Practical Routing in Delay-Tolerant Networks

by

Evan P. C. Jones

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2006

Author's Declaration for Electronic Submission of a Thesis

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Delay-tolerant networks (DTNs) have the potential to connect devices and areas of the world that are under-served by traditional networks. The idea is that an end-to-end connection may never be present. To make communication possible, intermediate nodes take custody of the data being transferred and forward it as the opportunity arises. Both links and nodes may be inherently unreliable and disconnections may be long-lived. A critical challenge for DTNs is determining routes through the network without ever having an end-to-end connection.

This thesis presents a practical routing protocol that uses only observed information about the network. Previous approaches either require complete future knowledge about the connection schedules, or use many copies of each message. Instead, our protocol uses a metric that estimates the average waiting time for each potential next hop. This learned topology information is distributed using a link-state routing protocol, where the link-state packets are flooded using epidemic routing. The routing is recomputed each time connections are established, allowing messages to take advantage of unpredictable contacts. Messages are exchanged if the topology suggests that a connected node is "closer" than the current node.

Simulation results are presented, showing that the protocol provides performance similar to that of schemes that have global knowledge of the network topology, yet without requiring that knowledge. Further, it requires a significantly less resources than the epidemic alternative, suggesting that this approach scales better with the number of messages in the network.

Acknowledgements

First, I need to acknowledge the assistance and support of Lily Li, Jakub Schmidtke, and Allen George. This work started as a course project that Lily and I worked on together. Her assistance was invaluable for that work. Jakub put an amazing amount of energy into rewriting the core of the simulator used for the experimental evaluation in this thesis. The experiments went much smoother with his help. Allen did all the detective work to figure out how to extract the data we needed from the Seattle bus schedule database.

Second, I would like to thank Professors Catherine Rosenberg and Keshav for agreeing to read my thesis. I also need to thank Professor Keshav for introducing me to the area of delay-tolerant networks. I would never have considered this topic for my research if it was not for him.

Next, I would like to thank my supervisor, Professor Paul Ward, for his unending encouragement and enthusiasm. There were many times that I was discouraged by a negative result, or by discovering that someone else had already investigated a particular idea. He was always able to find a positive way to look at the results, or another aspect to investigate. I greatly appreciate the many hours of discussions we have had over the past three years.

I need to thank my friends for keeping me reasonably sane and providing me with wonderful distractions, whether I needed them or not. My time in Waterloo has been so fantastic in large part because of them.

I thank my parents for encouraging me to always chase my dreams.

Finally, I thank Eran, for putting up with me when I am distracted, and for filling my life with so much love and joy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wired and wireless networks have enabled a wide range of devices to be interconnected over vast distances. For example, today it is possible to connect from a cell phone to millions of powerful servers around the world. As successful as these networks have been, they still cannot reach everywhere, and for some applications their cost is prohibitive. The reason for these limitations is that current networking technology relies on a set of fundamental assumptions that are not true in all environments. The first and most important assumption is that an end-to-end connection exists from the source to the destination, possibly via multiple intermediaries. This assumption can be easily violated due to mobility, power saving, or unreliable networks. For example, if a wireless device is out of range of the network (*e.g.* the nearest cell tower, 802.11 base station, *etc.*), it cannot use any application that requires network communication. Delay-tolerant networking (DTN) is an attempt to extend the reach of networks. It promises to enable communication between "challenged" networks, which includes deep space networks, sensor networks, mobile ad-hoc networks, and low-cost networks. The core idea is that communication can be enabled between these networks if protocols are designed to accommodate disconnection [9].

As an example of where these networks are useful, consider a classroom where each student has a laptop, but there is no network infrastructure. One would like the students to collaborate on projects using the wireless network cards in the laptops, and also to communicate with the Internet. Delay-tolerant networking can make this happen, as illustrated in Figure 1.1. The laptops communicate with each other to exchange data. If the destination laptop is not present, which may occur if the student has gone home, the network stores the messages until they return. To communicate with the Internet, the school could be serviced via a router attached to a bus traveling between the school and an Internet gateway. This device picks up

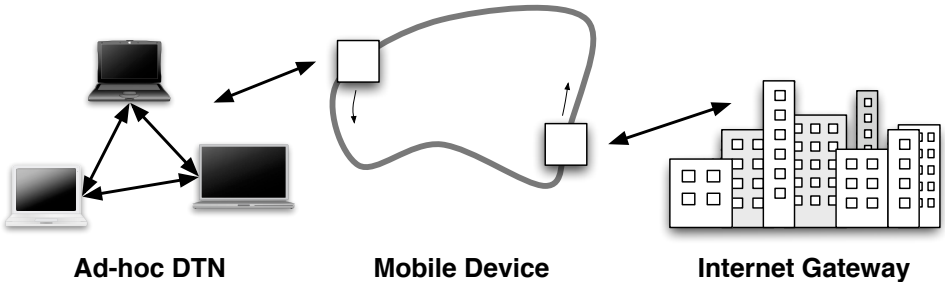


Figure 1.1: Laptops communicating with each other and the Internet via delay-tolerant networking

requests from the school and delivers them to the gateway, and then provides the responses on its next trip. First Mile Solutions sells a system called DakNet that is based on this idea [17], while the Wizzy Digital Courier Project uses a simple one-hop delay-tolerant network to provide Internet access to rural South African schools [37].

There are many other applications for delay-tolerant networks. In developing regions, applications range from education to health care to government services [2]. In developed nations, researchers have proposed augmenting low bandwidth Internet connections with a high-bandwidth delay-tolerant network built by sending physical media, such as DVDs, through the postal system [48]. This allows very large files to be quickly and cheaply exchanged while small files and control messages are exchanged over a low bandwidth link. Others have investigated using DTNs to provide Internet access to cars, by connecting temporarily to roadside wireless base stations [35]. DTNs could also be used to gather data from everything ranging from sensors in oceans [40], to satellites in space [4].

1.1 The Routing Problem

One of the fundamental problems that arises when designing networks that handle disconnection is routing. Before a network can be usable, it must be possible to get data from the source to the destination. Simple DTN-like networks have been built using static routing [17, 37], which is an effective approach for small networks. However, the benefit will increase if the networks can be scaled to service larger areas. To achieve this goal, routing protocols are needed to automate the configuration, and to cope with changes and failures.

This thesis presents a routing protocol designed to be easy to deploy. To do

so, it must meet three design goals. First, the routing must be self-configuring. This is critical for equipment that may be deployed far from network experts, and to maintain communication capability even when some components fail. Many application domains where DTNs can provide significant benefits have both of those problems. Second, the protocol must provide acceptable performance over a wide variety of connectivity patterns. This implies that the protocol will be a good choice for most DTN scenarios. This eliminates the need to perform any analysis to determine which protocol to use. Finally, the protocol must make efficient use of buffer and network resources. If the DTN becomes a valuable resource, it will be used frequently by a large number of users. Thus, it must be capable of scaling with the demand.

1.2 Contributions

This thesis presents a technique for shortest path routing in delay-tolerant networks. The main contributions of this work are as follows:

1. A routing protocol that can be deployed in delay-tolerant networks that does not require configuration. This provides a useful and efficient tool for building DTNs.

2. A metric for estimating the cost of available paths that relies strictly on observed information. This metric is sufficiently generic to be used by any routing protocol.

3. Techniques for improving the performance of shortest path routing protocols. Our results show that making routing decisions as late as possible provides resilience to resource shortfalls, and to path selection errors. We also show that hop-by-hop flow control can improve the performance when there is limited buffer space.

4. Strategies for injecting multiple copies of each message into the network, along disjoint paths, in order to increase the delivery ratio and decrease delay. We present techniques that use replication and retransmissions. These techniques provide a tunable resource consumption/performance trade-off.

1.3 Thesis Organization

In Chapter 2 we describe our model of DTNs, the important metrics used to evaluate performance in these networks, and the previous approaches to solving the routing

problem. We describe the design and operation of our practical routing protocol in Chapter 3, and evaluate its performance via extensive simulations in Chapter 4. Chapter 5 discusses techniques for further improving the performance of the basic protocol. Finally, our conclusions and future work are presented in Chapter 6.

Chapter 2

Background

Since routing is a fundamental problem that must be solved before a network can be used, there has been extensive research on this issue in DTNs. The research dates back to before the term "delay-tolerant" was widely used. The adjectives "intermittently-connected," "disruption-tolerant", "sparse," and "disconnected" are also used to describe networks without constant end-to-end connections. This chapter presents two properties that can be used to classify delay-tolerant routing strategies: replication and knowledge. Replication describes how a routing strategy relies on multiple copies of each message, and knowledge describes how information about the network is used to make decisions.

Before discussing the details of DTN routing, we first discuss the important properties of DTNs and the metrics used to evaluate each routing technique. Next, we present a system for classifying each strategy based on the two properties. We divide the routing strategies into two broad families, flooding and forwarding, based on their use of replication and knowledge. We analyze each family, and show how prior research fits into the classification scheme.

