Resource and Performance Tradeoffs in Delay-Tolerant Wireless Networks[*]

Tara Small
Wireless Networks Laboratory
Center for Applied Mathematics
Cornell University
Ithaca, NY 14853, USA
tsmall@cam.cornell.edu
http://wnl.ece.cornell.edu

Zygmunt J. Haas
Wireless Networks Laboratory
Department of Electrical and Computer Engineering
Cornell University
Ithaca, NY 14853, USA
haas@ece.cornell.edu
http://wnl.ece.cornell.edu

ABSTRACT

Wireless and mobile network technologies often impose severe limitations on the availability of resources, resulting in poor and often unsatisfactory performance of the commonly used wireless networking protocols. For instance, power and memory/storage constraints of miniaturized network nodes reduce the throughput capacity and increase the network latency. Through various approaches and technological advances, researchers attempt to somehow compensate for such hardware limitations. However, this is not always necessary. Sometimes, the required performance of such networks does not need to adhere to the level of services that would be required for performance-critical applications. For example, for some applications of sensor networks, minimal latency is not a critical factor and it could be traded off for a more limited resource, such as energy or throughput. Such networks are termed *delay-tolerant* networks. Thus, to reduce the energy expenditure, transmission range of such sensor nodes would be quite short, leading to network topologies in which the average number of neighbors of the network nodes is very small. If the sensor nodes are mobile, then most of the time a node has no neighbors; only infrequently another node migrates into its neighborhood. This means that the classical networking approach of *store-and-forward* would not work well, as there is nearly never an intact path between a source and a destination. Several routing protocols have been proposed for this type of networking environment, one example is the *Shared Wireless Infostation Model (SWIM)*, where a packet propagates through the network by being copied (rather than forwarded) from a node to a node, as links are sporadically created. The goal is that one of the copies of the packet reaches the destination. SWIM is an example of the way that non-critical performance could be traded off for insufficient resources, such as the tradeoffs between energy, delay, storage, capacity, and processing complexity. In this paper, we examine some of these tradeoffs, exposing the ways in which resources could be saved by compromising on the level of performance, as to satisfy the particular limitations of network technologies.

Categories and Subject Descriptors

H.1.0 [**Models and Principles**]: *Resource Tradeoffs*

General Terms

Algorithms, Design, Performance

Keywords

Sensor networks, store and forward, carry and forward, resource tradeoffs, SWIM, Shared Wireless Infostation Model, delay-tolerant networks, epidemic routing, message ferrying, Data MULEs, sparse networks

1. INTRODUCTION

Sparse mobile wireless networks, where the average number of neighbors of a network node is small (typically less than 1), suffer from frequent, often chronic, partitions. In these networks, nodes are often isolated, unable to communicate with any other network node due to small transmission ranges of the nodes, power constraints, signal propagation impairments (such as line-of-sight limitations or propagation obstacles), and large areas over which the nodes could be located. Almost all traditional routing protocols, wired and wireless[9], assume the existence of connected paths between the message sources and their destinations during the transmission and forwarding[1] of data. In this paper, we analyze protocols that alleviate the problem of chronically

[*]This work has been supported in part by the National Science Foundation grant numbers ANI-0329905 and ANI-0081357, and the DoD Multidisciplinary University Research Initiative (MURI) program administered by the Air Force Office of Scientific Research under the grant number F49620-02-1-0233.

[1]Thus the name *store-and-forward* routing

disconnected paths by having a node store the packet, carrying it until meeting another relay node, and forwarding the packet to the other relay node. We term this routing as the *store-carry-and-forward* paradigm. A packet is considered successfully delivered, if the node carrying the packet encounters the destination node and offloads the packet to the destination node. Thus, using the mobility of the nodes themselves, the network successfully delivers packets by forwarding them along *virtual links*, which are created by the movement of network nodes.

Similar approaches were considered before in the technical literature. For example, Grossglauser and Tse [5] showed that the long-term throughput for a source-destination pair in an ad hoc network can remain constant as the density of nodes increases. Their algorithm is based on the idea that maximizing throughput involves scheduling transmissions over sufficiently good quality paths[2] at any given time. Although the message is guaranteed to be delivered in some finite average time (given infinite storage capacity at nodes), this time may be very long. Thus, [5] trades off network delay for network capacity. Shah et al. [10] introduced a three-tiered architecture to lessen the power requirements of mobile sensors that wish to send data to collection stations. The work establishes a collection of mobile entities called dataMULEs (Mobile Ubiquitous LAN Extensions), placed on creatures or devices that are already present in the environment. Their scheme uses the natural motion of the nodes and devices to create virtual links, in which the dataMULEs approach the sensors, retrieve data, and deliver the data to the collection stations. This scheme provides a low-power transport medium to recover sensor data. Message ferrying [14] uses special mobile nodes called message ferries to aid communication services, similar to dataMULEs. However, the ferries follow a non-random predetermined route (known by the nodes), rather than relying on their natural motion. This improves the latency significantly over the dataMULE case. Whenever a source node generates a packet to be delivered to an out-of-range destination node, the source transmits the message to a ferry and the ferry delivers the message.

