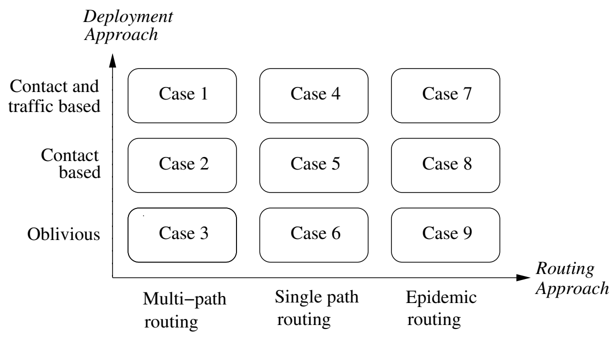


Fig. 2. Throwbox deployment and routing framework.

D. Throwbox Deployment and Routing Framework

A defining characteristic of DTNs is that it is difficult to gather data about the performance of the network itself. Information about contact opportunities and traffic load may not be available as input to deployment algorithms. Similarly, algorithms for routing can vary in sophistication from simple epidemic replication to more efficient strategies that do not replicate messages and balance load across multiple paths.

Specifically, we consider three different deployment scenarios that differ by what information is available, and for each we evaluate three different routing scenarios that differ in their sophistication. Fig. 2 illustrates these nine cases. For deployment, the three cases are:

- *Contact and traffic-based (CTB):* both contact opportunity C and traffic demand $\mathbf{b}_{\mathbf{n}\times\mathbf{n}}$ information are input to the deployment algorithm;
- *Contact-based:* information about contact opportunities C is input to the deployment algorithm;
- *Oblivious:* no information about the contact opportunities or traffic matrix is available for the deployment algorithm.

In the simplest case, routing in DTNs is based on *epidemic* style replication of data that is oblivious to information about traffic load and contact opportunities. With more sophistication, an algorithm can take these data as input and compute the optimal *single path* from source to destination without using replication. Improved performance is possible when different messages are routed each along different single paths (without replication) to make use of more resources in the network, which we call *multi-path* routing. These routing algorithms are optimal as they have complete knowledge of the contact opportunities and traffic demand. We prefer these solutions because they show the upper bound on throwbox performance. This is independent of any particular routing algorithm and only dependent on the single versus multi path assumption. The remainder of this paper is organized along the routing scenarios.

We present algorithms for throwbox deployment for multi-path routing (Cases 1, 2, and 3) in Section III. We present algorithms for throwbox deployment for single path routing (Cases 4, 5, and 6) in Section IV. Section V shows the performance results for these algorithms.

Because deployment for epidemic routing (Cases 7, 8, 9) is simple to describe and is a subset of the other cases, we do not detail this scenario until Section V. We evaluate FIFO-style epidemic routing as an example of an oblivious routing algorithm. In this case a node will forward messages to all other nodes it meets, flooding messages throughout the network. The schedule of packets transmitted during a contact opportunity is in a first-in-first-out order.

III. Capacity Enhancement with Multi-path Routing

In this section, we study the use of throwboxes in the context of multi-path routing (Cases 1, 2, and 3 in Fig. 2). In CTB (Case 1), both contact and traffic information are used as input for deployment. Since full information is available, we can jointly optimize deployment and routing simultaneously. Unfortunately, this joint problem is NP-hard to solve optimally, so we describe a greedy approximation. For case of contact-based deployment (Case 2), where deployment is determined without knowledge of traffic demand, we are able to tractably compute the optimal solution. For oblivious deployment (Case 3) we use a simple random deployment strategy. In all three cases, recall that the routing algorithm has knowledge of demand as well as deployment locations.

Ideally, a throwbox can be placed at any location in an area. However, it would be difficult to solve any deployment problem in a continuous domain because there are an infinite number of potential locations. Instead, we must approximate the problem by dividing the area into a grid of cells and placing throwboxes at the centers of these cells. By using small cells, we can approximate an arbitrary deployment.

A. CTB Deployment (Case 1)

In this case, both contact and traffic information are available for deployment and routing, so we jointly evaluate the issues of deployment and routing for optimal performance. In the following, we formulate this joint problem as a mixed integer programming problem. We then develop a greedy algorithm to solve it.