2.1 Network Characteristics

In order to discuss the routing problem, we need a model that describes the network. A DTN is composed of computing systems participating in the network, called nodes. One-way links connect some nodes together. These links may go up and down over time, due to mobility, failures, or other events. When the link is up, the source node has an opportunity to send data to the other end. In the DTN literature this opportunity is called a contact [9]. More than one contact may be available between a given pair of nodes. For example, a node might have both a

high-performance, expensive connection and a low-performance, cheap connection that are available simultaneously for communication with the same destination. The contact schedule is the set of times when the contact will be available. In graph theory, this model is a time-varying multigraph. The DTN architecture proposes to use this network by forwarding complete messages over each hop. These messages will be buffered at each intermediate node, potentially on non-volatile storage. This enables messages to wait until the next hop is available, which may be a long period of time.

As described by Jain *et al.*, the amount of time for a message to be transferred from one node to another can be divided into four components: waiting time, queuing time, transmission delay, and propagation delay [21]. The waiting time is the amount of time a message must wait between when it arrives at a node and when the contact to the next node becomes available. This depends on the contact schedule and the message arrival time. The queuing time is the time it takes to drain the queue of higher priority messages. This depends on the contact data rate and the competing traffic in the network. The transmission delay is the time it takes for all the bits of the message to be transmitted, which can be computed from the contact's data rate and the message length. The propagation delay is the time it takes a bit to propagate across the connection, which depends on the link technology.

2.1.1 Challenges

Delay-tolerant networks present many challenges that are not present in traditional networks. Many stem from the need to deal with disconnections, which directly impacts routing and forwarding. However, because these networks enable communication between a wide range of devices, there are secondary problems that routing strategies may need to be aware of, such as dealing with limited resources.

Contact Schedules

Of the four inter-node delay components, the most significant is likely to be the waiting time, since it might range anywhere from seconds to days whereas the others are typically much shorter. Thus, one of the most important characteristics of a DTN is the contact schedule, which depends strongly on the application area under consideration. Contact schedules can be placed on an approximate spectrum based on how predictable they are, as shown in Figure 2.1. At one extreme we have contact schedules that are very precise. An example would be deep space networks, where disconnections are caused by movements of objects in space that

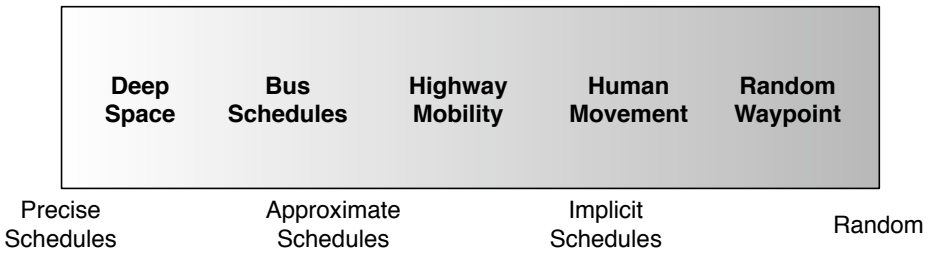


Figure 2.1: A spectrum of contact schedule predictability

can be calculated very accurately. One step less predictable would be scheduled networks with errors. For example, consider a DTN where the nodes are mounted on city buses. These buses have a schedule, but it is not precise. Due to traffic, equipment failures, or accidents, the actual arrival times can vary significantly, as anyone who regularly takes public transit can attest. Many human activities have implicit schedules, such as work. There is no guarantee when a person will be at work, but their schedule is fairly regular. Finally, at the other end of the spectrum are networks with completely random connectivity. These networks are widely studied in the ad-hoc networking community because the models are simple to work with. This spectrum is similar to the one presented in [30].

Some work on DTN routing has investigated networks with proactive mobility, where the nodes actively move in response to communication needs [30, 28, 51, 52]. In our model, this can be represented as contacts that can be selectively brought up or down. Networks with this type of mobility fall somewhere in the middle of the spectrum because some contacts are unpredictable, but the controlled contacts are predictable. The routing techniques discussed in this paper are equally applicable to networks with proactive mobility. However, these networks also require some type of cost/benefit optimization to make decisions about proactive movement, which is outside the scope of this chapter.

Contact Capacity

A question that is closely related to the contact schedule is how much data that can be exchanged between two nodes. This depends on both the link technology and the duration of the contact. Even if the duration is precisely known, it may not be possible to predict the capacity due to fluctuations in the data rate. At a first glance, it might appear that this is a simple issue for routing strategies to deal with. A naive approach would be to ignore the contact capacity, except in cases where the message is simply too large to be sent across the contact without

fragmentation. If the volume of traffic is very small compared to the capacity of contacts in the network, then this is a reasonable approach. However, if the volume of traffic increases due to a large number of users, or due to large messages being exchanged, the contact capacity becomes very important. In this situation, the best contact could become one that is "inefficient" according to other criteria, but has the largest contact volume and thus is best equipped to handle large traffic demands.

Although there are studies of real world contact duration and capacity [3, 18], few of the routing strategies surveyed attempt to use this information. One exception is the EDLQ and EDAQ schemes proposed by Jain *et al.*, which compute the delay caused by waiting for competing traffic, then route messages on the paths with the smallest delay [21].

Buffer Space

In order to cope with long disconnections, messages must be buffered for long periods of time. This means that intermediate routers require enough buffer space to store all the messages that are waiting for future communication opportunities. From one point of view, this means that intermediate routers require buffer space proportional to demand. An alternate point of view is that routing strategies might need to consider the available buffer space when making decisions. In the studies surveyed here, all nodes have an equal amount of buffer space and the strategies do not make decisions based on this resource.

Processing Power

One of the goals of delay-tolerant networking is to connect devices that are not served by traditional networks. These devices may be very small, and similarly have small processing capability, in terms of CPU and memory. These nodes will not be capable of running complex routing protocols. The strategies presented in this paper are not designed for extremely small sensors. However, research in routing for wireless sensor networks has extensively investigated this issue [1]. The routing strategies presented here could still be used on more powerful gateway nodes, in order to connect the sensor network to a general purpose delay-tolerant network.

Energy

Some nodes in delay-tolerant networks may have limited energy supplies either because they are mobile, or because they are in a location that cannot easily be

connected to the power grid. Routing consumes energy by sending, receiving and storing messages, and by performing computation. Hence, routing strategies that send fewer bytes and perform less computation will be more energy efficient. Additionally, routing strategies can optimize power consumption by using energy-limited nodes sparingly. While researchers have investigated general techniques for saving power in delay-tolerant networks [24], none of the routing strategies surveyed has incorporated power-aware optimizations. Thus, we will not discuss this topic further.

2.1.2 Evaluation Criteria

In order to compare routing strategies, we must define some metrics for evaluating their performance. Since the exact numbers for the metrics depend on many factors, we will only discuss them in relative terms.

Delivery Ratio

In a delay-tolerant network, the most important network performance metric is the delivery ratio. However, in DTNs, a message is rarely actually "lost." Rather, the network was unable to deliver messages within an acceptable amount of time. Thus, we define the delivery ratio as the fraction of generated messages that are correctly delivered to the final destination within a given time period.

Latency

A secondary metric is the latency, the time between when a message is generated and when it is received. This metric is important since many applications can benefit from a short delivery latency, even though they will tolerate long waits. Many applications also have some time window where the data is useful. For example, if a DTN is used to deliver e-mail to a mobile user, the messages must be delivered before the user moves out of the network.

Transmissions

Some routing strategies transmit more messages than others, either because they use multiple copies of each message, make different decisions about the next hop, or because of protocol overhead. The number of transmissions is a measure of the amount of contact capacity consumed by a protocol. It is also an approximate measure of the computational resources required, as there is some processing required

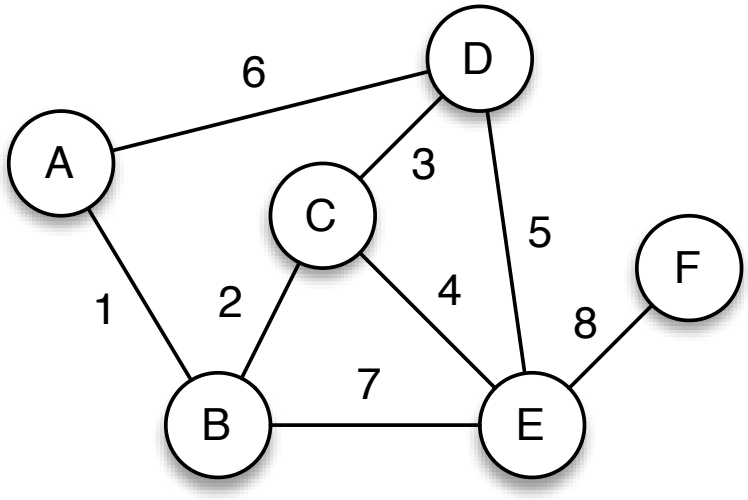


Figure 2.2: Example DTN scenario

for each message. Additionally, each transmission consumes energy, so it is also an approximate measure of power consumption.

2.1.3 Example

We will use the scenario shown in Figure 2.2 throughout this chapter to illustrate the routing strategies. There are six nodes labeled A through F. The contacts go up and down one at a time, in the sequence shown by the labels on each line. All links are bidirectional, and messages are always generated by node A.

2.2 Strategy Properties

We categorize delay-tolerant routing strategies using two properties. The first property, replication, denotes how the strategy uses multiple copies of a message, and how it chooses to make those copies. The second property, knowledge, indicates how the strategy uses information about the state of the network in order to make routing decisions, and also how it obtains that information.