Another approach to routing in sparse networks is the *Shared Wireless Infostation Model (SWIM)*[3] [12]. This model is based on the same *store-carry-and-forward* routing, but now each time that a node encounters another node, the packet(s) stored in its buffer are replicated (copied), rather than forwarded, to the encountered node. Thus the copies of the packet can be seen as propagating by diffusion. Now, a packet is considered successfully delivered, if any node carrying a copy of the packet encounters the destination node and offloads the packet to the destination node. Since there are typically many nodes that carry copies of the packet, the time until offloading is significantly reduced. That is, the network delay is reduced at the expense of additional storage in the network nodes. As well, the system is more reliable.[4].

Virtual links, and the associated *store-carry-and-forward* networking paradigm are quite useful in establishing communication in sparse networks. Unfortunately, it is also a fact that this strategy often leads to high latency, since nodes can carry the packets for long time while disconnected from the network. Of course, when network latency is not critical, such as is the case in *delay-tolerant* networks, the *store-carry-and-forward* paradigm can prove to be adequate. For example, this is the case when the eventual delivery of the messages is very important, possibly more important than the delay. With the *store-carry-and-forward* communication paradigm, the delays of packets depend on the rate at which virtual links are created in the network, as well as availability of network resources, such as storage space and energy. Though some studies focus on reducing the energy usage in delay-tolerant wireless networks,[5] few explicitly examine tradeoffs between different aspects of the system: delay, storage, energy, and capacity. In this work, we wish to analyze such tradeoffs.

In fact, many works exist that study the above tradeoffs in wireless ad hoc and sensor networks, but the assumption is that these networks do not undergo frequent partitions. Bansal and Liu [1] showed that routing algorithms for ad hoc networks can achieve close-to optimal capacity while also keeping the delay small. Cui et al. [3] studied the energy-delay tradeoff in sensor networks by adjusting the transmission rates of the radios at the nodes and jointly designing the routing and scheduling together. Herdtner and Chong [6] consider not only the amount that the network can carry, but also the amount of storage space that is needed for the relayed packets. We hope to leverage some of these works in our study of sparse, delay-tolerant networks.

In this paper, we specifically focus on the energy-delay and the storage-delay tradeoffs. For simplicity, we center our attention on a wireless mobile sensor network, where packets are generated at mobile nodes and a small number of destination nodes that do not change are designated sinks, which are called collection stations. We assume that the applications of our sensor network are of delay-tolerant type. We use the *SWIM* to model routing in these networks. For the remainder of this section, we describe the mathematical model of SWIM and present the methods of calculating the energy, storage, delay, and throughput of the system. Section 2 describes a method of refining the scheme to improve the storage-delay tradeoff. Section 3 introduces a different enhancement scheme to adjust the energy-delay tradeoff. Finally, Section 4 concludes the paper.

Our evaluation framework introduced in this paper allows to evaluate the performance as we vary network parameters, such as the number of nodes, without complex and time-consuming simulations.[6] However, since a complete mathematical model is rather complicated, our framework is based on a number of well justified simplifications. For example, we claim [11] that it is sufficient to find a mathematical representation of propagation of an individual packet, which can represent any (independent) packet of the network, rather than modeling propagation of many packets in the network. However, this single-packet model can represent potentially complicated mobility patterns through N

[2]Where good quality refers here to a short path, one or two hops long.

[3]Infostations refer to the particular type of collection stations. [4]

[4]Possibly, this routing paradigm should be termed *store-carry-and-replicate*

[5]For example, the dataMULEs [10] and Message Ferrying [14] studies the reduction of energy.

[6]Similarly, if evaluation of an existing network is required, our framework could be used, avoiding subjecting the network to complicated and time-consuming measurements or disrupting the network operation.

parameters that represent the contact rates between the nodes and between the nodes and the collection stations. The Markovian SWIM representation metrics match many real life metrics such as the packet delays and queue sizes in biological data acquisition shown in [11]. SWIM is an extension of the S-I-R model from epidemic routing [13].

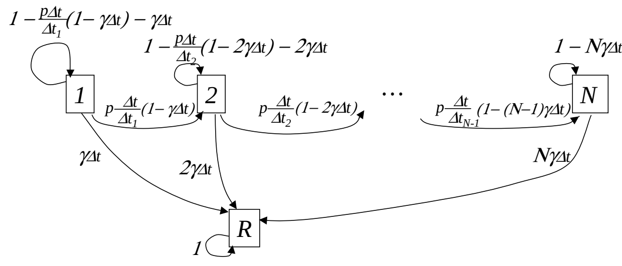
Epidemic routing is based on replication and propagation of copies of a message to many mobile nodes. In epidemic algorithms, mobile hosts forward their packets to randomly-chosen relay nodes, as well as retaining a copy of the packet. The name epidemic routing stems from the behavior of the messages, which is similar to the behavior of an infectious disease. A node carrying a packet is analogous to a carrier of an infectious disease, a carrier which we deem both infected and infective. A susceptible node does not carry the packet but has the potential to do so. An infective node replicates and forwards a message to a susceptible one, once they come into communication range due to their movement.