Maximize λ subject to

$$\sum_{i\in P} f_{ij}^{sd} = \sum_{i\in P} f_{ji}^{sd}, \quad s,d\in N, j\in P-\{s,d\} \tag{1}$$

$$\sum_{i\in P} f_{is}^{sd} = 0, \sum_{i\in P} f_{si}^{sd} = \lambda b_{sd}, \quad s,d\in N \tag{2}$$

$$\sum_{i\in P} f_{id}^{sd} = \lambda b_{sd}, \sum_{i\in P} f_{di}^{sd} = 0, \quad s,d\in N \tag{3}$$

$$\sum_{s,d\in N} (f_{ij}^{sd} + f_{ji}^{sd}) \le c_{ij}, \quad i,j\in N \tag{4}$$

$$\sum_{s,d\in N} (f_{ij}^{sd} + f_{ji}^{sd}) \le c_{ij} x_j, \quad i\in N, j\in L \tag{5}$$

$$\sum_{s,d\in N} (f_{ij}^{sd} + f_{ji}^{sd}) = 0, \quad i,j\in L \tag{6}$$

$$\sum_{i\in L} x_i \le m \tag{7}$$

$$x_i \in \{0,1\}, \quad i\in L \tag{8}$$

$$f_{ij}^{sd} \ge 0, \quad i,j\in P, s,d\in N \tag{9}$$

Fig. 3. Joint deployment and routing problem formulation.

1) Problem Formulation: Let N be the set of n nodes and L be the set of potential throwbox locations. We denote P as the union of N and L. Let x_i be the number of throwboxes that are deployed at location $i \in L$. x_i can be zero or one, indicating whether a throwbox is placed at location i.

Since λ is the total data rate, then λb_{ij} is the data rate from node i to node j. For data sent between a source-destination pair (s,d), we denote the traffic load forwarded from node i to j as f_{ij}^{sd}. We denote $\mathbf{x} = \{x_i\}$ and $\mathbf{f} = \{f_{ij}^{sd}\}$ as the deployment and routing vectors, respectively. Given m throwboxes, the joint problem is to determine a deployment vector $\mathbf{x}$ and a routing vector $\mathbf{f}$ that maximize the total data rate λ. This problem is formulated in Fig. 3. The notation is summarized in Table II.

Constraint (1) represents the flow conservation condition for traffic between a source-destination pair. That is, at every node or throwbox that is not the source or destination, the amount of incoming traffic is equal to the amount of outgoing traffic. Constraint (2) states that source s has no incoming traffic and λb_{sd} outgoing traffic. Similarly, constraint (3) restricts the amount of traffic to and from destinations. Constraint (4) requires that the total amount of traffic between node i and node j is no greater than the average capacity c_{ij} as determined by transfer opportunities. Similarly, constraint (5) enforces the average capacity between nodes and throwboxes. We use x_j to account for whether a throwbox is deployed at location j. Constraint (6) specifies that throwboxes never communicate with each other. Constraint (7) states that the total number of throwboxes is m. Constraint (8)

Algorithm 1 Greedy algorithm for multi-path routing

```
1: $x_i = 0, i \in L$;
2: for $t = 1$ to $m$ do
3:     for all $i \in L$ and $x_i = 0$ do
4:         Compute $\lambda$ when a throwbox is deployed at $i$;
5:     end for
6:     Let $h$ be the location that achieves the maximum $\lambda$;
7:     $x_h = 1$;
8: end for
9: Compute $\lambda$ based on throwbox deployment;
```

specifies that the number of throwboxes deployed at each location is 0 or 1. The last constraint restricts the amount of traffic load between nodes and throwboxes to be non-negative.

Note that the number of throwboxes deployed at each location (i.e., x_i in Fig. 3) must be 1 or 0. This formulation is a mixed integer programming problem, which is generally NP-hard to solve optimally.

2) Greedy Deployment Algorithm: As solving the joint deployment and routing problem optimally is computationally expensive, we develop a greedy heuristic. In this algorithm, throwboxes are deployed iteratively. At each stage, a throwbox is deployed to a location that maximizes λ, the total achieved data rate. Algorithm 1 shows a sketch of the greedy algorithm. Initially, no throwbox has been deployed. That is, $x_i = 0$ for all $i \in L$. Then the algorithm tries to deploy the first throwbox. For each potential location i satisfying $x_i = 0$, i.e., no throwbox is deployed at i, the algorithm will compute the achieved data rate λ when the throwbox is placed at location i. Let h be the location that achieves the maximum λ. A throwbox will be deployed at location h. The algorithm repeats this process until all throwboxes are deployed. Finally, with all throwboxes deployed, the algorithm will compute the total data rate λ and the routing vector.

