An Energy-Efficient Architecture for DTN Throwboxes

Nilanjan Banerjee
Department of Computer Science
Univ. of Massachusetts, Amherst
Email: nilanb@cs.umass.edu

Mark D. Corner
Department of Computer Science
Univ. of Massachusetts, Amherst
Email: mcorner@cs.umass.edu

Brian Neil Levine
Department of Computer Science
Univ. of Massachusetts, Amherst
Email: brian@cs.umass.edu

***Abstract*—Disruption Tolerant Networks rely on intermittent contacts between mobile nodes to deliver packets using store-carry-and-forward paradigm. The key to improving performance in DTNs is to engineer a greater number of transfer opportunities. We earlier proposed the use of throwbox nodes, which are stationary, battery powered nodes with storage and processing, to enhance the capacity of DTNs. However, the use of throwboxes without efficient power management is minimally effective. If the nodes are too liberal with their energy consumption, they will fail prematurely. However if they are too conservative, they may miss important transfer opportunities, hence increasing lifetime without improving performance.**

In this paper, we present a hardware and software architecture for energy efficient throwboxes in DTNs. We propose a hardware platform that uses a multi-tiered, multi-radio, scalable, solar powered platform. The throwbox employs an approximate heuristic for solving the NP-Hard problem of meeting an average power constraint while maximizing the number of bytes forwarded by it. We built and deployed prototype throwboxes in UMassDieselNet – a bus DTN testbed. Through extensive trace-driven simulations and prototype deployment we show that a single throwbox with a 270 cm^2 solar panel can run perpetually while improving packet delivery by 37% and reducing message delivery latency by at least 10% in the network.

I. Introduction

Recent efforts in Disruption Tolerant Networking (DTN) promise networking infrastructures that are robust to widespread failures and operate in highly challenged environments [8], [9], [3]. Such networks can be used to route information in completely unconnected and decentralized scenarios, including natural disasters, sparse sensor deployments, underwater networking, and highly mobile systems.

DTNs rely on intermittent contacts between mobile nodes to deliver packets using a store-carry-and-forward paradigm. The routing performance in a DTN, including deliverability and delay, is critically dependent on the node mobility patterns that drive the frequency, duration, and sequence of contact opportunities. One method to improve these metrics is to engineer a greater number of opportunities. For instance, several papers have proposed altering the mobility of nodes to enhance network performance [4], [5], [26] Unfortunately, movement patterns are often inherent to the nodes and cannot be modified.

Another solution for improving DTN performance is to place additional *stationary* nodes in the network, which increases the number and frequency of contact opportunities. In previous work, we proposed the use of *throwboxes* within a DTN for this purpose [27]. Throwboxes are inexpensive, battery-powered, stationary nodes with radios and storage. When two nodes pass by the same location at different times, the throwbox acts as a router, creating a new contact opportunity. In our previous work, we focused on algorithms that place throwboxes in locations that maximize network performance. We found that even a small number of throwboxes can significantly increase the delivery rate found in a DTN [27]. However, this addressed only part of the problem.

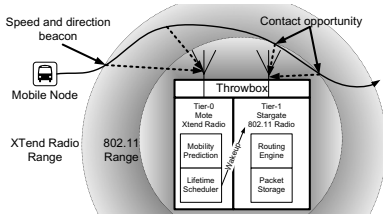

Fig. 1. Overview of our throwbox architecture.

Throwbox deployment without the use of power management is minimally effective. The high power consumption of always-on nodes results in a short lifetime; conversely, nodes powering on and off intermittently without regard to node mobility may miss numerous connection opportunities, increasing lifetime without necessarily improving performance. The key goal is to provide energy efficiency without sacrificing network performance. Moreover, throwboxes are constrained to consume power at a rate dictated by a battery lifetime requirement or the availability of scavenged energy, such as solar power. In this paper, we address these issues in the context of the following challenge: *how can a throwbox in a DTN meet an average power constraint and simultaneously maximize the number of packets forwarded by it?*

We show that the primary source of overhead is the energy cost of neighbor discovery: idling or turning the platform and radio on and off to search for contacts in a sparse network wastes the vast majority of node energy. Due to the sparseness of a DTN, waking up for non-existent, or brief, contact opportunities is not worth the cost of turning on the platform and radio.

In this paper, we propose a novel paradigm for power

management in DTNs that provides more efficient neighbor discovery by detecting the mobility of other nodes at a minimum cost and predicting the cost and opportunity of each possible contact. Using these predictions, the throwbox can intelligently choose the most fruitful contact opportunities, and it can limit the number of contact opportunities to meet energy constraints. In our architecture, we pair a *tier-1* platform — a PDA-like Stargate and 802.11 radio — with a low-power *tier-0* platform — a Mote and XTend radio. The XTend radio is a *long-range, low-bitrate* radio for neighbor and mobility discovery. It has a range of more than a kilometer, which is more than five times the range of WiFi radios. By coupling the XTend radio with a low-power Mote, we reduce the total energy cost of the throwbox platform.

Figure 1 presents an overview of our approach and the problems we address in this paper. Mobile nodes (shown as a bus in the diagram) beacon their position, direction, and speed using the long-distance radio. While listening for other nodes, the throwbox needs very little computational power and memory, and the tier-0 platform is sufficient to process the beacons. If a throwbox hears a beacon, tier-0 predicts if, when, and for how long the mobile node will come in contact with the tier-1, 802.11 radio. Using a constrained single-objective optimization to meet a power consumption target, the node decides whether to take the transfer opportunity. If it does, tier-0 wakes the Stargate and WiFi radio in advance of the mobile node entering radio range, which then transfers DTN data. After the contact opportunity, the tier-1 platform returns to a sleep state. For this scenario, we develop a method for predicting the contact opportunities based on the observed mobility of remote nodes and an algorithm for choosing which contacts to take given power constraints.