Although many different mobility patterns could be used, we chose the *random directional mobility*, similar to the pattern described by Bettstetter [2], on a closed (toroidal) network area. For a specified number of discrete time-steps, nodes move with a speed chosen uniformly between v_{min} and v_{max} and in a direction chosen uniformly at random. This relatively simple mobility pattern allows nodes to move anywhere in the network area. Of course, our framework is general and can be applied to any mobility pattern.

The models from epidemic theory are adapted, so that the Markov models of SWIM better and more flexibly represent our networking scenarios. As mentioned above, since modeling of a system with many interactions is complicated, we model the replication of a single packet at a time. If an arbitrarily large number of packets can be sent in any "interaction between nodes, then this method sufficiently well describes the true behavior. This is so, because the replication of one unique packet does not affect the replication of another packet in any way. The Markov chain shown in Figure 1 models the replication of one individual packet in the system.

Since it is easier to model a discrete-time system, in our framework time is quantized into very small time intervals of length Δt allowing only one forwarding event in a single Δt. Each state i represents the number of copies of the packet in the system and R represents the system state in which at least one of the copies has been offloaded to a collection station. γ is the contact rate[7] from $\underline{\text{one}}$ node to $\underline{\text{any one}}$ collection station; this parameter appears in the transitions from non-offloaded states i to the offloaded state R. If the system is in state i, then i nodes carry copies of the packet, so the rate of offloading is $i \cdot \gamma$, and the probability of offloading is $i \cdot \gamma \cdot \Delta t$.

The Δt_i variables represent the average number of time-steps that the system remains in state i (i.e., that there are i copies of the packet) before it is further replicated to another node. The transition probability for the geometric trials from state i to state $i+1$ is $\frac{\Delta t}{\Delta t_i}$. When within range of another node, the packet is copied with probability p; this accounts for the factor of p in the transitions from state i to state $i+1$. Based on the size of Δt, a single node-to-node forwarding event occurs per time-step; however, during that time-step it is possible for a copy to be offloaded. The of-



$1 - \frac{p\Delta t}{\Delta t_1}(1-\gamma\Delta t) - \gamma\Delta t$
$1 - \frac{p\Delta t}{\Delta t_2}(1-2\gamma\Delta t) - 2\gamma\Delta t$
$1 - N\gamma\Delta t$
1
2
...
N
$p\frac{\Delta t}{\Delta t_1}(1-\gamma\Delta t)$
$p\frac{\Delta t}{\Delta t_2}(1-2\gamma\Delta t)$
$p\frac{\Delta t}{\Delta t_{N-1}}(1-(N-1)\gamma\Delta t)$
$\gamma\Delta t$
$2\gamma\Delta t$
$N\gamma\Delta t$
R
1

Figure 1: Model representing transmission of a single packet in an $N-$node system

floading event is given priority in the model, since offloading is our primary goal. Therefore, the overall transition probability from state i to $i+1$ is the conditional probability that the packet is shared between nodes and not offloaded. This results in the $(1 - i\gamma\Delta t)$ factor.

The parameter p can be used to manage the utilization of resources in the system. For example, a node might have restricted power, so it chooses to send packets during one out of every three interactions. On average, each individual packet is sent in an interaction with probability $\frac{1}{3}$, so we can account for this power-saving strategy by assuming $p = \frac{1}{3}$. In this way, the power is reduced, but the delay of the packets is increased. Additionally, the factor p can account for failures in reception. If the communication channel introduces bit errors (due to radio propagation impairments or thermal noise), we would set p to be the probability of successful reception of an entire packet.

This Markov model assumes that node-to-node encounters and node-to-station encounters occur at exponentially distributed intervals and that all nodes have the same probability of reaching a relay or a destination. This is a reasonable assumption for a sensor network with mobile nodes whose motion is independent from the location of the data-collecting stations. For example, consider a network of tags on animals where the animals are foraging for food. Each node independently follows a pseudo-random pattern and is equally likely to meet another node or station in its area at any time.

Using the model in Figure 1 and appropriate initial conditions, we are able to find the time-dependent probability distribution of the different states of the system. We define the random variable T as the delay of a packet, which is defined as the time from the packet creation (time $t = 0$) until one of its copies is offloaded to a collection station (i.e., when the system enters state R). Therefore, finding the probability of the system in the state R as a function of time provides us with the cumulative distribution function for the delays of the packets, $F(T)$. The energy consumption of the system is calculated by recording the number of transitions between states in this model. The system storage at time-step T, for a unique packet and its copies is found by adding up the probabilities of the different states multiplied by the number of packet copies in those states.