2.2.1 Replication

Delay-tolerant networks may rely on components that are unreliable or unpredictable. To compensate for this, many routing strategies make multiple copies

of each message, in order to increase the chance that at least one copy will be delivered, or to reduce delivery latency. The intuition is that having more copies of the message increases the probability that one of them will find its way to the destination, and decreases the average time for one to be delivered. This is a clear trade-off between cost and performance. The cheapest approach is to have a single copy of the message. However, a single failure will result in the message being lost. The most reliable approach is to have each node carry a copy of the message. In this case, the message is lost only if all the nodes in carrying it are unable to deliver it. However, this consumes bandwidth and storage resources proportional to the number of nodes in the network.

A related issue is characterizing the best approach to making replicas. Jain *et al.* present a theoretical approach to determine which set of paths to use, provided that the path failure probabilities are known and independent [20]. Erasure coding and networking coding schemes have also been investigated to attempt to keep the benefit of multiple copies while reducing the resource costs [49, 50]. These techniques appear promising, and it should be possible to integrate them with the routing strategies presented here.

2.2.2 Knowledge

Some routing strategies require more information about the network than others. At one extreme, a node can make decisions with zero knowledge about the network, except which contacts are currently available. These strategies use static rules that are configured when the strategy is designed, and every node obeys the same rules. This leads to simple implementations that require minimal configuration and control messages, since all the rules are hard-coded ahead of time. The disadvantage is that the strategy cannot adapt to different networks or conditions, so it may not make optimal decisions. At the other end of the spectrum, a node might need to know the complete future schedule of every contact in the network. Provided that the information is accurate, this allows routing strategies to make very efficient use of network resources by forwarding a message along the best path. There is a range of values in between these two extremes. For example, approximate information about the future contact schedules might be available. Or, a strategy might require no information in advance, but instead will learn it automatically.

2.3 Strategy Families

We divide DTN routing strategies into two families based on which property a strategy uses in order to find the destination. Like all classification schemes, there are some cases that do not fall cleanly into either group, but we attempt to select the primary technique that a strategy uses. The families are flooding strategies, which rely primarily on replicating messages to enough nodes so the destination receives it, and forwarding strategies, which rely on knowledge about the network to select the best path to the destination. We first describe each family in general, and then describe specific examples in each family.

2.3.1 Flooding Strategies

Strategies in the flooding family deliver multiple copies of each message to a set of nodes, called relays. The relays store the messages until they connect with the destination, at which point the message is delivered. The earliest work in the area of DTN routing fall into this family. Many of them date before the term "delay-tolerant" became popular. Traditionally, these strategies have been studied in the context of mobile ad-hoc networks, where random mobility has a good chance of bringing the source into contact with the destination. Message replication is then used to increase the probability that the message gets delivered. The basic protocols in this family do not need any information about the network, however more advanced schemes use some knowledge to improve performance.

Direct Contact

This strategy waits until the source comes into contact with the destination before forwarding the data. This is the degenerate case of the flooding family, where the set of relays contains only the destination. It can also be considered a degenerate case of the forwarding family, where it always selects the direct path between the source and the destination. However, since this strategy does not require any information about the network and only uses a single hop, we will consider it to be a flooding strategy. Due to its simplicity, it does not consume many resources, and it uses exactly one message transmission. However, it only works if the source contacts the destination. The Infostation architecture proposed using direct contact delivery between mobile nodes and fixed gateways as a technique for increasing wireless network throughput and decreasing cost [11]. Grossglauser and Tse showed that in their mobile ad-hoc network scenario, this strategy has a capacity that approaches zero as the number of nodes increases [13].

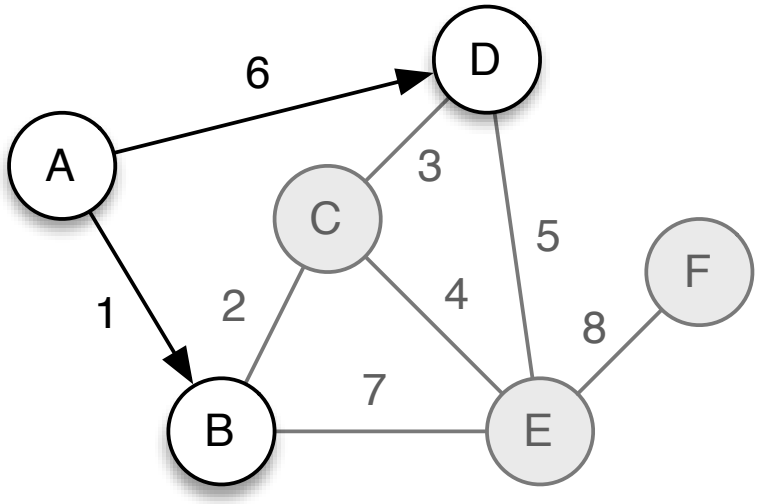


Figure 2.3: Direct Contact routing example

In the example scenario, node A can only deliver messages to nodes B and D, as shown in Figure 2.3. Additionally, it is faster for node A to deliver a message to node D via the path A-B-C-D, which it cannot do.

Two-Hop Relay

In this strategy, the source copies the message to the first n nodes that it contacts. The source and the relays hold the message and deliver it to the destination. Since there are now $n + 1$ copies of the message in the network, more bandwidth and storage are consumed. However, the resource consumption is limited and can be tuned by adjusting the number of copies. This strategy has a much better chance of delivering the message than the Direct Contact strategy. If we assume that each node contacts the destination with an independent probability p, then this strategy will deliver each message with probability $1 - (1 - p)^{(n+1)}$, which is approximately $(n + 1)p$ if p is very small. Similarly, increasing the number of copies decreases the average latency, since the message is delivered as soon as any of the $n + 1$ nodes contacts the destination. This strategy has the same fundamental limitation as Direct Contact: If the $n+1$ nodes never reach the destination, the message cannot be delivered. In scenarios where the mobility is random, this might be rare, but in networks with structured connectivity this could be very common.

In the example, if node A has a message for node E, it would send copies to both nodes B and D, as shown in Figure 2.4. When node B connects with node

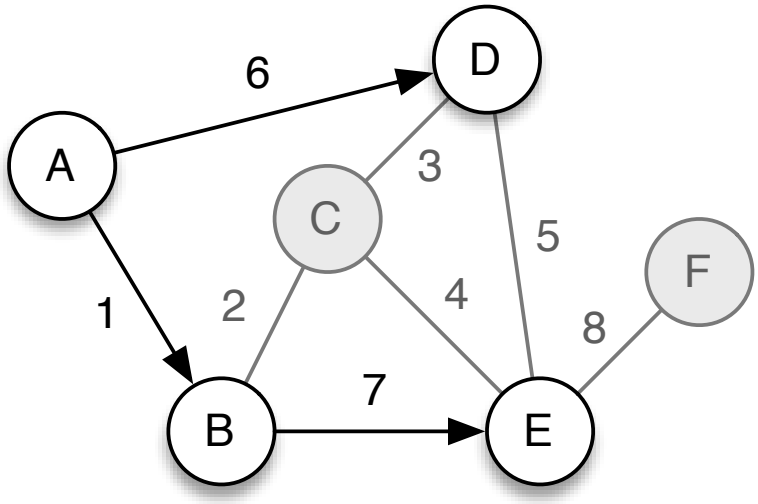


Figure 2.4: Two-hop relay example

C, it would not send it the message, since node C is not the destination. Finally, the message would be delivered at time 7 when node B connects with node E. At the end, nodes A, B, D and E have all received the message. Node A can reach all other nodes via two-hop relay except node F, since it is a minimum of three hops away.

Grossglauser and Tse showed that this strategy can be used to increase the capacity of mobile ad-hoc networks under ideal conditions [13]. This approach has also been studied as a routing strategy for sensor networks [39], and for scenarios with proactive mobility [51]. It has been proposed as a fall-back when ad-hoc routing cannot find a connected path [33].

Tree-Based Flooding

Tree-Based Flooding strategies extend two-hop relay by distributing the task of making copies to other nodes. When a message is copied to a relay, there an indication of how many copies the relay should make. This is called Tree-Based Flooding because the set of relays forms a tree of nodes rooted at the source. Two-hop relay can be viewed as Tree-Based Flooding with a depth of one.

There are many ways to decide how to make copies. A simple scheme is to allow each node to make unlimited copies, but to restrict the message to travel a maximum of n hops from the source [47]. This limits the depth of the tree, but places no limit on its breadth. A refinement is to also limit the node to make at

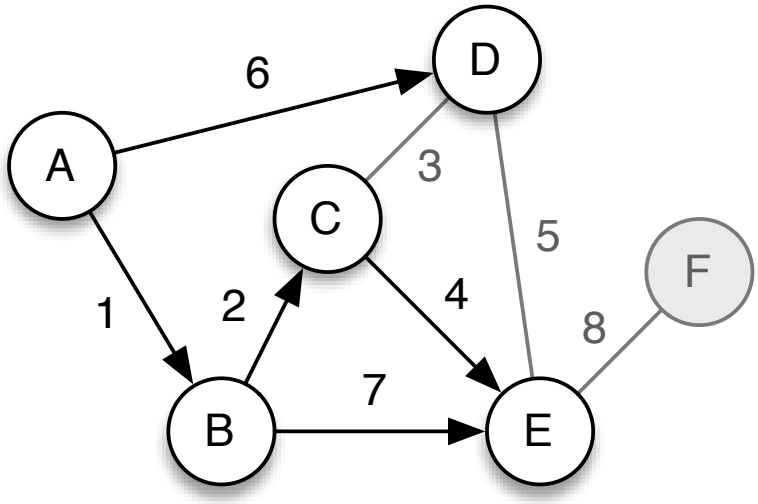


Figure 2.5: Tree-Based Flooding Example

most m copies [41]. This limits both the depth and the breadth of the tree, which limits the total number of copies to a maximum of $\sum_{i=0}^{n} m^n$. A more complex alternative is to limit the total number of copies to N. When a node makes a copy, it distributes the responsibility for making half of its current copies to the other node, and keeps half for itself [41, 43]. This scheme has been shown to be optimal if the inter-node contact probabilities are independent and identically distributed [43].