We meet our objectives through a mix of theoretical and practical results. In Section III, we present a mobility prediction algorithm that is sufficiently lightweight to operate on a Mote. In Section IV, we detail our scheduling algorithm that helps meet an average power constraint for the throwbox while maximizing the number of packets delivered in the network. We present theoretical bounds on the complexity of the optimization problem and the performance of an online algorithm based on token-buckets. We show that the problem is NP-Hard and present competitive bounds on the performance of the algorithms for a simple node mobility pattern. In Section V, we detail our design and implementation of a prototype throwbox and its deployment and integration into the UMassDieselNet DTN [3]. And finally, we show the performance of the system through an extensive set of trace-driven simulations and experiments on the deployed platform. Our methods reduce the power consumption of the throwbox from 2500 mW to 80 mW while delivering almost as many packets. Stated another way, we have deployed a throwbox that runs perpetually on solar cells similar to the size of the box itself.

II. Background

Peers in a mobile network alternate between two basic operations: *neighbor discovery* and *opportunistically transferring data*. In DTNs the latter operation has received much attention as part of routing protocols [3], [8]. However, in a sparse DTN network, searching for other nodes consumes a large percentage of time in comparison to transferring data. Consequently, searching for other nodes becomes the dominant drain on the energy of a battery-powered DTN node. For instance, we show in this paper that using an 802.11 radio to search for contacts in a DTN devotes 99.5% of the total energy just to find other nodes to exchange data with. Simple power management schemes are insufficient—a radio in idle mode has little savings as compared to its listening mode.

Previous work has addressed mobile system power management by using 802.11 radios for data transfers and *low-power, short-range* radios (e.g., 802.15.4 [14], Bluetooth [15], or CC1000 [16]) for neighbor discovery tasks. While this solution is appropriate for dense mobile networks, our recent work has shown it is inefficient for sparsely populated DTNs [11]—we have recently strengthened this claim through analytical results [2]. This is because short-range radios miss too many connection opportunities, which tend to be short-lived compared to the delay and cost of switching 802.11 radios from sleep to wake modes. Moreover, these architectures do not reduce the cost of other idle hardware components, predict the mobility of the node, nor choose which opportunities to take.

A key to realizing the gains of a second radio is to be able to predict neighbor mobility *before* the node is within contact of the throwbox. Mobility prediction algorithms in the past have been used to predict future network topology to perform route reconstruction proactively [23], accurately predicting hand-offs and bandwidth provisioning in cellular networks [6], [20], and location tracking in wireless ATM networks [13]. Such algorithms either require coordinated information from more than one base station [20], [21], they are computationally too expensive to implement on a extremely low power tier-0 subsystem [21], or they are built assuming very specific movement patterns of nodes [20]. Most mobility predictors are assumed to be implemented on powerful base stations, which are unconstrained platforms. Mobility prediction has also been used in DTNs, taking advantage of schedules for mobile nodes in the network [12]. In our work, we develop prediction algorithms that are lightweight enough to run on a small, extremely efficient platform, such as a Mote. This helps minimize the cost for deciding which contact opportunities to take. Additionally our technique does not depend on any centralized infrastructure, or scheduled node mobility.

III. Mobility Prediction Engine

Because DTNs are sparsely populated, throwboxes must expend significant energy searching for contact opportunities. Because of node mobility, throwboxes have a limited amount of time to exchange data with peers. Searching for contacts efficiently requires waking and sleeping rapidly (called *duty cycling*), and conversely, data transfer requires high-bandwidth connections. Our design philosophy is to separate neighbor discovery from data transfer, and to divide these tasks across the hardware that is best suited to each task. The long-distance

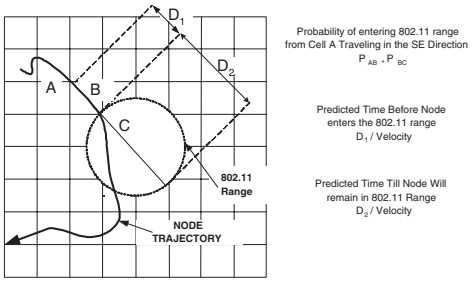

Fig. 2. The figure depicts the working of the Mobility Prediction Engine

radio and tier-0 platform can duty-cycle, listen, and idle efficiently, while the high-bandwidth radio and tier-1 platform can transfer large amounts of data during limited connection opportunities. While the tier-0 platform is listening for contacts, the tier-1 platform remains asleep.

The decision to wake the tier-1 platform and radio depends on an engine that detects, and predicts the mobility of nodes in the network, based on information gathered by the tier-0 platform and radio. By predicting the trajectory of an oncoming mobile node, the tier-0 platform can determine if, when, and for how long it will enter the range of the data transfer radio. If the throwbox determines that the mobile node will not come in contact with its data transfer radio, it can safely ignore that node. If it determines that the mobile node will enter at a certain time, it must make sure that the throwbox can make the most out of the contact opportunity. When the throwbox estimates the time that the mobile node will enter 802.11 range, it wakes the tier-1 node in advance of the mobile node arriving—a process that can take several seconds. The predicted length of the contact opportunity is then used to determine which contact opportunities to take, a process we explain in Section IV.