In step (4) and (9) of the greedy algorithm, we need to compute λ when a throwbox is placed at a potential location or when all throwboxes are deployed. In both cases, the number of throwboxes at each potential location is known. So the formulation in Fig. 3 becomes a linear programming (LP) problem. In fact, the computation of λ in these cases is a concurrent flow problem, which is a classic problem and can be solved using network flow techniques [1].

B. Contact-Based Deployment (Case 2)

We next study multi-path routing with contact-based deployment where throwbox deployment is conducted with only information about the transfer opportunities. In this case, the deployment is designed to *maximize contact opportunities* between nodes. After throwbox locations are determined, we can compute the routing vector to maximize throughput by solving a linear programming problem, as described in the previous section.

Consider two nodes i and j. If a throwbox is deployed at location h, data can be relayed between i and j via the throwbox with data rate $\min\{c_{ih}, c_{jh}\}$. So the contact enhancement is $\min\{c_{ih}, c_{jh}\}$. By considering all pairs of nodes and all locations, we define the absolute contact enhancement as $E_A = \sum_{i,j \in N} \sum_{h \in L} \min\{c_{ih}, c_{jh}\} x_h$ where x_h accounts for whether a throwbox is placed at location h.

Now the deployment problem is to find a deployment vector $\mathbf{x}$ such that E_A is maximized. We solve this problem using a greedy algorithm that is similar to Algorithm 1. The only differences are in steps (4) and (6): we compute E_A for each potential location (in step (4)) and denote h as the location that achieves the maximum E_A (in step (6)).

It is important to note that for Case 2 this greedy algorithm solves the deployment problem optimally. This is because $E_A = \sum_{h \in L} \sum_{i,j \in N} \min\{c_{ih}, c_{jh}\} x_h$. By maximizing E_A at each stage, the algorithm computes the optimal deployment vector.

C. Oblivious Deployment (Case 3)

In the case of oblivious deployment, without any knowledge of transfer opportunities or traffic demand, we are forced to deploy the throwboxes randomly. Given the grid approximation, we choose a random grid location and place the throwbox at that location.

D. Computing Routing (Case 1,2,3)

Once the throwboxes are deployed, the network must determine a routing vector for traffic to follow. For CTB (Case 1), the routing vector is automatically optimized in the joint calculation of deployment and routing. For contact-based and oblivious deployment, the routing vector must be determined by measuring traffic demand between nodes during the operation of the network and solving a linear programming problem to utilize throwboxes for maximum throughput, as described in Section III-A.2.

IV. Capacity Enhancement with Single-path Routing (cases 4,5,6)

In the previous section, we discussed throwbox deployment in the context of multi-path routing. Now we restrict the routing algorithm to use only a single path for all messages between a source-destination pair — MANET routing protocols often follow this paradigm.

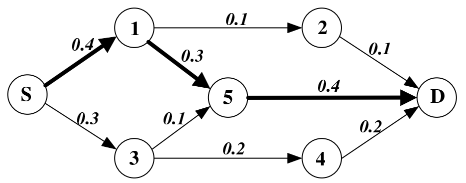


Fig. 4. An example of selecting forwarding paths in single path routing.

We observe that by enforcing this single path constraint, we can solve the deployment and routing problems using the same method. In the following, we focus solely on Case 4 in Fig. 2 where both contact and traffic information are used in deployment and routing. The results can be easily extended to Cases 5 and 6.

Recall that we formulated the joint deployment and routing problem in Case 1, which can be shown to be a NP-hard problem. We now extend the greedy algorithm we developed for multi-path routing to the case of single path routing. Specifically, we modify the computation of the total data rate λ in steps (4) and (9) of Algorithm 1. Recall that we can compute λ by solving a linear programming problem. The resulting routing vector $\mathbf{f}$, however, is for multi-path routing. That is, messages may be forwarded along multiple paths between a source-destination pair. To enforce the single path constraint, we select a single forwarding path for each source-destination pair based on $\mathbf{f}$, as described below. With these routing paths, we compute λ.

We now describe how to select a single path for messages between a source-destination pair. Suppose that $\mathbf{f}$ is the routing vector computed for multi-path routing. We will choose the path that carries the highest traffic load according to $\mathbf{f}$. Fig. 4 shows a simple example where data are forwarded from node S to node D. The number along each edge represents the traffic load in $\mathbf{f}$. The path that achieves the highest traffic load is $S-1-5-D$ and the traffic load is 0.3.