Tree-Based Flooding can deliver messages to destinations that are multiple hops away, unlike Direct Contact or Two-Hop Relay. However, tuning the parameters can be a challenge. If they are too conservative, many extra copies will be made. Conversely, if they are too aggressive, then the message may not propagate to the destination. Consider the example scenario if node A has a message for node E, and it can make a maximum of four additional copies. At time 1, it sends a copy to node B along with directions to make one copy ($\lfloor (4-1)/2 \rfloor$), shown in Figure 2.5. Node A keeps two copies for itself ($\lceil (4-1)/2 \rceil$). At time 2 when node B connects with node C, B's additional copy is delivered. At time 3, node C connects to node D. However, it cannot send D a copy because it has no copies to distribute. At time 4, C delivers the message to the E. At time 6, A sends D a copy, since it does not know that the message was already delivered. At this point, node A has one remaining copy that it will deliver if it contacts another node.

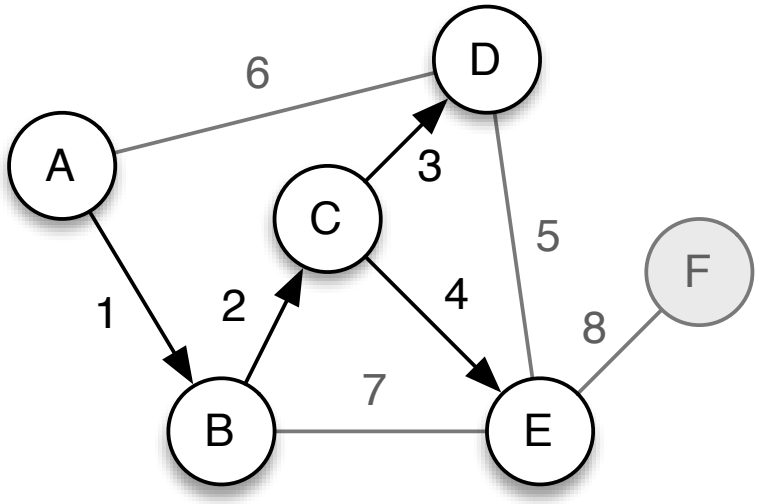


Figure 2.6: Epidemic Routing Example

Epidemic Routing

Epidemic algorithms were originally proposed for synchronizing replicated databases [8]. Vahdat and Becker applied these algorithms to forwarding data in a DTN [47]. In effect, the queue of messages waiting to be delivered is the database that needs to be synchronized. Epidemic algorithms guarantee that provided a sufficient number of random exchanges of data, all nodes will eventually receive all messages. Thus, the destination node is guaranteed to have received the data. Epidemic Routing works as follows. When a message is sent, it is placed in the local buffer and tagged with a unique ID. When two nodes connect, they send each other the list of all the messages IDs they have in their buffers, called the summary vector. Using the summary vector, the nodes exchange the messages they do not have. When this operation completes, the nodes have the same messages in their buffers.

Epidemic Routing represents the extreme end of the flooding family because it tries to send each message over all paths in the network. This provides a large amount of redundancy since all nodes receive every message, making this strategy extremely robust to node and network failures. Additionally, since it tries every path, it delivers each message in the minimum amount of time if there are sufficient resources. In the example, the message will be delivered from A to E via the fastest path (A-B-C-E), as shown in Figure 2.6. All nodes will receive the message except node F because node E does not replicate messages that are destined for itself.

Epidemic Routing is relatively simple because it requires no knowledge about

the network. For that reason, it has been proposed to use it as a fall-back when no better method is available [14]. The disadvantage is that a huge amount of resources are consumed due to the large number of copies. This requires large amount of buffer space, bandwidth, and power.

Many papers have studied ways to make Epidemic Routing consume fewer resources [30, 3, 41, 7, 29, 44, 16, 15]. One of the problems is that the message continues to propagate through the network, even after it has been delivered. The original epidemic algorithms paper proposed "death certificates" to solve this problem [8]. The idea is that a new message is propagated informing nodes to delete the original message and to not request it again. Ideally, the death certificate will be much smaller than the original message, so overall the resource consumption is reduced. Researchers have explored various schemes for tuning how aggressively the death certificates are propagated. Small and Haas show that the more aggressive the death certificate propagation, the less storage is required at each node [41], while Harras and Almeroth show that the more aggressive strategies transmit more messages [15].

A critical resource in epidemic routing is the buffer. An intelligent buffer management scheme can improve the delivery ratio over the simple FIFO scheme [7]. The best buffer policy evaluated is to drop packets that are the least likely to be delivered based on previous history. If node A has met B frequently, and B has met C frequently, then A is likely to deliver messages to C through B. Similar metrics are used in a number of epidemic protocol variants [30, 3, 7, 29, 44]. This approach takes advantage of physical locality and the fact that movement is not completely random. While these protocols are more efficient than the original Epidemic routing protocol, they still transmit many copies of each message.

2.3.2 Forwarding Strategies

The strategies in this family take a more traditional approach to routing data in a DTN. They use network topology information to select the best path, and the message is then forwarded from node to node along this path. A path can be found using location-based routing, assigning metrics to nodes or by assigning metrics to links. Some of these approaches have been explored in wired and multi-hop wireless networks. However, the protocols designed for these environments will not function in delay-tolerant networks, since they assume that links are usually connected. By definition, the strategies in this family require some knowledge about the network. They typically send a single message along the best path, so they do not use replication.

Location-Based Routing

The forwarding approach that requires the least information about the network is to assign coordinates to each node. A distance function is used to estimate the cost of delivering messages from one node to another. The coordinates can have physical meaning, such as GPS coordinates, as has been studied for mobile ad-hoc networks [31]. Alternatively, the coordinates can have meaning in the network topology space, instead of physical space, which has been used to estimate network latency between arbitrary nodes on the Internet [34]. In general, a message is forwarded to a potential next hop if that node is closer in the coordinate space than the current custodian.

The advantage of location-based routing is that it requires very little information about the network, eliminating the need for routing tables and reducing the control overhead. In order to determine the best path, a node only needs to know its own coordinates, the coordinates of destination, and the coordinates of the potential next hops. Given these three pieces of information, a node can easily compute the distance function and determine where the message should be sent.

Location-based routing has two well known problems. The first problem is that even if the distance between two nodes is small, there is no guarantee that they will be able to communicate. In the case of physical coordinates, consider two wireless nodes on opposite sides of a wall that blocks all radio signals, as shown in Figure 2.7. Node A wants to send a message to D. It has two potential next hops: nodes B and C. Since node B is the closest to node D, it forwards the message to B. However, B cannot communicate with D due to the obstruction, whereas node C has a line of sight. The problem is that location does not necessarily correspond to network topology. This problem is somewhat alleviated by using virtual coordinates, since they are designed to closely represent the network topology. However, it is still possible that the message can fall into a local minimum and not reach the destination. In mobile ad-hoc networks, protocols have been explored which attempt to route around obstructions like this [25]. The second problem is that a node's coordinates can change. If a node moves, its physical coordinates change. If the network topology changes, a node's virtual coordinates change. This complicates routing because the source needs the coordinates of the destination node. These two problems mean that implementing location-based routing is not as simple as it appears.

Lebrun *et al.* proposed using the motion vector of mobile nodes to predict their future location. Their scheme passes messages to nodes that are moving closer to the destination [26], which results in a better delivery ratio than two-hop relay with less overhead than Epidemic routing. Leguay *et al.* presented a virtual coordinate

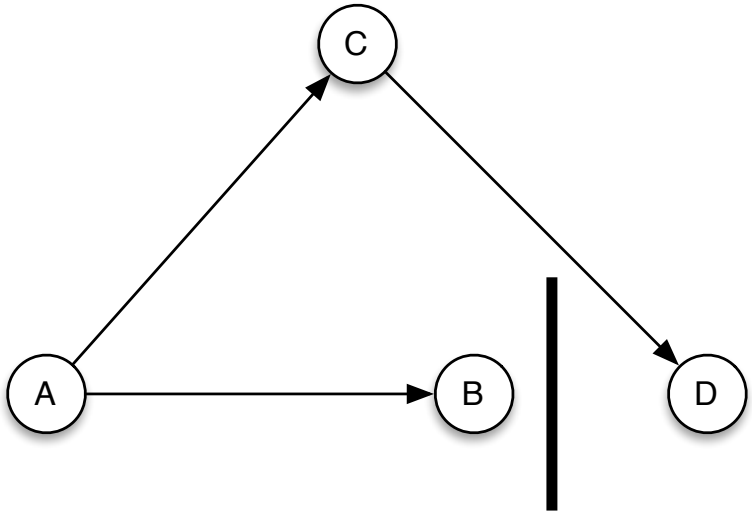


Figure 2.7: Location-based routing fails because of a local minimum

routing strategy called mobility pattern spaces [27]. In their strategy, the node coordinates are composed of a set of probabilities, each representing the chance that a node will be found in a specific location. Various distance functions are then computed on this vector. Their results show that this approach reduces the amount of resources consumed when compared with Epidemic Routing, while still delivering a substantial fraction of the messages. Neither of these works address the local minima or changing coordinates problems. However, they do show that these techniques are applicable to DTNs.