To determine trajectories, we require that mobile nodes periodically transmit location, speed, and direction over the long-distance radio as a beaconing message. By requiring mobile nodes to beacon their positions, we have shifted some of the energy burden from the stationary throwboxes to the mobile nodes. In this paper, we study the use of throwboxes in a vehicular network, where the mobile nodes have plentiful energy sources. Nonetheless, the techniques will aid systems where all nodes are energy constrained as it lowers the cost of discovery; however, we leave this as future work.

A. Prediction Algorithm

We model the movement pattern of the mobile nodes as a Markovian process. Hence, we assume the location of a node at time T_i is dependent only on its location at time T_{i-1}. Such a model is accurate for movement patterns that are both predictable and semi-predictable.

The algorithm models the problem as a virtual square of width $2r$, where r is the radius of the long-range radio. The throwbox is located at the center of the square, as illustrated in Figure 2. The large square is divided into square cells numbered from 0 to k. The algorithm makes use of a probability transition matrix T for mobile nodes in the network. Entry T_{ij} of the matrix is the probability that a node will transition to cell j given that it is in cell i. The transition matrix T is learned by the throwbox over time.

Mobile nodes beacon data in the form of tuples $< p, v, t, d >$, where p is the present GPS location of the mobile node, v is its speed, t is the time at which the data was sent, and d is the direction of motion of the node. When a node approaches, the throwboxes collect a set of tuples $\{< p_1, v_1, t_1, d_1 >, ..., < p_k, v_k, t_k, d_k >\}$ transmitted by the node. Using this set, as well as historic information in the form of the transition matrix T, the algorithm estimates :

- The probability that the mobile node would be in any cell within the data transfer radio range after time ΔT, where ΔT is the time required to wakeup tier-1 and start transferring application data—it is typically on the order of seconds
- The predicted time the node will spend in range of the data transfer radio and, subsequently, the amount of data it is likely to transfer

Figure 2 illustrates this process. At time $t = 0$, the mobile node's beacon from cell A is received. By dead-reckoning along a straight line of motion, the prediction engine estimates D_1, the distance until the node is within 802.11 range, and D_2, the distance for which the node would be within range, Accordingly, the predicted length of time until the mobile node reaches the range of the data radio is D_1/v; if this time is greater than the transition time to wakeup tier-1, then the mobile node is ignored. Otherwise, the throwbox calculates Pr, the probability that the node will enter the range of the data radio. (In the simple example presented in the figure, Pr is the product of the probability of transition from cell A to B and from B to C; i.e., $Pr = T_{AB} \cdot T_{BC}$.) The expected duration of time the mobile node will stay in data radio range is $Pr \cdot (D_2/v)$. Below we show how we calculate Pr under Markovian assumptions.

B. Estimating the Probability of Entering Data Radio Range

The prediction engine assumes that the node would move in the direction that it was last seen. Therefore, it constructs a straight line in the current direction of motion and finds the intersection with the range of the data-transfer radio range (idealized as a circle). Let c be the cell at which the node enters the data radio range; let D_{c,c_k} be the Euclidean distance between the present location of the node and the location of the point where it enters the data transfer radio range (assuming the present direction of motion of the node); let v_k be the speed of the node; and let $c_i, \ldots, c_{i+j}$ be a sequence of cells that are in the predicted path of motion of the node to the data transfer circle. Let $\{d_1, \ldots, d_i\}$ be the set of i possible directions of motion. For example, a direction could be north, south, east or west. The probability that a node enters the data transfer radio range after time ΔT is given by

$$Pr[X_{t_k+\Delta T} \leq R | (X_{t_k} = p_k, d_{t_k} = d_k), \ldots, (X_1 = p_1, d_{t_1} = d_1)], \quad (1)$$

where X_t is the position of the node at time t and R is the range of the data transfer radio. The above probability can be approximated as

$$P_{appr} = Pr[X_{t_k+\Delta T} \leq R | X_{t_k} = p_k, d_{t_k} = d_k] \quad (2)$$

assuming that the node movement is modeled as a Markovian process. P_{appr} is evaluated by the algorithm as

$$Pr[X_{t_k+\Delta T} \leq R | (X_{t_k} = p_k, d_{t_k} = d_k)] = \begin{cases} T_{c_i,c_{i+1}} \cdot \ldots \cdot T_{c_{i+j-1}c_{i+j}}, & \frac{D_{c,c_k}}{v_k} \leq \Delta T \\ 0, & \frac{D_{c,c_k}}{v_k} > \Delta T \end{cases} \quad (3)$$

Equation 3 shows that the probability of a node entering a cell of the data transfer radio after time ΔT is equal to the transition probability of the node from its present cell to cell c_{i+j} if the node is traversing fast enough to cover the shortest distance between the two cells in time ΔT.

C. Discussion

In contrast with previous work (e.g., [21], [20]), our mobility prediction engine is more general and sufficiently simple for implementation on our energy efficient resource-constrained tier-0 platform (10 KB of memory and an 8MHz processor).

If multiple contacts are simultaneously discovered by the throwbox, it takes the size of the connection event as the sum of the predicted connection size of each contact. Note that the size of the connection event refers to the expected amount of data send during a connection event.