From the cumulative distribution of the delay, we are able to determine the threshold probability P_{thresh} that represents the confidence with which we desire each packet to be successfully offloaded to a collection station. For example, if $P_{\text{thresh}} = 0.5$ and $F^{-1}(0.5) = 200$, then the packets need

[7]And, therefore, also the offloading rate

to remain in the system for 200 time-steps, after which time they are successfully offloaded with probability 0.5, and all the copies could (and should) then be removed from the system. This is accomplished by adding a *Time-To-Live* (TTL) field to each packet, which is decreased at each time-step. When $TTL = 0$, the packet is erased. Note that only the remaining TTL time is passed when the packet is shared between nodes; i.e., there is no need for a global clock.

Let us assume that the initial probability of state i is a_i and that the initial probability of state R is a_R. Clearly, $a_1 + a_2 + ... + a_N + a_R = 1$. Let us consider probabilities P(state, time).

In the following calculations, let $d_i = \frac{p\Delta t}{\Delta t_i}(1-i\gamma\Delta t)$. First, we calculate the probability of state 1 at any time-step j.

$$P(1, j\Delta t) = a_1(1 - d_1 - \gamma\Delta t)^j \qquad (1)$$

With $P(1, j\Delta t)$ known, we calculate the probabilities of state 2 as:

$$\begin{aligned} P(2, 0) &= a_2 \\ P(2, j\Delta t) &= P(2, (j-1)\Delta t)(1 - d_2 - 2\gamma\Delta t) \\ &\quad + d_1 P(1, (j-1)\Delta t). \qquad (2) \end{aligned}$$

Similarly, we find the probabilities of states i up to $N - 1$ to be:

$$\begin{aligned} P(i, 0) &= a_i \\ P(i, j\Delta t) &= P(i, (j-1)\Delta t)(1 - d_i - i\gamma\Delta t) \\ &\quad + d_{i-1} P(i-1, (j-1)\Delta t), \qquad (3) \end{aligned}$$

and for $i = N$ the probabilities are:

$$\begin{aligned} P(N, 0) &= a_N \\ P(N, j\Delta t) &= P(N, (j-1)\Delta t)(1 - N\gamma\Delta t) \\ &\quad + d_{N-1} P(N-1, (j-1)\Delta t). \qquad (4) \end{aligned}$$

Finally, we find the cumulative distribution of the offloading times, $F(j\Delta t) = P(R, j\Delta t)$, to be:

$$\begin{aligned} P(R, 0) &= a_R \\ P(R, j\Delta t) &= P(R, (j-1)\Delta t) \\ &\quad + \sum_{i=1}^{N} (i\gamma\Delta t) P(i, (j-1)\Delta t). \qquad (5) \end{aligned}$$

2. IMPROVING STORAGE BY INCREASING COMPLEXITY

The SWIM scheme can potentially lead to a large number of redundant packet copies in the system. We reduce the storage at the nodes by eliminating packets that have at least one copy already offloaded and thus, are no longer necessary. There are numerous methods that could accomplish packet removal in this way, but we consider five possible methods here: JUST_TTL, FULL_ERASE, IMMUNE, IMMUNE_TX, and VACCINE. These methods progressively extend and improve the performance of one another. In all of these methods, the original packet and **all** of its copies are erased by $T = F^{-1}(P_{\text{thresh}})$ time-steps after the original packet was created.

- **JUST_TTL** is the simplest method. All packets remain in the system until $T = F^{-1}(P_{\text{thresh}})$ time-steps have elapsed from the original packet creation.