Gradient Routing

An alternate approach is to assign a weight to each node that represents its suitability to deliver messages to a given destination. When the custodian of a message contacts another node that has a better metric for the message's destination, it passes the message to it. This approach is called gradient routing because the message follows a gradient of improving utility function values to-wards the destination. The idea was first applied to ad-hoc networks in 2000 [36]. This requires more network knowledge than location-based routing for two reasons. First, each node must store a metric for all potential destinations. Second, sufficient information must be propagated through the network to allow each node to compute its metric for all destinations. The metric could be based on many parameters, such as the time of last contact between the node and the destination, remaining battery

energy, or mobility. An extremely simple utility function is to forward a packet with a certain probability. This approach does not seem practical, but it has been used as a baseline for comparison with more advanced techniques [42, 21].

Gradient Routing has been shown to decrease the delay when compared to direct contact [42]. Similar schemes have been proposed for routing data to-wards base stations in sensor networks. For example, the history-based protocol presented for ZebraNet is a gradient routing strategy [23]. A theoretical analysis of gradient routing can be found in [42], and a discussion about predicting utility function values can be found in [32].

One of the shortcomings of gradient routing is that it can initially take a long time for a good custodian to be found, since it may take some time for the utility function values to propagate, or because the metric values in the region around the initial custodian are all equally poor. One approach that has been shown to reduce the delivery latency is to initially use random forwarding until the utility value reaches a certain threshold [42]. This hybrid approach initially allows a message to actively explore the network until it finds a good carrier, and then it uses the standard utility routing to efficiently reach the destination. Burgess *et al.* use a similar technique to quickly propagate a new message in their epidemic routing variant [3].

Link Metrics

Routing strategies that use link metrics resemble traditional network routing protocols. They build a topology graph, assign weights to each link and finally run a shortest path algorithm to find the best paths. This requires the most network information as each node must have sufficient knowledge to run a routing algorithm. Link weights are assigned to try and provide optimal service to the endpoints, based on some performance metric: the highest bandwidth, lowest latency, and the highest delivery ratio. In delay-tolerant networks, the most important metric is the delivery ratio, since the network must be able to reliably deliver data. A secondary metric is the delivery latency. Thus, the challenge is to determine a system for assigning link metrics that maximizes the delivery ratio and minimizes the delivery latency. Some metrics may also attempt to minimize resource consumption, such as buffer space or power.

The first paper that proposed using link metrics for routing in delay-tolerant networks suggests that an appropriate metric is to minimize the end-to-end delivery latency [21]. The intuition is that this minimizes the amount of time that a message consumes buffer space, and thus it should also maximize the delivery ratio since there is more space available for other messages. Their work uses a metric that is the

time it will take for a message to be sent over each link. Since this value may depend on the time a message arrives at a node, the authors present a time-varying version of Dijkstra's shortest path algorithm. Finding the path that delivers the message with the shortest delay has also been used for proactive mobile networks [28].

Jain *et al.* present a variety of different metrics for networks with precise schedules, each of which require different amounts of information. However, all of the metrics assume that the contact schedule is precise. The first metric, called Minimum Expected Delay (MED), is the metric that requires the least amount of information. It assumes that the queuing time is zero, and that the average of the sum of transmission time, propagation delay, and waiting time is known precisely. This value is the expected delay: the average amount of time it takes for a message to go from one node to another, assuming that all arrival times are equally likely. The next metric is Earliest Delivery (ED). The queuing delay is assumed to be zero, and the propagation and transmission delays are assumed to be known precisely. The path is selected that will get the message to the destination at the earliest time. This requires the complete contact schedule, whereas MED only requires the average value of the waiting time. The next metric is Earliest Delivery with Local Queuing (EDLQ), which uses the buffer occupancy at each node to add an estimate of the queuing delay to the ED metric. Finally, the Earliest Delivery with All Queues (EDAQ) uses the information about the traffic demands for all nodes in order to compute the exact queuing delay. This paper shows that the protocols with more information have higher delivery ratios and lower delay. However, for some scenarios the difference between them is small.

2.4 Observations

If we plot the approximate location of each of the routing strategies discussed here using the Replication and Knowledge properties, as shown in Figure 2.8, we can see that the edges along the two axes are very well explored, but the middle space is not. The middle represents strategies that take advantage of both Replication and Knowledge in order to improve the delivery ratio and decrease the delivery latency. In particular, the area represented by the lower left-hand corner, that uses a small amount of both Replication and Knowledge, is likely to yield routing strategies that can be applied in the real world, as they will not require precise schedules or substantial amounts of configuration, nor will they consume large amount of resources by flooding the network with duplicate messages. The variants of Epidemic Routing that take advantage of some learned topology information are a good first step in this direction. In this thesis, we attempt to address the area

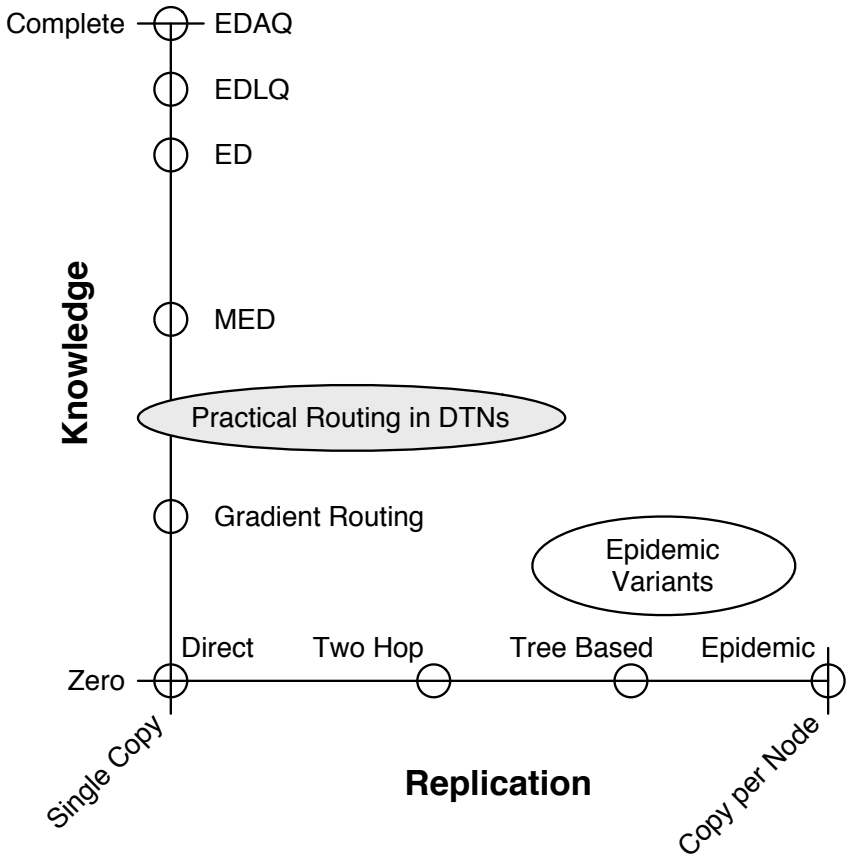


Figure 2.8: DTN Routing Strategy Properties

indicated by the shaded ellipse. This region represents protocols that use learned knowledge about the network, and use small amounts of replication. An early version of this work was presented at the SIGCOMM DTN Workshop, 2005 [22].

Chapter 3

Protocol Design

This chapter presents a shortest path routing protocol for delay-tolerant networks. Its design is based on routing in traditional networks, but some design decisions were modified for this new environment. First we discuss some of the issues in selecting a path metric and present the metric we use. Next we investigate when to make routing decisions, and finally how to distribute topology information.

3.1 Path Metrics For DTNs

Paths must be carefully selected to extract the best performance from a network. In a DTN, the primary requirement is that messages are reliably delivered. Thus, delivery ratio is a very important metric. Unfortunately, it is not clear how a metric can be constructed to directly maximize delivery ratio along a path. To resolve this problem, we follow the same approach as Jain *et al.* and choose to minimize the end-to-end delay [21]. This reduces the amount of time a message occupies buffers in the network, which intuitively should reduce the number of messages dropped, assuming that buffer overflow is the primary cause of loss. Additionally, previous work in the distributed systems community has shown that it is not possible to implement many important algorithms, such as consensus, election, or membership, using networks that provide an asynchronous, time-free model [10]. However, it is possible to implement them using an asynchronous timed mode [5]. Thus, it seems useful to strive for timely delivery of messages.

In a delay-tolerant network, the end-to-end delay has four components. First, the message must wait for the next contact to arrive (waiting time). Next, the data queued ahead of the current message must be delivered (queuing delay). The message must then be transmitted (transmission delay), and finally the signal must

propagate to the next hop (latency). Delay is an attractive metric because these four factors can be combined into a single number, assuming that sufficient information is available. However, to simplify the discussion in this paper, we assume that links have a very high throughput and low latency, which means that the waiting time is the only significant factor.