We also note that while searching for mobile nodes, the long-distance radio does not need to be constantly powered; it needs to only wake-up often enough to predict if, when, and for how long the mobile node will be within range of the data transfer radio. This is consistent with past approaches to lowering wireless power consumption such as PSM, which duty-cycle the radio to save energy [1]. Our throwbox design uses a MaxStream XTend radio that supports several very efficient duty-cycling modes at the hardware and MAC layer. The MAC protocol built into the long-range radios is very similar to the B-MAC protocol built for CC1000 Mote radios [17]. We have implemented an adaptive controller for duty cycling the radio, but due to space limitations, we refer the reader to a technical report [2].

IV. Token Bucket Lifetime Scheduler

Mobile nodes may present a throwbox with many transfer opportunities. However, because they have limited energy, they may not participate in every opportunity. Additionally, mobile nodes may only skirt the outside range of the data transfer radio, and by the time tier-1 platform is awake, the contact may have moved out of range.

A limited energy supply can be viewed as a constraint on the *average power* that the system consumes in two different scenarios. First, the throwbox may be designed to last for a certain period of time—computing the average power from the capacity of the batteries is straightforward. Second, throwboxes may be capable of scavenging energy, such as the solar panels used in our prototype system. In that case, we assume the average power constraint to be the average power produced by the solar cells. This is a simplification, as there is great variance in radiant solar energy and the battery must be large enough to smooth fluctuations; however, we omit this complexity in this paper.

In this section, we first show that computing the optimal subset of opportunities to participate in is NP-Hard. We then present our sub-optimal solution, which is linear in the number of connections and requires constant memory. In Section VI, we show that our algorithm performs within 80% of optimal.

Optimization criteria other than energy can be considered concurrently. For example, mobile nodes may transmit the priority of packets that they are carrying, or the throwbox can decide on which opportunities to take based on the likelihood of being able to route the packet to its final destination. We leave these possibilities as future work.

A. Complexity of the Optimization Problem

The most efficient strategy for throwboxes is to amortize the transition energy over the largest transfer opportunities. For example, if a mobile node skirts the outer range of the throwbox, it is relatively expensive to wake the tier-1 system for such a short opportunity. However, determining the optimal set of transfer opportunities to wake the tier-1 system is NP-Hard, even as an offline problem, as we show below.

The inputs to the above optimization problem, O, are a set of connection events $E = \{E_1, E_2, ..., E_n\}$ that occur within time interval $[0, t]$ and the throwbox energy constraint $P \cdot t$. Each connection event E_i is associated with an energy cost e_i that includes the transition energy to wake up tier-1 and the data transfer radio as well as the amount of energy spent while transferring data during the connection event. Let b_i be the number of bytes likely to be forwarded as a result of the data transfer during E_i. The solution to the optimization problem O is a subset of events from E_s, such that energy goal $P \cdot t$ is not exceeded and maximum number of bytes are forwarded by the throwbox.

The decision version of this problem, O_d, takes as input an additional positive number k. A solution to O_d does not spend more than $P \cdot t$ energy and forwards at least k bytes of data. We prove below that the above decision problem is NP-Complete.

Theorem 1. *O_d is NP-Complete.*

Proof: We first prove that O_d is in NP. We nondeterministically guess a subset E_s of connection events and check the following: (1) If $\sum_{E_i \in E_s} e_{E_i} \leq P \cdot t - e_I$, where e_I is the idle energy consumption of the system in time t. The idle energy includes the energy consumed by the tier-0 system and the discovery radio. (2) If $\sum_{E_i \in E_s} b_{E_i} \geq k$. These checks can be done in time polynomial in the input size. Therefore, $O_d \in NP$.

We next prove that $O_d \in$ NP-Hard through a poly-time reduction from the 0-1 knapsack problem. An instance of the 0-1 knapsack problem is a set of items $I = \{I_1, I_2, ..., I_n\}$, a capacity C and a number V. Each item is associated with a weight w and a value v. The goal is to find a subset of the

items such that the sum of their weights is less than or equal to C and the sum of their values is greater than or equal to V. Given an instance of the knapsack problem, an instance of the problem O_d is generated as follows.

A connection event E_i is taken as the same as an item I_i of the knapsack. The energy cost and the number of bytes transfered during E_i is taken as the same as the weight and value of the item respectively. The energy constraint $P \cdot t - e_I$ is taken as equal to the capacity C of the knapsack and k is taken as equal to V. It is clear that a solution to the knapsack problem exists if and only if there is a subset of the connection events that meets the energy constraint and forwards at least k bytes.

■

B. A Token-Bucket Approach

Since O_d is NP-Complete, it is clear that for the online version of the above optimization problem an exact solution would turn out to be too costly to obtain. However, an approximate online solution would follow the following intuition: since an online algorithm has no knowledge of the future, at any time t it should take a contact opportunity only if the energy cost does not lead to a violation of the average power constraint till time t. In other words, the heuristic should regulate energy flow based on the average power constraint. Moreover, since contact opportunities may occur in bursts, the scheduler should also be capable of handling bursts of energy consumption.

Token buckets have been used in a similar scenario in networking to regulate network traffic [24]. Token buckets allow bursty traffic to continue transmitting while there are tokens in the bucket, up to a user-configurable threshold thereby accommodating traffic flows with bursty characteristics [10].