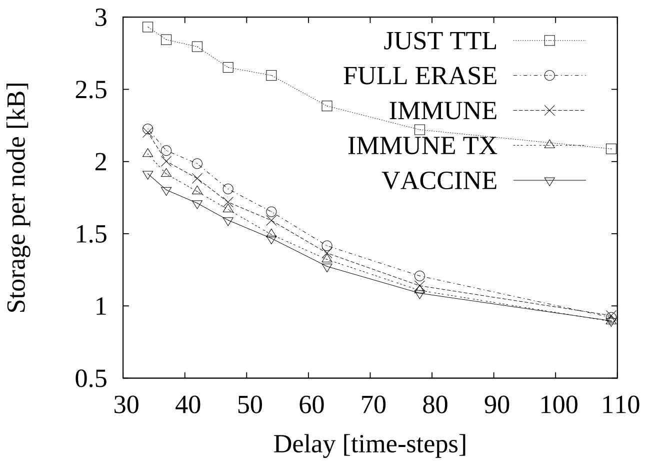


Figure 2: Storage-delay tradeoff of SWIM using the different methods of erasure

- **FULL_ERASE** erases the copy of the packet from the offloading node just after it has been offloaded to a collection station. It is reasonable to erase the packet because none of the copies are needed in the system after one of them has been offloaded to a collection station. It is, however, possible that other nodes still carry the packet copies, once it has been erased from the offloading node, so a node might receive the same packet multiple times.
- **IMMUNE** erases the packet from the offloading node when it is offloaded like FULL_ERASE, but keeps an identifier of the offloaded packet, so the node will not receive that packet again. We refer to this identifier as an "antipacket,[8]" since it prevents re-infection of the node with the packets.
- **IMMUNE_TX** erases the packet from the offloading node when it is offloaded, keeping the antipacket, like IMMUNE. It also shares this antipacket with other nodes that carry copies of the offloaded packet. This means a node may receive an identifier "antipacket" from another node only if the node stores a copy of the offloaded packet. At that time, the packet copy would be erased and the "antipacket" identifier kept.
- **VACCINE** erases the packet from the offloading node when it is offloaded, like previous methods. It also shares all packets and antipackets between nodes. In this case, a node may receive an antipacket from another node even if the receiving node does not have a copy of the packet stored.

Each of these methods corresponds to a different storage-delay tradeoff in the system. Let us fix the desired $P_{\text{thresh}} = 0.9$. Depending on the number of nodes, the number of collection stations, the mobility pattern and the probability of sending p, we obtain different values for the network delay and for the required node storage. We choose systems with one collection station, a random directional mobility pattern [8], and $p = 1$, then we vary the storage and the delay

[8]Similar to *antibody* of a biological agent

by changing the density of nodes. The higher node density induces more copying of the packets between nodes; clearly, to achieve shorter delay, one must invest more storage in the system. Assuming that the small antipacket headers are 4 bytes and that the full packets are 330 bytes (as can be true in some animal tagging applications), Figure 2 exhibits the storage-delay tradeoff due to this increased packet copying.

3. IMPROVING ENERGY BY RESTRICTING TRANSMISSIONS

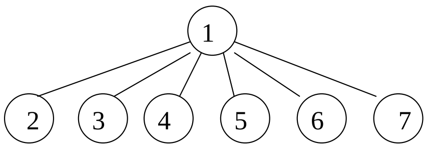
SWIM uses replication to increase the probability that a packet copy is offloaded to a collection station. Suppose that there are N_c packet copies in a SWIM system; i.e., the packet is transmitted $N_c - 1$ times between nodes. If N_c were constant over the entire lifetime of the packet, offloading would be a geometric process with mean $\frac{1}{N_c \gamma \Delta t}$, following the notation in Figure 1. Note that the difference $(\frac{1}{N_c \gamma \Delta t} - \frac{1}{(N_c+1)\gamma \Delta t})$ is smaller when N_c is larger, but the difference between N_c and $N_c + 1$ is always 1. Therefore, the reduction in average delay is smaller for the same energy expenditure as the number of copies in the system grows. We wish to improve SWIM by spending energy intelligently, so that each transmission is likely to more significantly reduce the delay.

By insisting that a packet is replicated exactly $N_c - 1$ times, we ensure that the energy spent per packet never exceeds a desired level.[9] Each of the N_c copies acts in packet replication as a subnetwork that behaves in the same manner as the original SWIM scheme. If we wanted the original system to offload a packet with probability P_{thresh}, then each of the N_c independent SWIM systems only needs to offload with probability $1 - (1 - P_{\text{thresh}})^{1/N_c}$ to provide a probability of P_{thresh} overall. Using the Markov chain of Figure 1 with $p = 0$, we can calculate $TTL_c = F^{-1}[1 - (1 - P_{\text{thresh}})^{1/N_c}] < TTL$ to achieve this desired probability that each node carrying a packet copy will cause the packet to be offload to a collection station by itself or by one of the generations of its copy.

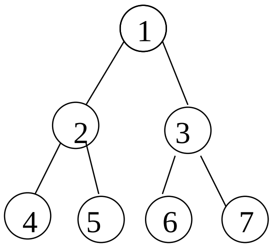
If these N_c copies of the packet could all be distributed to different nodes independently at time 0 and remain for TTL_c time-steps, then this solution would be exact. Unfortunately, such synchronized distribution cannot be practically implemented, though its performance provides a lower bound for the delay. In the SWIM scheme, the packet is generated at one node, and it must be transmitted to other nodes in the node-to-node interactions to obtain independent copies. Each time a node transmits a copy to another node, it assigns the new copy a Time-To-Live of TTL_c. In this way, we will have N_c copies in the system for at least time TTL_c each, though they are not necessarily present at the same time. We will call this process the *retransmission scheme*. In the next sections, we discuss methods to obtain these N_c copies using only local information at the nodes and analyze the energy-delay tradeoff of the retransmission scheme.

We claim that specifying the exact number of copies reduces variability in this system, reducing variability in the delays of packets as well. That is, the cumulative distribution of the delays for the retransmission scheme will be steeper than those of the original SWIM system. Furthermore, we spend energy more intelligently by transmitting only when there are small numbers of packets in the system, so we expect a lower energy-delay tradeoff curve.

[9]We assume that the lists of packet identifiers are shared between the nodes at the beginning of each interaction, so that packets are not unnecessarily transmitted.