A variety of metrics for minimizing the end-to-end delay in a DTN have been explored by Jain *et al.* [21]. However, most of them require knowledge of future contact arrival times. An exception is the Minimum Expected Delay (MED). This metric assigns a cost to each edge equal to the average waiting time plus the transmission delay. Once this value is computed, the contact schedule is not needed. Assuming that message arrival times are uniformly distributed, the waiting time probability distribution is a piecewise linear function. It is a straightforward application of basic probability to compute the expected value. The derivation of the metric is included in the AppendixA.

We propose a variant of MED we call the Minimum Estimated Expected Delay (MEED). Instead of computing the expected waiting time using the known contact schedule, MEED uses the observed contact history. This assumes that the future connectivity will be similar to the previously observed connectivity. The details of this metric are reviewed in Section 3.6.

3.2 Routing Decision Time

The earliest opportunity that the path for a message can be decided is at the source, which is called source routing. This is a simple approach but it is inappropriate for our protocol because as the message travels closer to the destination, the nodes will likely have more recent and accurate information about the destination's connectivity. Hence it seems natural that these intermediate nodes can make better decisions than the source.

The next time to make forwarding decisions is when a message arrives at an intermediate node, which is called per-hop routing. When the message arrives, the node determines the next hop for the destination and places it in a queue for that contact. This is also not a good solution for DTNs, as changes to the topology could occur after the message arrives. This would result in the message waiting to be forwarded over a sub-optimal link.

In order to make routing decisions with the best possible information, we use what we call per-contact routing. Instead of computing the next hop for a message in advance, the routing table is recomputed each time a contact arrives. After a new routing table has been computed, we examine each message in the buffer to

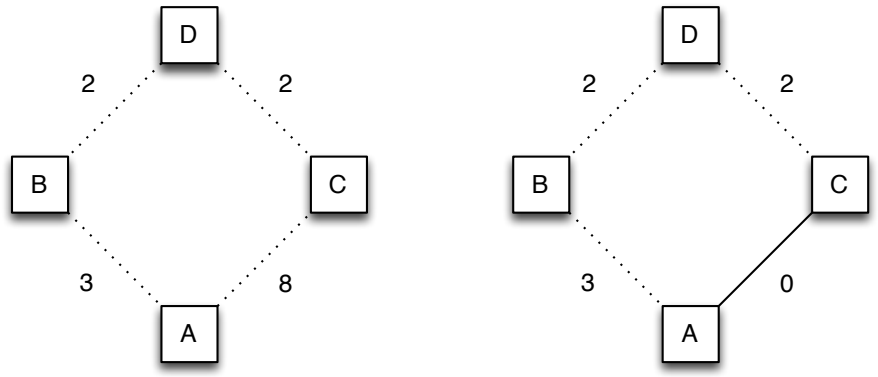
determine if any of them need to be forwarded over any of the available contacts. This assures that each routing decision is made with the most recent information. The disadvantage is that this approach uses more processing resources, as the routing is recomputed frequently. Additionally, the routing must be recomputed before any messages may be forwarded. Thus, there may be some additional delay before a link will be used. As long as the processing power of the nodes is appropriate for the size of the network, this delay will not be significant. However, it may be a limiting factor in scaling this approach to very large networks.

Since we are recomputing the routing table each time a contact becomes available, we can improve the performance further by temporarily assigning the contact a cost of zero in our local routing table. This value is used when computing the routing table, but is not propagated to other nodes. This "short circuits" the routing decisions made by the link-state protocol, allowing messages to take advantage of good timing. This is similar to the approach used in some epidemic routing variants [7, 44], and to what Handorean *et al.* call a "path update" [14]. Per-contact routing combined with short circuiting is effective for delay-tolerant networks because it guarantees that decisions are always made with the most recent information possible, and it can take advantage of serendipitous contact arrivals to make the routing more efficient.

For example, imagine we have a network with four nodes. Node A has a message for node D. There are two possible next hops: B with a total path cost of 5, and C with a total path cost of 10. This topology is shown in Figure 3.1(a). Thus, the current routing state says the message should wait for B to reach D. However, node C connects first. Thus, the cost to go from A to C becomes zero, as shown in Figure 3.1(b). With per-hop or source routing, the message would remain queued at A waiting for the path with cost 5, through node B. However, per-contact routing takes advantage of this unforeseen contact and delivers the message to C, where it will wait for a path with cost 2.

3.3 Impact of Deferring Routing Decisions

Making routing decisions as late as possible seems like a clear win for delay-tolerant networks because it allows the path taken by a message to change while it is in transit. For example, consider the situation where the next hop for a message never arrives due to a failure. At some point, source or per-hop routing would give up and drop the message because the next hop has disappeared. However, per-contact routing will automatically use an alternate contact, as soon as the routing metric decides that is the best path.



(a) (b)

Figure 3.1: Per-contact routing example

Unfortunately, this design decision have one drawback. Link-state routing is loop free only if the same topology is used to make all the routing decisions along the path of a message. Source routing guarantees that this occurs because all the routing decisions are made at the source. In connected networks, per-hop routing assumes that the topology does not change while the message is in transit. This is reasonable since the end-to-end delay is measured in milliseconds, and topology changes are relatively rare, but this assumption does not hold for DTNs. If the link weights change while the data is in transit, it is possible for a packet to get passed between two nodes indefinitely. For a loop to occur, the link weights must change enough so that when the packet gets to one of the nodes, the routing directs the packet back the way it came.

A situation where this loop could occur in a DTN is shown in Figure 3.2. Initially, node B has a message for node A, and the shortest path is BCA. When node C connects to B, the message is forwarded to C. Now, imagine that the cost of the contact BA decreases because A connects to B. Meanwhile, the cost of the CA contact increases to 6 because C has not been connected to A for some time. At this point, the situation is the mirror image of how it began. The message would have been delivered if it had stayed at B. Now if B connects to C, the message will be sent back to B. If the connectivity pattern is periodic, the message will bounce between B and C indefinitely. Short circuit routing aggravates this problem because the link cost between B and C no longer matters, so the link costs need to fluctuate less.

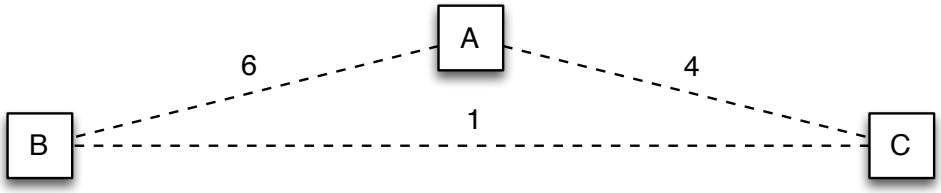


Figure 3.2: Routing loop caused by per-contact routing

This problem could be mitigated by adding hysteresis. In order to backtrack, the path must improve upon the next best path by some threshold. The threshold could increase if a node is revisited multiple times. We do not implement a solution in our simulations because it would add a significant amount of complexity. In our simulations, we have observed situations where messages do backtrack, but have not seen an actual loop where a message makes no forward progress. A real implementation may need to include a solution because while this situation seems unlikely to occur, it would cause significant disruption if it did.

3.4 Topology Distribution

Once we have costs for individual links, the information needs to be distributed throughout the network. Traditional networks typically use link-state or distance vector algorithms for this purpose, although other choices are possible. We chose to implement a link-state routing algorithm for two reasons. The primary reason is that there is a natural match between flooding in link-state algorithms and epidemic message distribution. Flooding distributes a copy of each link-state table to all nodes. Epidemic replication distributes a copy of each message to all nodes. Thus, we can implement a link-state protocol in a DTN using an epidemic algorithm, which is known to be very robust.

The secondary reason we chose to implement link-state routing is that it provides the complete topology at each node, which allows the topology to be updated in a single contact. If a node has been disconnected for a long time it can obtain the entire topology in a single exchange with any other node. A distance vector algorithm would only distribute paths that pass through the other node, and multiple exchanges would be required to obtain the entire topology.

Link-state routing does have its disadvantages. First, each node must store the entire topology in its routing tables, which could be larger than the state required for distance vector routing. Second, merging topology information from multiple nodes

becomes more complicated because there is more information to be synchronized.

3.5 An Epidemic Link-State Protocol

Upon connection, nodes exchange summary vectors that list the link-state tables the nodes have received. Each table is tagged with a sequence number which permits the nodes to determine which ones are the most recent. The nodes exchange any missing updates so that they both have the same topology state. Then they recompute their routing tables and finally can forward messages to the other node. Since we use per-contact routing, the metric for each link is temporarily set to zero to represent the fact that messages can be immediately sent to the other node. A protocol state machine, shown in Figure 3.3, is maintained for each connection.

When the local link state table changes, an update must be propagated to all nodes in the network. This is an expensive operation. To reduce the overhead, a node may suppress updates that it decides are unnecessary. However, it is essential that it continues to make routing decisions using the table that it last advertised, otherwise the routing tables could be inconsistent between connected nodes, which might cause an immediate routing loop. Our simple implementation propagates an update if at least one weight changed by more than 5%, or if a new contact has been added.