Algorithm 1 Token Bucket Scheduler

```
Average power constraint = P
Generate energy tokens at the rate of P/sec
Energy cost of present connection event = E_p
Size of present connection event in bytes = b_p
Number of tokens accumulated till now = m
if m · P < E_p then
    Do not wakeup Tier-1
else
    Mean size of the last k connection events (in bytes) = b_k
    Mean energy of the last k connection events = E_k
    Mean inter-arrival time of connection events = T_m
    if m · P − E_p + T_m · P < E_k and b_p < b_k then
        Do not wakeup Tier-1
    else
        Wakeup Tier-1
    end if
end if
```

The scheme, shown in Algorithm 1, generates energy tokens at the rate of the average power constraint. For any given connection event, if the number of tokens accumulated is less than the amount of energy required, the event is ignored. However, if the number of tokens accumulated is greater than the energy required for a connection event the system makes a simple choice: should it take this connection event, the next connection event, or both. First, the algorithm estimates the size and energy cost of the current connection event based on the output of the mobility prediction engine. It then estimates the size and energy cost of the next connection event based on the mean of the last k connection events and assumes that the connection event arrives at the mean inter-arrival time of the last k connections. Taking into account the number of tokens that are accumulated between now and the next connection event, if it estimates it can take both connection events it takes the current one. Otherwise it chooses the larger of the two. This process repeats for the next contact opportunity.

The algorithm runs in time linear in the number of connection events and requires $O(k)$ amount of space. If k is a constant, the space required by the algorithm is a constant. Therefore, the algorithm is space and time efficient and is easily implementable on the low power resource constrained tier-0.

We perform a competitive analysis on the performance of the token bucket scheduler for a simple mobility model. We prove (proof omitted due to lack of space — see [2]) that for systems similar to UMassDieselNet, the token bucket scheduler is at most 4-competitive. In Section VI, we show that the algorithm is 1.25-competitive, better than the above bound.

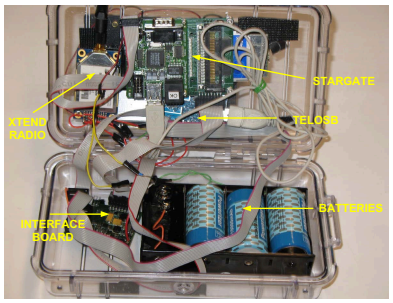

Fig. 3. Prototype throwbox system

Characteristic	XTend	Dlink-Air
Frequency	900 MHz	2.4GHz
Ranges	1000 m	150 m
Receive Power	360 mW	1000 mW
Transmit Power (max)	1 W	1.2 W

Fig. 4. Characteristics of the two radios

V. Throwbox Prototype Implementation and Deployment

We constructed prototype throwboxes, shown in Figure 3, using a Crossbow Stargate (tier-1) [25] and a TelosB Mote (tier-0) [18]. We chose these hardware platforms because they handle the two DTN activities, data transfer and neighbor discovery, efficiently. The Stargate platform runs Linux, allowing us to run the same Java-based routing software used in UMassDieselNet.

The Stargate contains a 32-bit, 400 MHz PXA255 XScale processor, 64 MB of RAM, 32 MB of internal flash, and a DLink-Air 802.11b interface. The TelosB Mote contains an

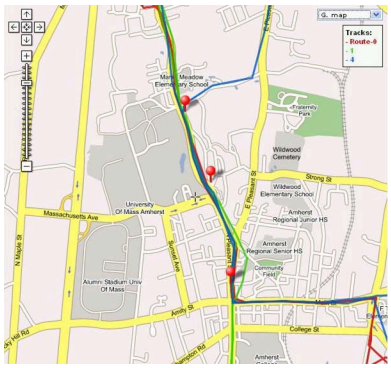

Fig. 5. The Map shows the locations where the throwboxes were placed

8-bit, 8MHz microcontroller, 10 KB of RAM, and 1 MB of external flash. The Mote is attached to a Maxstream XTend 900 MHz OEM module. The prototype also employs two 5V PowerFilm solar panel as additional energy source. The characteristics of the two radios are summarized in Table 4.

We fabricated a power supply board for attaching both platforms to a single battery and support charging from solar energy. This board provides an additional hardware element necessary to throwboxes, a Maxim DS2770 fuel gauge chip. The fuel gauge chip gives accurate readings of the energy stored in the battery, similar to those found in laptops. This accounts for energy consumed by the platforms and radio, as well as energy produced by the solar panels. The TelosB Mote reads this value periodically and corrects the token bucket to account for the actual energy used and produced.

To support a real-world test of the throwbox, we used our DTN testbed, the UMassDieselNet [3]. The testbed normally consists of 40 buses covering an area of more than 150 square miles. However, when the experiments were performed, during a reduced summer bus schedule, only 10 buses were running on three routes. Each bus is a highly mobile DTN node using a small computer with an attached access point and WiFi interface. Buses constantly scan for other nodes and transfer DTN data whenever a connection can be made.

We augmented the equipment on the buses with an XTend radio and added scripts to beacon the position, speed, and direction of motion of the buses once each second. We deployed three always-on throwbox prototypes in fixed locations for three weeks on the UMassDieselNet bus routes (shown in Figure 5). The throwboxes were placed according to our deployment algorithm in previous work [27].

VI. Evaluation

We evaluated the throwbox system through two techniques: trace-driven simulations and prototype experimentation. The simulations use traces collected by placing three always-on throwboxes in UMassDieselNet for three weeks changing batteries manually. These prototypes logged connection events on the 802.11b radio and the XTend radio and we fed the data into a Java-based simulator. Evaluation through trace-driven simulations are necessary since buses change routes over time and comparison with other systems is unfair through prototype experimentation. For all simulations, we use the MaxProp [3] protocol to route data across nodes. The size, distribution and rate of packets generated are taken as the same as in [3].