To find the path with the highest traffic load, we use a binary search algorithm, which is illustrated in Algorithm 2. Consider a source-destination pair (s, d) and a routing vector $\mathbf{f}$. A path is referred to as a α-path if the traffic load on all edges of this path is at least α according to $\mathbf{f}$. To determine whether a α-path exists for a specific α, the algorithm generates a graph $G(V, E)$, where V consists all nodes and throwboxes, and E consists of edges with traffic load at least α in $\mathbf{f}$. The algorithm determines whether s and d is connected in $G(V, E)$. If so, there must exist a path from s to d such that the traffic load on each edge is at least α. That is, a α-path exists. Using a binary search, Algorithm 2 finds the maximum α such that a α-path exists. The initial upper bound for α is the maximum load between source/destination and other nodes, and the lower bound is the minimum positive traffic load among all edges in $G(V, E)$. Algorithm 2 terminates when the gap between the upper and lower bounds is no greater than a predefined parameter α_{th}.

Algorithm 2 Compute single path with highest traffic load between a source-destination pair (s,d)

```
1: $\alpha_{up} = \max\{f_{si}^{sd}, f_{id}^{sd} | i \in P\}$;
2: $\alpha_{low} = \min\{f_{ij}^{sd} | f_{ij}^{sd} > 0, i, j \in P\}$;
3: while ($\alpha_{up} - \alpha_{low} > \alpha_{th}$) do
4:    $\alpha_{mid} = (\alpha_{up} + \alpha_{low})/2$;
5:    Generate $G(V, E)$ where $E = \{(i, j) | f_{ij}^{sd} \geq \alpha_{mid}\}$;
6:    if $s$ and $d$ is connected in $G(V, E)$ then
7:       Find a path $\mathbf{p}$ between $s$ and $d$ in $G(V, E)$;
8:       $\alpha_{low} = \alpha_{mid}$;
9:    else
10:      $\alpha_{up} = \alpha_{mid}$;
11:   end if
12: end while
13: Return path $\mathbf{p}$;
```

V. Performance Evaluation

In evaluating our algorithms, we wish to verify several hypotheses: (i) throwboxes enhance throughput in a DTN, (ii) deployments that utilize information about load and jointly optimize for routing see greater gains than those that ignore that information, and (iii) throwboxes enhance connectivity in multiple mobility scenarios. To demonstrate this, we evaluate our deployment and routing algorithms using *ns* simulations. A more detailed description of our simulations can be found in a technical report [27].

A. Methodology and Metrics

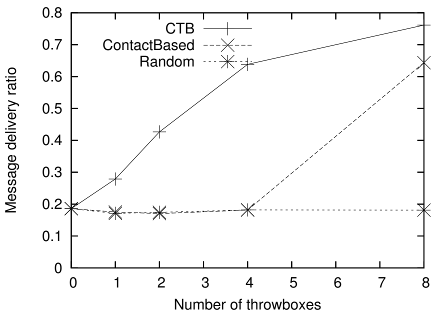
In this paper, we simulate mobile DTNs with various node mobility patterns. Our simulations include two components: computation of deployment and routing vectors, and packet-level simulation using *ns2* simulator [17]. In the computation component, we first calculate the statistics for contacts between nodes by generating node movement for the entire simulation duration. Nodes are considered in contact when the distance between them is less than the radio range. We then determine the set of potential locations for throwbox deployment using the cell based approach described in Section III. In our simulations, the area is 25Km $\times$ 25Km and the cell width is 500m. To reduce the computation time of CTB deployment, we use contact based deployment to select the 50 best locations as the set of potential locations. We compute the deployment and routing vectors using the algorithms proposed in Section III and IV. Based on the computed deployment and routing vectors, we run *ns* simulations according to the forwarding paths and traffic load computed by the computation component.

We consider different node mobility models[1]. The first model is the UMass model based on the UMass bus network of 9 buses repeatedly following fixed, distinct routes [6]. With the traces of real bus movement obtained using GPS devices, we generate synthetic traces by adding variation into the bus movement. Specifically, the traces record the bus locations every minute. In our simulations, we compute the bus speed for every minute based on the traces and vary it by a random factor within [0.95, 1.05]. Each simulation runs for 40,000 seconds in simulation time and message timeout is 20,000 seconds. The second model is the random-waypoint (RWP) model [15]. Under this model, we simulate 40 nodes moving in a 25Km $\times$ 25Km area. Each node repeatedly moves to random locations in the area with random speeds between 7.5m/s and 22.5m/s. We run the simulations for 80,000 simulated seconds and messages timeout is 40,000 seconds. The UMass and RWP models represent different environments, e.g., semi-predictable and constrained movement in the UMass model while unconstrained and random movement in the RWP model.