To estimate the overhead of this protocol the size of each protocol message must be determined. There are two protocol messages exchanged: summary vectors and topology updates. The summary vector contains one (source id, sequence number) pair for each node in the network. If we assume each value has a fixed length, the size of this message scales linearly with the size of the network. The topology update contains a set of (source id, sequence number, link partner id, metric) tuples for each contact. In the worst case, the complete topology must be transmitted. In this case, the topology update has a size that is $O(ND)$, where N is the number of nodes in the network, and D is the degree of each node. If we assume that the average node degree is constant, this overhead also scales linearly with the size of the network. As a somewhat realistic example, if we encode these messages as arrays and assume each value is a 32-bit integer, an 802.11 packet containing 1 500 bytes of data can store an update with information about 92 links, which would be sufficient for a network with 15 nodes with an average degree of 6. Thus, the overhead is acceptable for small networks, but may be a problem for very large networks.

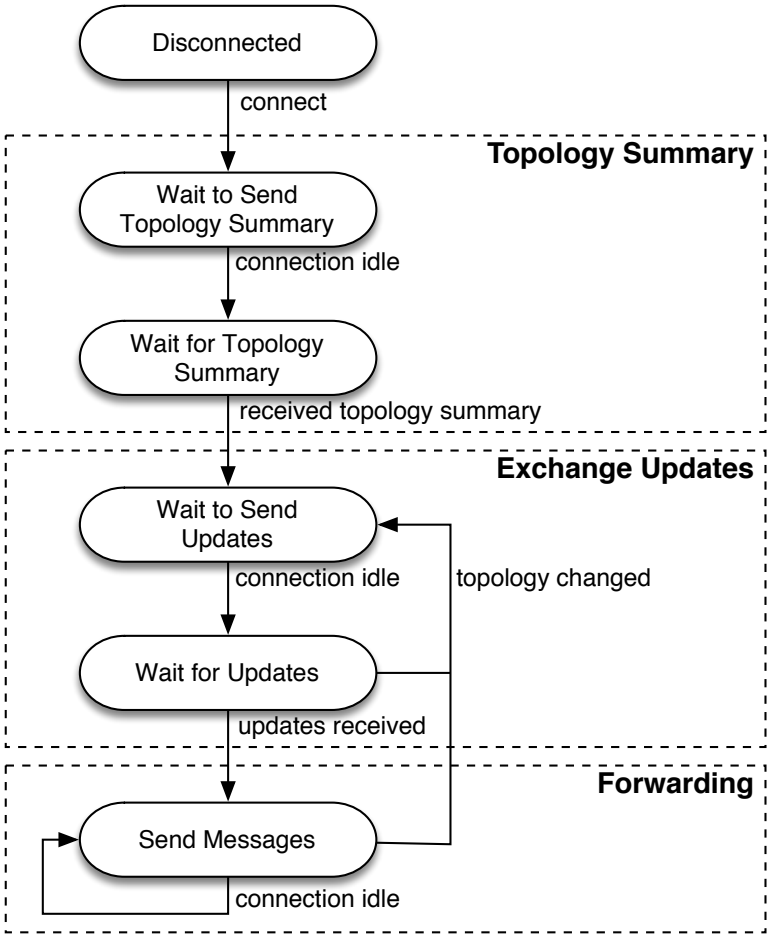


Figure 3.3: The epidemic link-state protocol state machine

3.6 The MEED Metric

Our protocol relies on an estimate of the connectivity in order to make intelligent routing decisions. To do this, we use the expected-delay metric originally presented by Jain *et al.* [21]. This metric computes the expected delay for a message to go from one node to another using a given contact, assuming that all message arrival times are equally likely. The derivation of this metric is simple, and is included in the AppendixA. The final computed metric value is shown in Equation (3.1), where n is the total number of disconnected periods, d_i is the duration of a given disconnected period, and t is the total time interval over which these disconnections were observed.

$$\frac{\sum_{i=1}^{n} d_i^2}{2t} \tag{3.1}$$

The original metric is computed using the contact schedule for the entire period that the network is in use. However, it is possible to compute this metric for any arbitrary time period. If we assume that the future behavior of a contact will be similar to the past, we can use the value for the past as the current estimate. We present three techniques for computing this metric: the infinite history window, the sliding history window, and the exponentially-weighted moving average.

3.6.1 Infinite History Window

The simplest approach is to compute the metric over the entire history of a node. This is easy to do since the computation only requires a sum of squared disconnection times, the current time, and the initial time where the observations began. Unfortunately, this scheme is somewhat impractical. First, the sum of disconnection times continues to increase without bound. This means that it may be difficult to compute this value if a node is running for a long time. Second, if the connectivity pattern for a contact changes for some reason, the metric will average both the old behavior and the new behavior. This means this approach will take a long time to adjust to changes.

There is also a small problem with this simple approach. If the contact is currently down and we include the interval between now and the last time the contact was available, the metric value can actually decrease. Consider the case where a contact has a periodic pattern and is up for 5 time units, then down for 15. This pattern has an expected delay of $15^2/(2 \times 20) = 5.625$. In Fig. 3.4(a) we plot this contact's state in the bottom part of the graph, and the metric in the top part of the graph. The raw metric line in Fig. 3.4(a) increases when the contact is down,

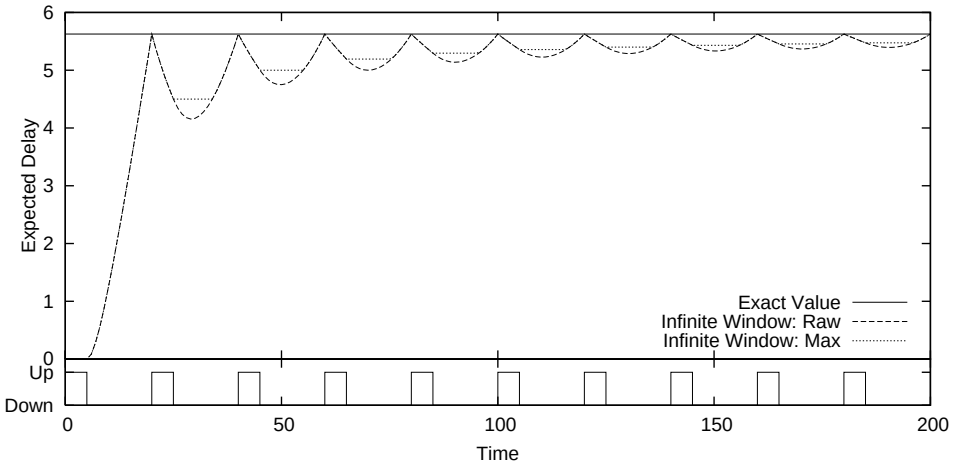
and decreases when it is up. However, immediately after the contact goes down, it continues to decrease. This occurs because we do not know when the contact will become available next, so including the current disconnected interval makes a best-case estimate, assuming that the contact will come up in the next instant. To solve this, we take the maximum of this best-case estimate and the metric when the contact was last available, as shown by the "max" line in Fig. 3.4(a). This way, the metric will only decrease when the contact is up, and never when the contact is down. For a periodic contact the infinite window computation equals the ideal value just before the contact comes up. However, the metric approaches the correct value as time approaches infinity.

3.6.2 Sliding History Window

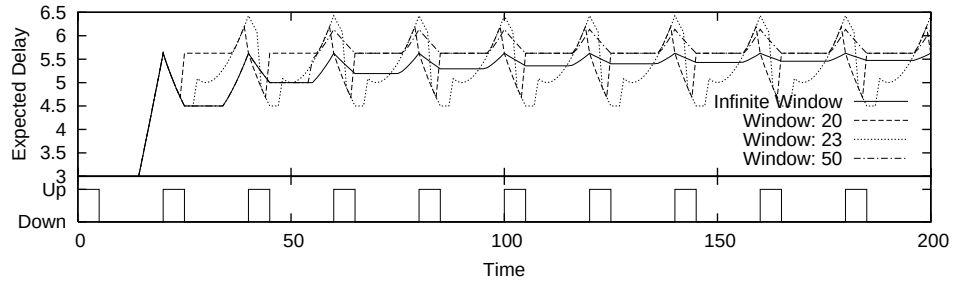
In order to make the metric more responsive to changes, we can compute it over a sliding window. This means that if the contact-connectivity behaviour changes, the old behavior will eventually be removed from the history window and no longer included in the metric. Unfortunately, this also works in reverse. If the history window does not capture a large enough sample of the contact behavior, it may fluctuate significantly and not accurately represent the average behavior. In order to avoid some undesirable behavior that happens when averaging over partial up or down periods, we round our history window up to the last connect or disconnect event. Because of this, the metric will oscillate around the true average, even if the window is exactly the same size as the period of the contact.

Consider the case where a contact has a periodic pattern and is up for 5 time units, then down for 15. This pattern has an expected delay of $15^2/(2\times 20) = 5.625$. Even when the sliding window size is a multiple of the period (20), it oscillates around the exact metric, as shown in Figure 3.4(b). However, windows that are much larger than the period, such as the window of size 50 in the figure, oscillate less.