In the simulations, we compare our system with the following systems.

- Optimal (Dual platform): The optimal system uses the two-radio, two-platform system, but with a perfect mobility prediction algorithm and an optimal scheduler. It has an oracle that knows the exact time and length of every connection opportunity. The system uses an optimal dynamic programming algorithm for the knapsack problem to select the exact set of connection events that maximizes data delivery while meeting an energy constraint [7]. This system represents the best our dual hardware design can do with an oracle of future events.
- PSM*: This system is a single tiered single radio system that periodically powers off its wireless interface to save energy. In some ways this is similar to the PSM system found in WiFi cards. PSM* wakes up its wireless interface and scans for connection events and goes back to sleep if it finds none. When the WiFi card is switched off, the platform is in its idle state. We exhaustively searched the state space to set parameters (time between wakeups and time for scanning) that use the minimum energy with equivalent data transfered. This provides a comparison to the best system that does not use extra hardware.
- WoW*: This system is adapted from Wake-on-Wireless [19] which uses a second radio as a discovery radio. The published WoW system always wakes up when it sees data addressed to it on the discovery radio and it is assumed that the discovery radio has the same range as the data transfer radio. To provide a fair comparison, we adapt WoW to a DTN environment. The WoW* system uses our mobility prediction engine to intelligently decide when to wakeup the data transfer radio. Without this modification, the WoW system would wakeup on every spurious contact over the long-range radio. However, it does not use the scheduling algorithm to decide which opportunities to take, nor does it duty-cycle the long-range radio. This is consistent with the always-on discovery radio in the published WoW paper [19].
- Always-on: This throwbox remains on all of the time, and thus does not use the second radio, and it is not penalized by the additional energy needed for the long-distance radio and Mote.
- No throwbox: Without a throwbox the DTN delivers data normally.

First, we show the statistics of the collected data to provide some insights into the rest of the results. We also compare the data collected by the throwbox to the statistics from a purely mobile DTN. Second, we show how well the mobility prediction and token bucket schedulers operate in comparison to the optimal and WoW* schemes. We show that the mobility prediction algorithm has a zero percent false-positive rate, a

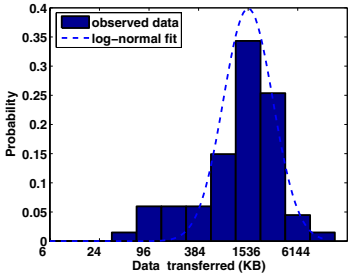

Fig. 6. The PDF of the amount of data transfered per contact opportunity with a throwbox

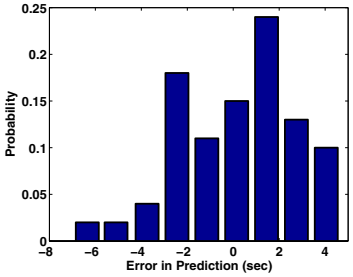

Fig. 7. The PDF of the error introduced by the Mobility Prediction Engine

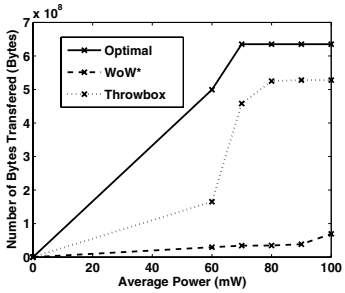

Fig. 8. The number of bytes transfered by the throwbox for different average power constraints

10% false-negative rate, and generally predicts a node's arrival within 4 seconds. While the scheduling algorithm is inferior to the optimal algorithm (1.25-competitive), it delivers significantly more packets than WoW* and meets the average power constraint. Third, we show the trade off between the network-wide throwbox benefits and its average power constraint. The results demonstrate that almost all of the benefits can be obtained with a very modest power budget of 80 mW, 3.2% of the always-on throwbox and 8% of the PSM* system. Fourth, we demonstrate that this power constraint requires small-sized solar cells that fit on our prototype's case. Lastly we show the results of deploying a real throwbox and demonstrate that small-sized solar cells can meet our goal of a perpetually operating throwbox while delivering almost as many packets as an always on system.

A. Data Statistics

Figures 6 shows the PDF of the amount of data transfered per contact. The amount of data transfered during one contact is seen to be approximately a log-normal distribution. This is similar to the distribution observed for bus-to-bus data transfer [3]. However, the mean of the data transfered between a throwbox and a bus is around two times that of a bus to bus transfer. This increase in the throughput is due to two factors. First, the throwboxes are stationary increasing the time they are in contact with a bus. Second, the throwboxes have been strategically placed to increase the delivery rate [27]. The locations are in relatively dense areas of connectivity in the DTN. The inter-contact time (result in [2]) between throwboxes and buses is seen to be a bimodal distribution.

B. Trace-Driven Simulation Results

1) Mobility Prediction Engine: We next test the accuracy of the mobility prediction engine. The traces collected from the XTend radio of the always-on throwbox prototypes are taken as an input to this experiment. The mobility prediction engine is run on this trace to predict probable connection events, their time of occurrence, and sizes of the connection events. The 1km-by-1km square around a throwbox was divided into 25 200m-by-200m cells. The prediction engine was warmed-up with only one contact opportunity from each route. Figure 7 shows the PDF of the error of the mobility prediction engine.