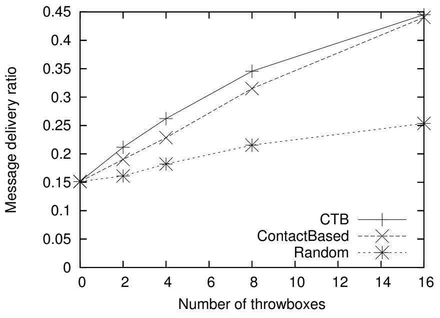
In our simulations, we use the following default settings unless specified otherwise. We use the IEEE 802.11 MAC layer, and the radio range and data rate are 250m and 1Mbps respectively. We consider a uniform traffic model where 20 nodes are chosen as sources with random destinations. Each source generates messages at the same data rate according to a Poisson process. Messages are 1,500 bytes. The buffer size of both nodes and throwboxes is 50,000 messages.

For node discovery, each node broadcasts a beacon message every 5 seconds. To avoid duplicate message transmissions, nodes first exchange meta data to determine which messages should be sent. Messages are transmitted from the buffer in a FIFO order. When the buffer overflows, a node will drop the first message in the buffer. Each result is averaged over five runs with different random seeds.

We consider two performance metrics, namely message delivery ratio and delay. The *message delivery ratio* is defined as the ratio between the number of unique messages being delivered and the number of generated messages. Messages might be dropped because of buffer overflows or message timeout. The delivery ratio is computed over the simulation duration, which measures how successful each scheme is in delivering messages. The *message delay* is the average time from the generation of a message to the earliest reception of the message at the destination. The message delay considers delivered messages only.



(a) UMass



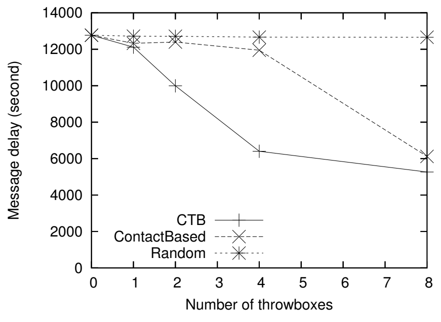
(b) RWP

Fig. 5. Message delivery ratio under multi-path routing.

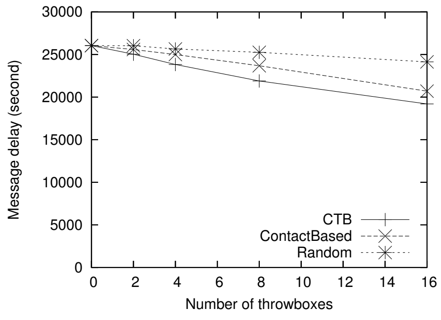
B. Multi-Path Routing (Case 1,2,3)

We first consider the case of multi-path routing. Fig. 5 shows the delivery ratio for the UMass and RWP models. The total traffic load is relatively high, i.e., 334Kbps for the UMass model and 125Kbps for the RWP model. We make the following observations. First, the delivery ratios improve significantly as the number of throwboxes increases under CTB deployment. For example, for CTB deployment under the UMass model, the delivery ratio increases by a *factor of three* using four throwboxes. Throwboxes are able to improve delivery ratios under both regular mobility (i.e., UMass) and random mobility (e.g., RWP). Second, contact based deployment generally performs worse than CTB deployment, especially in the UMass model. This is because CTB deployment is able to utilize traffic information to optimize performance. Third, for oblivious deployment, the delivery ratio increases under the RWP mobility model but is not affected in the UMass model. This is because under UMass model, due to the constrained node movement and the relatively large span of the area, random deployment tends to place throwboxes to locations where nodes do not visit. In contrast, due to random movement, nodes in the RWP models would meet with throwboxes eventually even though throwboxes are placed randomly in the area.

[1]Due to space limitation, we omit the results for the Manhattan model, which can be found in [27].