(a) Original node sends six



(b) Binary tree with two levels

Figure 3: Possible tree structures for $N_c = 7$

3.1 Distributing N_c Copies

The delays of the packets are minimized if we copy the packet in *every* node-to-node interaction, replicating the packet as fast as possible, but stopping after $N_c - 1$ copies are present. However, this would require global synchronization in the system. An alternative is based on the observation that a node is aware of the number of hops that a packet has already traveled and the number of times that the node itself has replicated the packet. By pre-defining a tree with N_c vertices, we can ensure that exactly $N_c - 1$ copies are made.

Figure 3 shows two possible trees for $N_c = 7$: (a) the original node delivers 6 copies of the packet to other nodes that are only allowed to transmit to the collection station, (b) every node must transmit copies to exactly 2 other nodes, but the packet cannot travel more than 2 hops, unless it is offloaded to the destination. Though we assign the Time-To-Live of any packet to TTL_c when it is created, if the packet reduces its TTL to zero before the packet has been transmitted the exact number of times specified by the tree level, the packet is not erased. Such a packet copy remains in the system until it has replicated sufficient times; then it is removed.

We would like to choose the tree structure that best approximates the delay-minimized situation for each N_c. Intuitively, we realize that the 1−level tree is unlikely to be the best approximation, because all nodes except the generating node will refuse retransmission of a packet in their interactions. In a binary tree, more nodes are able to transmit packets in their interactions; however, the nodes will still refuse to transmit after two copies are made. We now show how to design a tree based on the probabilities of in-

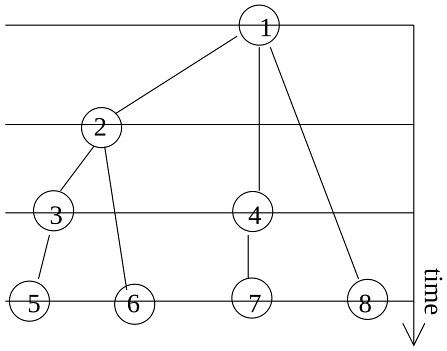


Figure 4: Locally optimal tree for $N_c = 8$

teractions between nodes, so that the smallest number of retransmissions are refused. We call this *locally-optimal* tree.

At time $t = 0$, the original node, Node 1, carries the packet and records that $N_c - 1$ replications are required. After the first replication from Node 1 to Node 2, both nodes have the same probability of meeting other nodes, so $\frac{N_c-1}{2}$ of the remaining replications are allocated to Node 1 and its future children, and $\frac{N_c-1}{2}$ are allocated to Node 2 and its future children. In a similar way, if Node 2 transmits to Node 3, then each node is given half of the remaining transmissions (i.e., $\frac{Nc-2}{4}$ transmissions) for itself and its future children.

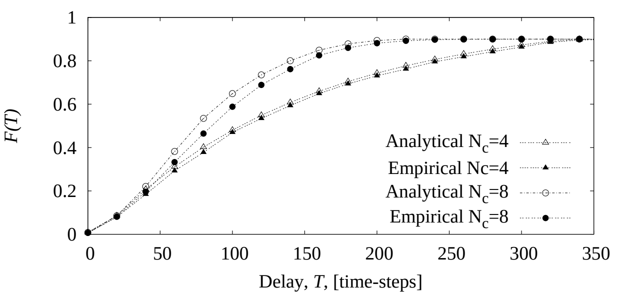
For example, Figure 4 shows a locally-optimal tree for $N_c = 8$. This tree has $\log_2(N_c) + 1$ levels, where the original node is in the 0^{th} level and 2^{i-1} nodes are in the i^{th} level $\forall i, 0 < i \leq \log_2(N_c)$. The nodes on the same level take on the average the same time to be distributed. We will say that the i^{th} *epoch* is the time between levels $i - 1$ and i.

3.2 Delays of Packets

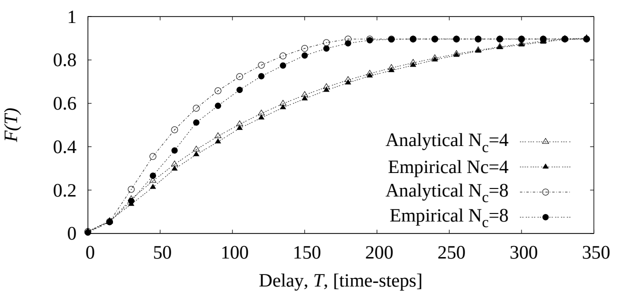
We can think of the time for offloading of a packet as the sum of two processes. First, there is the process of receiving a copy of the packet at each of the N_c nodes. Second, there is the process of offloading the packet from that node to a collection station. From the Markov model of Figure 1 with initial condition $P(1, 0) = 1$, $p = 0$ and Time-To-Live TTL_c, we already know the distribution of delays of the identical offloading phases for each node. Let us estimate the length of the reception phases R_i for each node i.