The results show that the prediction engine was able to predict probable connection events with an average error of less than 4 seconds. From Figure 7, it is clear that the system was woken up earlier than the actual connection events most of the time and hence it does not miss many transfer opportunities, though it consumes slightly more energy than the ideal case. Two additional results not shown in the graph are that the system never predicts false-positives (buses that it predicts will enter 802.11 range, but don't) and a 10% false negative rate (connection opportunities that it misses). The false negative rate comes from the duty-cycling algorithm used in the long-distance radio—described in Section III-C.

2) Token Bucket Scheduler: In the next experiment, we evaluate the token bucket scheduler. The experiment uses the predicted length of the connection events and the predicted time of occurrence from the mobility prediction engine. We evaluate whether the token bucket scheduler is able to meet an average power constraint while delivering a large number of packets. We have set the average power constraint to 80 mW and compare the results with a WoW* system and an optimal system. Recall that an optimal system uses the same hardware as a throwbox but has a perfect mobility prediction engine and prior knowledge of all connection events. Figure 8 shows the amount of data delivered by each of the systems, and Figure 9 depicts the depletion of battery energy with time.

The results show that the WoW* system delivers less than 20% of the amount of data delivered by the token bucket scheme. The token bucket scheduler transfers more than 80% of the amount of data transfered by the optimal scheme, and hence is approximately 1.25-competitive. Figure 9 shows that the throwbox scheduler meets the average power constraint

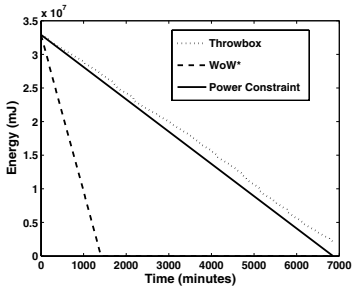

Fig. 9. How energy is used by the different schedulers as a function of time

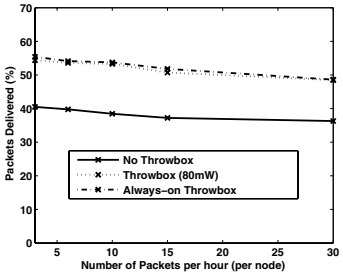

Fig. 11. The increase in delivery rate with an 80 mW power constraint

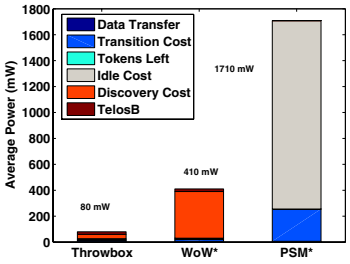

Fig. 10. The breakup of energy consumed by components

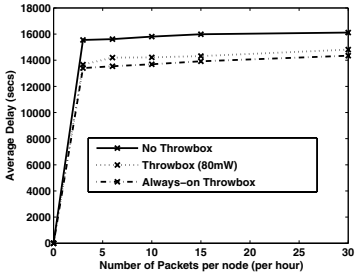

Fig. 12. The decrease in latency with an 80 mW power constraint

(shown as a straight line), while the WoW* system quickly depletes its energy. This is due to two effects. First, WoW* does not discriminate based on the size of the connection events and takes short events, providing little benefit and wasting energy. Second, the WoW* system does not duty-cycle the long distance radio, an additional benefit of our throwbox design.

To show these results in more detail, we compare the throwbox system with the WoW* as well as the PSM* system. We did not show the PSM* system in the previous graphs as it performs orders of magnitude worse than the other systems. We ran a trace-based simulation of the throwbox system with a power constraint of 80 mW. We then ran simulations on the WoW* and PSM* systems to find the amount of power used by the systems to transfer the same number of bytes. Figure 10 shows the average power consumed by these systems, broken down by how the power is spent. We have divided the bars into: energy for useful work (Data Transfer), the number of tokens left over by the scheduler at the end of the simulation (Tokens Left), the energy used by the long distance radio (Discovery Radio), the cost of turning the tier-1 system on (Transition Cost), the cost of the idling tier-1 system while it is searching for contacts (Idle Cost), and the power consumed by the tier-0, Mote system (TelosB).

The results of the PSM* system plainly illustrates the value in using the tier-0 platform and radio. PSM* devotes 99.5% of its energy turning the platform on and off and idling while searching for contacts. The WoW* system virtually eliminates the idle cost as it only turns on the tier-1 system when there is a contact present. However, it does devote a large amount of energy to the long-distance radio, as it does not use the duty-cycling employed by our throwbox design—described in Section III-C. Additionally, the WoW* system has higher costs for transitions as it amortizes that cost over smaller connection events as compared to the throwbox system.

3) Average Power Constraint Versus Network Performance Boost: We next study the performance boost provided by placing a throwbox in the UMassDieselNet [3] for a given power constraint. We use the MaxProp [3] routing protocol to transfer data among nodes in the network. Each node used a 1 GB buffer and the packets transfered were 1 KB in size. We varied the number of packets generated per hour and calculated the delivery rate and average latency of transfered packets. We ran the experiments for three weeks of simulated time. Figures 11 and 12 show the increase in packet delivery and decrease in packet delivery time when deploying a throwbox with an average power constraint of 80 mW.

We find from the results presented in the figures that the throwbox with an average power constraint of 80 mW performs as well as an always-on throwbox and leads to an increase in delivery percentage of more than 37% and decrease of at least 10% in the packet delivery time. When compared to an always-on throwbox, the power constraint of 80 mW yields a system that can last 31 times longer on the same battery while delivering almost as many packets.