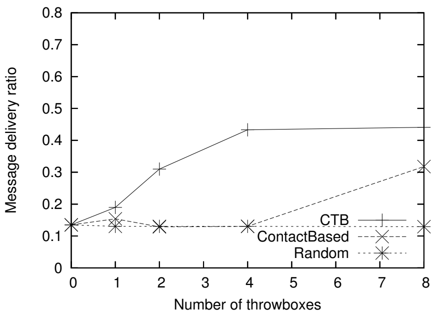


(a) UMass

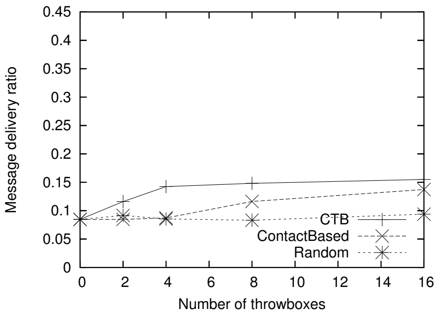


(b) RWP

Fig. 6. Message delay under multi-path routing.



(a) UMass



(b) RWP

Fig. 7. Message delivery ratio under single path routing.

Fig. 6(a) shows the message delay when nodes follow the UMass model and the traffic load is high at 334Kbps. We can see that the use of throwboxes can significantly reduce the message delay. For example, the message delay is reduced from 12,000s to 6,000s using four throwboxes. This is because nodes are able to communicate with each other more frequently via throwboxes. The contact based deployment achieves less reduction in message delay. As expected, random deployment does not affect the message delay. We have obtained similar results for message delay when the traffic load is low. Fig. 6(b) depicts the message delay for the RWP model when the traffic load is 125Kbps. It can be seen that the use of throwboxes reduces the message delay. However, the improvement is less evident than in the UMass model.

C. Single Path Routing (Case 4,5,6)

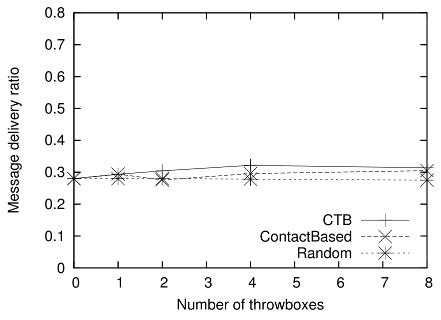
In this section, we consider the case of single path routing. Fig. 7(a) depicts the delivery ratio for the UMass model when the traffic load is at 125Kbps. We can see that as the number of throwboxes increases, the delivery ratio improves with CTB deployment. Contact based deployment, on the other hand, improves the delivery ratio only when the number of throwboxes is relatively large. This confirms that using traffic information improves the effectiveness of throwbox deployment. Oblivious deployment, as expected, does not affect the delivery ratio.

Fig. 7(b) shows the delivery ratio for the RWP model when the traffic load is 42Kbps. Due to the random movement of nodes and the relatively large span of the area, the delivery ratio is generally low. We observe that throwboxes do improve the delivery ratio. In addition, CTB performs better than contact based deployment.

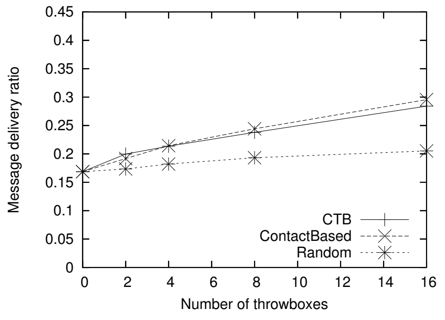
As compared to multi-path routing, we observe that single path routing achieves lower delivery ratios and is more sensitive to the deployment scheme used. This is because in single path routing, the utility of throwboxes for data delivery between a source/destination pair depends on those throwboxes that are able to support the highest data rate. Multi-path routing, on the other hand, is able to utilize all throwboxes available for data forwarding, leading to better delivery ratios. In addition, our results, which are elided here, show that with throwboxes, message delay is reduced under the UMass model but not affected under the RWP model. Generally, throwboxes are less effective in reducing message delay under single path routing than under multi-path routing.

D. Epidemic Routing (Case 7,8,9)

We now consider the case of epidemic routing. Epidemic routing does not utilize any information about the network. Since message forwarding is fixed based on actual contact between nodes, we focus on the deployment issue. Specifically, we compute throwbox locations as in multi-path routing. The intuition is that epidemic routing is able to exploit all paths available to propagate



(a) UMass



(b) RWP

Fig. 8. Message delivery ratio under epidemic routing.

messages. So epidemic routing would benefit from the use of throwboxes even if the deployment is intended for multi-path routing.