In any tree, the root node begins with a packet that it wishes to replicate throughout the tree. Given an interaction, it is assumed that a node with a packet replicates the packet to its left-most child that does not yet carry the packet. To understand the details of the procedure, we concentrate first on the single-level tree case, where only the root node is able to transmit the packet to another node.

The root node transmits to its first child following a geometric process with probability $\frac{\Delta t}{\Delta t_1}$, then two of the N total nodes carry the packet. Since there are only $N - 2$ nodes that do not carry the packet at that point, the chance for a particular data-carrying node to meet a non-data carrying node is reduced by a factor of $\frac{N-2}{N-1}$. This means that sending the packet to the root's second child follows a geometric process with probability $(\frac{N-2}{N-1}\frac{\Delta t}{\Delta t_1})$. Similarly, sending the packet to the third child follows a Geo$(\frac{N-3}{N-1}\frac{\Delta t}{\Delta t_1})$ process and so on. Therefore we can say that the mean reception phase



(a) Single-level tree



(b) Locally-optimal tree

Figure 5: Cumulative distribution of delays

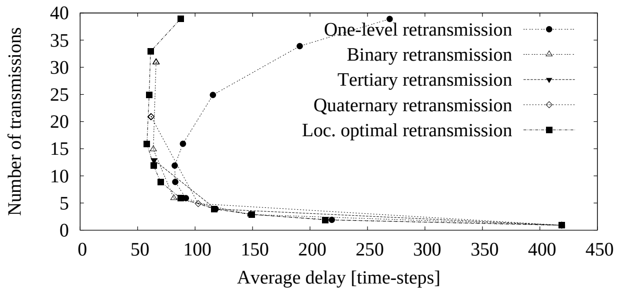
at the i^{th} node (assuming the original generating node is Node 1) has length $\mathrm{E}R_i = \Delta t_1 \sum_{j=1}^{i-1}(\frac{N-1}{N-j})$.

Using the cumulative distribution F of our model (equation (5)) to represent the offloading phase, we are able to predict the cumulative distribution for the total delay of a packet. Recall that the delay, T, is the time until the first offloading by *any* node, so:

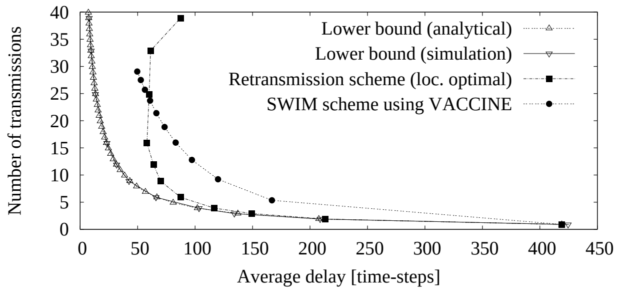
$$\begin{aligned} F(T) = \quad & 1 - P(\text{no offload from Node 1 at } T) * ... \\ & *P(\text{no offload from Node } N_c \text{ at } T) \end{aligned}$$

where $P(\text{no offload from Node } i \text{ at } T) =$
$1 - F\{T - [\Delta t_1 \sum_{j=1}^{i-1}(\frac{N-1}{N-j})]\}$. Figure 5(a) shows good agreement between the analytical and the empirical curves, where we assumed $N_c = 4$ and $N_c = 8$ in a system with 40 nodes and 1 collection station.

We use the same ideas to predict the delays for locally-optimal N_c-trees, but recalling that the structure of the trees allows some of the transmissions to occur concurrently. As before, the first epoch follows a geometric process with probability $\frac{\Delta t}{\Delta t_1}$. In the second epoch, the two data-carrying nodes wish to copy their packets to two different non-data-carrying nodes. This means that there are $N - 3$ available nodes (since two of the N nodes carry data and the two second-epoch nodes must be different) instead of the $N - 1$ nodes that were available in the first epoch. Therefore, the second epoch follows a geometric process with probability $\frac{N-3}{N-1}\frac{\Delta t}{\Delta t_1}$. Similarly, the third epoch follows a geometric process with probability $\frac{N-7}{N-1}\frac{\Delta t}{\Delta t_1}$ and so on. Therefore the cu-



(a) Using different trees



(b) Comparing retransmission and SWIM

Figure 6: Energy-delay tradeoffs with confidence level $P_{\textbf{thresh}} = 0.9$

mulative distribution of the delay is:

$$\begin{aligned} F(T) = \;& 1 - P(\text{no offload from Node 1 at } T) * ... \\ & *P(\text{no offload from Node } N_c \text{ at } T) \end{aligned}$$

where $P(\text{no offload from Node } i \text{ at } T) =$ $1 - F\{T - [\Delta t_1 \sum_{j=1}^{\lceil \log_2(i) \rceil} (\frac{N-1}{N-2^j-1})]\}$. Figure 5(b) shows that the analytical solution closely approximates the simulation result.