C. Prototype Evaluation

We deployed a two-platform, two-radio prototype with a power constraint of 80 mW, using the mobility prediction, scheduling, and duty-cycling algorithms described in the paper. The system was equipped with solar panels 220 cm^2 in area and a 1 Ah battery that was 20% full. The box was deployed in the UMassDieselNet for a day and it logged the battery capacity remaining in the battery as a function of time. The results are

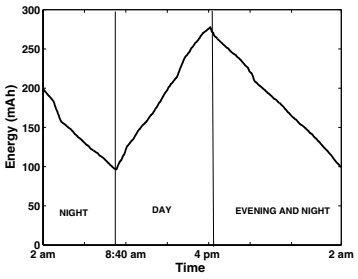

Fig. 13. The consumption of energy for the throwbox prototype over a period 24 hours

shown in Figure 13. The results show that during the day the throwbox stores excess energy and during nighttime it spends the excess accumulated energy. We determined through this experiment that the solar panels produce on an average power of 65 mW over 24 hours. This is about 15 mW less than that the amount necessary to make the throwbox last perpetually. The amount of energy produced will vary from day to day depending on the intensity of the sun. Therefore, throwboxes can be made to run perpetually through the use of a slightly larger solar cell area (270 cm^2) and accurate modeling of the solar power (e.g., using eFlux [22]).

VII. Conclusion

This paper presents a novel paradigm for power management in DTNs that provides efficient neighbor discovery and prediction of cost and opportunity of each possible contact. Through these predictions, throwboxes can select the most useful contact opportunities such that energy constraints can be met while maximizing the number of packets delivered. Our methods increase the lifetime of the throwbox by 31 times over an always-on throwbox while delivering almost as many packets. We show through extensive trace-driven simulations and prototype experimentation that a throwbox can run perpetually on solar cells slightly larger than the size of the box itself.

Acknowledgments

This material is based upon work supported by the National Science Foundation under awards CNS-0447877 and NSF-0133055, CNS-0520729, CNS-0519881, DUE-0416863. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF. We would also like to thank our anonymous reviewers for their helpful comments.

References

[1] M. Anand, E. B. Nightingale, and J. Flinn. Self-Tuning Wireless Network Power Management. In *Proc. ACM Intl Conf on Mobile Computing and Networking (MobiCom)*, 2003.

[2] N. Banerjee, M. D. Corner, and B. N. Levine. An energy efficient architecture for throwboxes in disruption tolerant networks. Technical Report 06-39, UMass Amherst, 2006.

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

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

[5] 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.

[6] S. Choi and K. G. Shin. Predictive and adaptive bandwidth reservation for hand-offs in qos-sensitive cellular networks. In *ACM Sigcomm*, 1998.

[7] T. H. Cormen, C. E. Lieserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[8] 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)*, pages 141–148, October 2001.

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

[10] P. Ferguson and G. Huston. *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*. Wiley and Sons, Inc, 1998.

[11] H. Jun, M. H. Ammar, M. D. Corner, and E. Zegura. Hierarchical Power Management in Disruption Tolerant Networks with Traffic-Aware Optimization. In *Proc. ACM SIGCOMM Workshop on Challenged Networks (CHANTS)*, September 2006.

[12] H. Jun, W. Zhao, M. Ammar, E. Zegura, and C. Lee. Trading latency for energy in densely deployed wireless ad hoc networks using message ferrying. *Elsevier Journal of Ad Hoc Networks*, 2006. To Appear.

[13] T. Liu, P. Bahl, and I. Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks. *IEEE JSAC*, 16(6):922–936, 1998.

[14] N. Mishra, K. Chebrolu, B. Raman, and A. Pathak. Wake-on-WLAN. In *Proc. Intl Conf on the World Wide Web (WWW)*, pages 761–769, 2006.

[15] T. Pering, Y. Agarwal, R. Gupta, and R. Want. CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces. In *Proc. ACM MobiSys*, pages 220–232, June 2006.

[16] T. Pering, V. Raghunathan, and R. Want. Exploiting Radio Hierarchies for Power-Efficient Wireless Device Discovery and Connection Setup. In *VLSI Design*, pages 774–779, 2005.

[17] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *ACM Sensys*, 2004.

[18] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling Ultra-Low Power Wireless Research. In *Proc. Intl Conf on Information Processing in Sensor Networks (IPSN/SPOTS)*, April 2005.

[19] E. Shih, P. Bahl, and M. J. Sinclair. Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices. In *Proc. ACM Mobicom*, pages 160–171, September 2002.

[20] W.-S. Soh and H. S. Kim. Dynamic Bandwidth Reservation in Cellular Networks Using Road Topology Based Mobility Predictions. In *Proc. IEEE Infocom*, March 2004.

[21] L. Song, U. Deshpande, U. C. Kozat, D. Kotz, and R. Jain. Predictability of WLAN Mobility and its Effects on Bandwidth Provisioning. In *Proc. IEEE INFOCOM*, 2006.

[22] 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.

[23] W. Su, S. Lee, and M. Gerla. Mobility prediction and routing in ad hoc wireless networks. *International Journal of Network Management*, 2001.

[24] A. S. Tanenbaum. *Computer Networks*. Prentice-Hall, 3rd edition, 2003.

[25] R. Want, T. Pering, G. Danneels, M. Kumar, M. Sundar, and J. Light. The Personal Server - Changing the Way We Think about Ubiquitous Computing. In *Proc. ACM UbiComp*, Sept. 2002.

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

[27] 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.