Fig. 8 shows the delivery ratio of epidemic routing. Fig. 8(a) depicts the case of the UMass mobility model when the traffic load is 334Kbps. We can see that all deployment schemes achieve similar delivery ratios. As compared to the cases of multi-path and single path routing, throwboxes have limited improvement on the delivery ratio because of the poor utilization of resources caused by message flooding in epidemic routing.

Fig. 8(b) depicts the delivery ratio under the RWP mobility model when the traffic load is 42Kbps. We observe that contact-based and CTB deployment achieve significant improvement in the delivery ratio, while random deployment has modest improvement. Epidemic routing benefits from the use of throwboxes for all deployment schemes because of the random movement of nodes.

VI. Related Work

DTNs are a general class of networks that exhibit non-Internet-like characteristics, e.g., intermittent connectivity, large delay, and high link loss. DTNs can be applied to military networks [9], deep space communication [21], and everyday vehicular scenarios [25], [6]. To achieve interoperability between various types of DTNs, Fall [11] proposes an architecture that is based on an asynchronous message forwarding paradigm. Jain et al. [14] study unicast routing in DTNs and develop several routing algorithms based on different levels of knowledge about the network.

Most related work focuses on algorithms that exploit node mobility to deliver data. *Reactive* approaches rely on the inherent movement of devices or users to help deliver data. Davis et al. [10] evaluate several algorithms for managing DTN routing when buffers are the limited resource in the network. Shah et al. [22] propose to exploit mobile entities to transport data to conserve energy in resource-limited sensors. *Proactive* approaches dictate or restrict the movement of devices for routing. Li and Rus [16] consider proactive movement of nodes to deliver messages in a disconnected environment. Zhao et al. [26] propose the use of special nodes called *message ferries* to provide communication services and exploit controlled mobility to improve performance. Burns et al. [7] present a multi-objective control approach to determine movement of mobile robotic agents to enhance performance. Other works include [13], [18], [24].

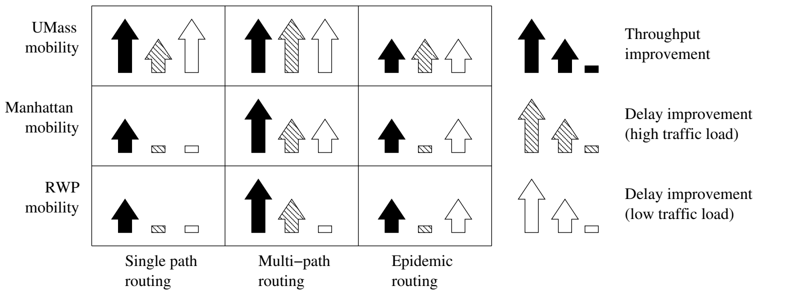
The node placement problem has been studied in wireless [20] and sensor networks [8], [19]. In [20], the authors study the placement of access points to form a mesh network for Internet traffic and proposes greedy algorithms. The work in [8] studies the placement of relaying nodes in a sensor network to maintain global connectivity. The work in [19] studies node placement in sensor networks to prolong network lifetime. In contrast, our paper considers mobile DTNs where no end-to-end path exists between nodes. In addition, our paper studies various routing and deployment approaches, including epidemic and single path routing.

Our work is also related to the study on Infostation [12] and facility location [5]. Facility location has been studied extensively in the operation research community and generally considers selecting facility locations to minimize cost for specified demand.

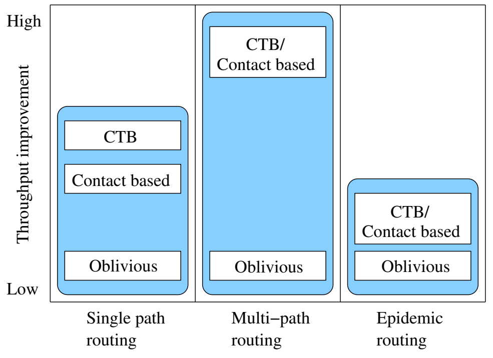
VII. Conclusion

We proposed the use of throwboxes to enhance network capacity in mobile DTNs. By relaying data, throwboxes increase the transmission opportunities and throughput between nodes. We presented a framework to systematically study the issues of deployment and routing, and we developed algorithms for various deployment and routing approaches.