3.3 Energy-Delay Tradeoff Results

Let us use a system with 40 nodes and 1 collection station in a $150 * 150$ area, using random directional mobility to compare the energy-delay tradeoffs of the retransmission scheme using different distribution trees. The communication radii of both the nodes and the collection station are 7 units and we desire that 0.9 of the unique packets are offloaded. By choosing different values for N_c, we achieve different delays and different energy usage. For N_c small, we use very little energy, but it takes a long time for the packets to be offloaded with 0.9 probability. However, if we increase N_c then we have more packets in the system; there are more offloading opportunities and the delays are reduced. As shown in Figure 6(a), the tradeoffs of schemes using any of the distribution trees perform similarly for small N_c, but the behavior varies considerably as N_c grows. These longer latencies have to do with the delay in packet receptions for nodes lower in the distribution tree.

Figure 6(b) compares the energy-delay tradeoffs of the optimized retransmission scheme (with the locally-optimal distribution tree) to the original SWIM scheme. As a reference, we also show the ideal lower bound of the energy-delay curves, where the distribution of the N_c copies is instanteneous. The theoretical lower bound is calculated using the model in Figure 1 with initial conditions $P(N_c, 0) = 1$ and setting $p = 0$. The simulated lower bound distributes all N_c copies in the system uniformly at random at time $t = 0$. In this figure, the lower bound and retransmission scheme curves are obtained by adjusting N_c, while the SWIM scheme curve changes the probability p of sending a packet in a particular interaction. We see that for higher values of p and N_c, the retransmission scheme requires more time to distribute the N_c independent copies. The delays increase in this region and the original SWIM scheme outperforms the retransmission scheme.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we have examined the storage-delay and energy-delay tradeoffs in delay-tolerant wireless networks, and we have proposed a number of approaches to control the tradeoffs. The use of antipackets, small headers that are retained after a packet is offloaded to its destination, assists the network in removing obsolete, already offloaded packets. This reduces the utilized storage in the network, not only without adversely impacting the packets' delays, but in fact causing some reduction in delays. As a point of reference, by removing redundant data, the storage requirement in our example system could be reduced by more than 50%.

The proposed retransmission scheme allows us to specify the maximum energy that each packet is allowed to consume. The protocol then bounds the energy consumption to this selected maximum level and defines replication rules in the system, so that the energy is most effectively used. Therefore, by selecting different energy levels, different instantiations of the energy-delay tradeoff are realized. For low to moderate values of the energy usage, a system using the retransmission scheme achieves nearly optimal delay.

In this paper, we have exposed some aspects of the resource and performance tradeoffs in delay-tolerant wireless systems. Though we have examined these tradeoffs in the context of delay-tolerant wireless networks (mostly with sensor networks in mind) and using the Shared Wireless Infostation Model, our evaluation framework presented here can be applied to analyze and develop the resource tradeoffs of many delay-tolerant routing protocols. Our work could also provide intuition for future protocol designs. We plan on studying other tradeoffs and other techniques to control those tradeoffs. As a matter of fact, the immediate focus of our future work is the analysis and refinement of the capacity-delay tradeoff.

5. REFERENCES

[1] N. Bansal and Z. Liu, "Capacity, Delay and Mobility in Wireless Ad-Hoc Networks," *IEEE INFOCOM'03*, April 2003

[2] C, Bettstetter, "Smooth is Better than Sharp: A Random Mobility Model for Simulation of Wireless Networks," *MSWiM*, July 2001

[3] S. Cui, R. Madan, A. J. Goldsmith, and S. Lall, "Energy-delay Tradeoff for Data Collection in Sensor Networks," *ICC'05*, May 2005

[4] D. J. Goodman, J. Borràs, N.B. Mandayam, and R.D. Yates "INFOSTATIONS: A New System for Data and Messaging Services," *IEEE VTC '97* 2, 1997

[5] M. Grossglauser and D.N.C. Tse "Mobility increases the capacity of ad hoc wireless networks" *IEEE/ACM Transactions on Networking* 10, 2002

[6] J.D. Herdtner, E.K.P. Chong, "Scaling Laws in Ad Hoc Wireless Networks," `http://www.math.colostate.edu/~estep/doe_multiscale/slides/herdtner.pdf`

[7] R.R. Kompella and A.C. Snoeren, "Practical Lazy Scheduling in Wireless Sensor Networks," *ACM Sensys*, November 2003

[8] P. Nain, D. Towsley, B. Liu, and Z. Liu, "Properties of Random Direction Models," *INRIA technical report RR-5284*, July 2004

[9] C. Perkins, Ad Hoc Networking, *Addison-Wesley*, 2001

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

[11] T. Small, "Modeling Trade-offs in Networks with Intermittent Connectivity," *Cornell University PhD thesis*, August 2005

[12] T. Small and Z.J. Haas, "The Shared Wireless Infostation Model – A New Ad Hoc Networking Paradigm (or Where there is a Whale, there is a Way)," *MobiHoc '03*, June 2003

[13] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," *Technical report, Duke University*, 2000

[14] W. Zhao, M. Ammar, and E. Zegura, "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks," *MobiHoc '04*, May 2004