We evaluated different routing and deployment approaches using *ns* simulations. Fig. 9(a) summarizes our results on the utility of throwboxes in performance enhancement. First, throwboxes are very effective in improving throughput and delay, especially when node movement is regular or multi-path routing is used. As



(a) Performance improvement using throwboxes under CTB deployment



(b) Conceptual performance of various deployment schemes

Fig. 9. Summary of simulation results.

compared to multi-path routing, single path routing is less effective in using throwboxes because data forwarding is limited along a single path. Due to the poor utilization of network resources, epidemic routing achieves the least improvement when using throwboxes. Second, the improvement in throughput is generally more significant than delay. Third, single path routing is most sensitive to deployment locations because single path routing limits data forwarding to a single path. On the other hand, epidemic routing is the least sensitive due to the poor utilization of resources. Fig. 9(b) shows the conceptual performance of different deployment schemes. We found that both CTB and contact based deployment perform well. CTB deployment, which utilizes both traffic and contact information, achieves better performance when single path routing is used. As expected, random deployment performs poorly.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] I. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: A survey. *Computer Networks Journal (Elsevier)*, March 2005.

[3] N. Banerjee, M. Corner, and B. Levine. An energy-efficient architecture for DTN throwboxes. Technical Report 06-39, University of Massachusetts-Amherst, 2006.

[4] N. Banerjee, J. Sorber, M. Corner, S. Rollins, and D. Ganesan. Triage: A power-aware software architecture for tiered microservers. Technical Report 05-22, University of Massachusetts-Amherst, April, 2005.

[5] M. L. Brandeau and S. S. Chiu. An overview of representative problems in location research. *Management Science*, 35(6):645–674, 1989.

[6] J. Burgess, B. Gallagher, D. Jensen, and B. Levine. MaxProp: Routing for vehicle-based delay-tolerant networks. In *IEEE INFOCOM*, April 2006.

[7] B. Burns, O. Brock, and B. N. Levine. MV routing and capacity building in disruption tolerant networks. In *IEEE INFOCOM*, March 2005.

[8] X. Cheng, D. Du, L. Wang, and B. Xu. Relay sensor placement in wireless sensor networks. *IEEE Transactions on Computers*, 2001.

[9] DARPA Disruption Tolerant Networking Program, http://www.darpa.mil/ato/solicit/DTN, July 2006.

[10] J. Davis, A. Fagg, and B. Levine. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. In *International Symposium on Wearable Computing*, October 2001.

[11] K. Fall. A delay-tolerant network architecture for challenged internets. In *ACM SIGCOMM*, 2003.

[12] D. Goodman, J. Borras, N. Mandayam, and R. Yates. INFOSTATIONS: A new system model for data and messaging services. In *IEEE VTC'97*, pages 969–973, May 1997.

[13] A. A. Hasson, R. Fletcher, and A. Pentland. DakNet: A road to universal broadband connectivity. *Wireless Internet UN ICT Conference Case Study*, 2003.

[14] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *ACM SIGCOMM*, Portland, OR, 2004.

[15] D. Johnson and D. Maltz. Dynamic source routing in ad-hoc wireless networks. In *ACM SIGCOMM*, August 1996.

[16] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *ACM MOBICOM*, August 2000.

[17] Network simulator. http://www.isi.edu/nsnam/ns, July 2006.

[18] P. Juang et al. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. In *ASPLOS*, October 2002.

[19] J. Pan, Y. Hou, L. Cai, Y. Shi, and S. Shen. Topology control for wireless sensor networks. In *ACM MOBICOM*, San Diego, CA, 2003.

[20] L. Qiu, R. Chandra, K. Jain, and M. Mahdian. Optimizing the placement of integration points in multi-hop wireless networks. In *IEEE ICNP*, 2004.

[21] S. Burleigh et al. Delay-tolerant networking – an approach to interplanetary internet. *IEEE Communications Magazine*, June 2003.

[22] R. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a three-tier architecture for sparse sensor networks. In *IEEE SNPA Workshop*, 2003.

[23] StarGate, http://www.xbow.com/Products/XScale.htm, 2006.

[24] Tara Small and Zygmunt Haas. The Shared Wireless Infostation Model - A New Ad Hoc Networking Paradigm (or Where there is a Whale, there is a Way). In *ACM MobiHoc*, June 2003.

[25] H. Wu, R. Fujimoto, and G. Riley. Analytical models for data dissemination in vehicle-to-vehicle networks. In *IEEE VTC 2004/Fall*, 2004.

[26] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *ACM MobiHoc*, Tokyo Japan, May 2004.

[27] W. Zhao, Y. Chen, M. Ammar, M. Corner, B. Levine, and E. Zegura. Capacity enhancement using throwboxes in mobile delay tolerant networks. Technical report, College of Computing, Georgia Institute of Technology, 2006.