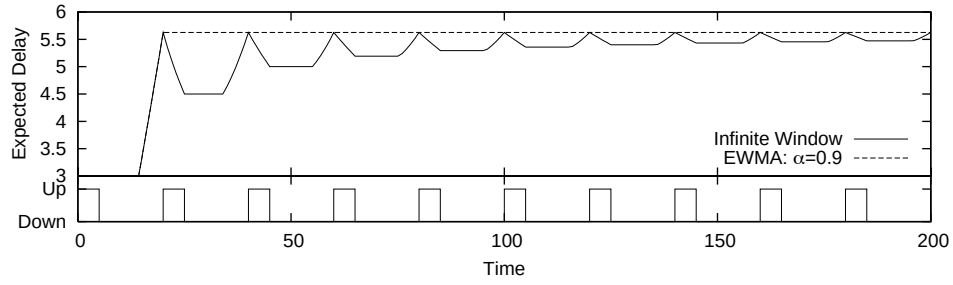
This version is more complex to compute than the infinite history window approach. For this metric, we must record all the connection and disconnection times within the window. In theory, this means that the potential amount of data that must be stored is infinite, as the contact could go up and down an infinite number of times within a given time window. Practical systems will have some logical minimum connection period that is dictated by the link technology. However, the number of disconnected periods that must be stored could potentially be very large.



(a) Infinite window expected delay



(b) Sliding window expected delay



(c) Exponentially weighted moving average expected delay

Figure 3.4: Comparison of MEED metric variants

3.6.3 Exponentially Weighted Moving Average (EWMA)

It is not initially obvious how to use an exponentially weighted moving average to compute a continuous metric like the expected delay. However, it is possible to compute it if we pretend that contact behaviour is perfectly periodic, and use an EWMA to estimate the average connection and disconnection lengths. Thus, each time a contact goes up or down, we update the estimate as follows:

$$D = \alpha D + (1 - \alpha) d_i$$
$$C = \alpha C + (1 - \alpha) c_i$$
$$E = \frac{D^2}{2(D + C)}$$

E is the estimated delay, D is the average disconnection time, and C is the average connection time. With this computation, α is a tuning parameter, where a higher value means that the metric will react slowly to changes, but will be more robust to short term perturbations. With a periodic contact, as shown in Fig. 3.4(c), the metric will match the exact average.

This implementation has the advantage that it requires fixed resources to implement. Unfortunately, it also has a built-in tendency to underestimate the average waiting time. The reason is that the averaging of D weights all disconnection periods equally. However, the exact expected delay computation weights long disconnections much more than small disconnections, because in those periods a message must wait longer, on average, and because there is a higher probability of arriving during a long wait period.

The figures in this section show that the differences between these three metrics are relatively small. Thus, we use the infinite window metric, unless otherwise specified.

Chapter 4

Experimental Evaluation

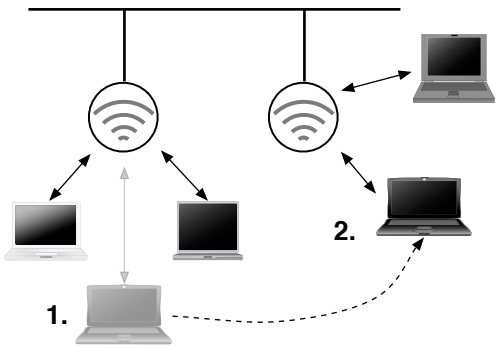
In this chapter, we present the results of extensive simulations, in order to evaluate the performance of our proposed protocol. We first describe the scenarios we simulate, then we look at microbenchmarks designed to examine specific aspects of the design. Finally, we examine the performance of the protocol in plausible scenarios. For the evaluation, we created DTNSim2, a discrete-event simulator for delay-tolerant networks [46]. It is based on the simulator used for the original DTN routing paper by Jain*et al.* [21]. It uses FIFO, reliable links with fixed bandwidth and delay.

4.1 Scenarios

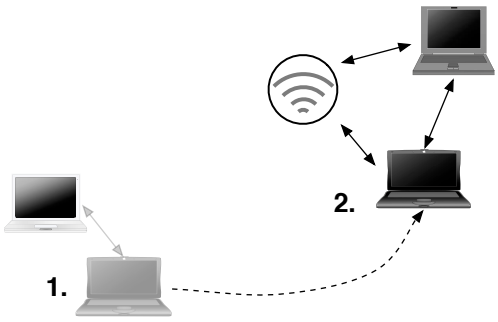
In order to validate our requirement of providing good performance over a range of connectivity patterns, we present simulation results from two very different scenarios. To give the scenarios a basis in reality, we used real mobility data.. The first scenario is based on extensive wireless LAN traces from Dartmouth College [6]. The second scenario uses the schedule from Seattle's bus network [45].

4.1.1 Wireless LAN Scenario

In this scenario, mobile users carry computing devices with radios. When they are in range of another user, they exchange data. This represents an ad-hoc DTN that could be created by students at a school, employees at a company, or people attending a conference. We used mobility data from Dartmouth College's extensive wireless LAN traces [6]. They record the network connectivity of more than 2 000 users over two years. The traces show when each user connects and disconnects to



(a) Wireless LAN scenario



(b) Ad-hoc DTN scenario

Figure 4.1: Wireless trace converted into a DTN scenario

any of Dartmouth College's 500 access points. This data is useful because while the mobility appears to be random, there are patterns that can be exploited.

To transform the wireless LAN traces into an ad-hoc DTN, we consider two nodes to be connected when they are associated with the same access point at the same time. Access points are also DTN routers. In order to make the scenarios a manageable size, only a connected subset of the nodes from the wireless LAN trace are included. As an example, consider the wireless LAN scenario shown in Figure 4.1(a). At time 1, the trace file will show that the laptop user is connected to the access point on the left. Later, it moves out of range of the access point, and at time 2 it associates with the second access point. A DTN scenario that could be generated from this data is shown in Figure 4.1(b). One laptop and one access point from the original scenario were removed.

The raw Dartmouth data set is much too large for us to simulate. To select a reasonably sized subset of the data, we used the data from a single month (February, 2003). We then randomly selected an initial node. Next, we found that node's "good" neighbours. We defined a "good" neighbour as one that had at least ten opportunities to communicate over a period of one month. Then we randomly selected a new node from the good neighbour set, and added that node's good neighbours to the set. We repeated this until we had 30 nodes, and we generated ten different topologies in this fashion. This simple algorithm guarantees that we do in fact have a connected graph, where each node has some opportunities to communicate with the rest of the network. In order to eliminate the warm-up and cool down effects, we only record statistics for messages generated in the second week.

4.1.2 Bus Scenario

This scenario represents a city that is providing a DTN network by equipping its buses with wireless devices. As the buses come within wireless range of each other, they exchange messages. To generate this data, we used the Seattle bus network schedule, as provided by the University of Washington's Intelligent Transportation Systems group [45]. This region's bus system is quite large, so again we needed to select a subset of it. To do this, we selected all the buses that ever service a single route as part of their day. This includes 36 buses, which is comparable in size to the parameters used for the Dartmouth data. We compute all the times where the buses are within wireless range, which we considered to be 200 metres, the nominally quoted range for 802.11b. The scenario lasts for five weekdays and each day's schedule is identical. We only record statistics for messages generated during the second day.

4.1.3 Comparison

These two scenarios were selected because they represent very different connectivity patterns. The wireless LAN scenario has unpredictable mobility, with some statistical regularity. The bus scenario, on the other hand, is a planned and scheduled system, and thus has the exact same connectivity on each of the five days. Additionally, the nature of the connectivity is very different between the scenarios. To quantify this difference, some scenario metrics are shown in Table 4.1. The scenario duration is obviously different, as they were generated from different data sets. The number of nodes is comparable in the two scenarios. The average node degree is the average number of contacts that appear at least once during the scenario. The

Table 4.1: Comparison of the wireless LAN and bus scenario parameters

Parameter	Wireless LAN Scenario	Bus Scenario
Scenario Duration	28 days	5 days
Nodes	30	36
Avg. Node Degree	15.6	22.8
Avg. Up Time	2050 seconds	77.0 seconds
Avg. Inter-contact Time	9.27 hours	5.32 hours
Avg. Total Connected Duration	23.7 hours	0.405 hours
Avg. Num. Connections	41.6	20.0

bus scenario's average degree is larger, even accounting for the fact that there are more nodes in the scenario. This indicates that the nodes tend to be connected to more neighbors than in the wireless LAN scenario. The up time is the time where each contact is available, the inter-contact time is the time where each contact is down, and the total connected duration is the total up time for a single contact. These values are much, much longer in the wireless LAN scenario, because users carrying laptops tend to stay in one place and move slowly, whereas the buses only stop for breaks and move quickly. Finally, the number of connections is the number of times a single contact becomes available. Counter-intuitively, this is higher in the wireless LAN scenario. The reason is that there are some contacts which go up and down a large number of times in a short period of time. These anomalies are likely caused by poor wireless LAN connectivity, and not frequent mobility.

4.2 Microbenchmarks

In order to understand the performance of our protocol, we first consider microbenchmarks designed to investigate specific aspects of the performance of the protocol. First we look at the performance under ideal conditions, before varying buffer space and bandwidth. Next we consider the MEED metric and variants. Third, we closely examine the performance of per-contact routing and hop-by-hop flow control. Finally, we study the overhead of our protocol.

We compare the performance of the MEED protocol to three other delay-tolerant network routing protocols: the earliest delivery (ED) and minimum expected delay (MED) metrics [21], and epidemic routing [47]. The ED protocol is used to illustrate the performance that can be achieved if complete contact schedule

data is available. MED is presented because it uses the same average-delay metric as MEED, except that its values are computed using future knowledge. Finally, Epidemic is another protocol that does not require schedule information.

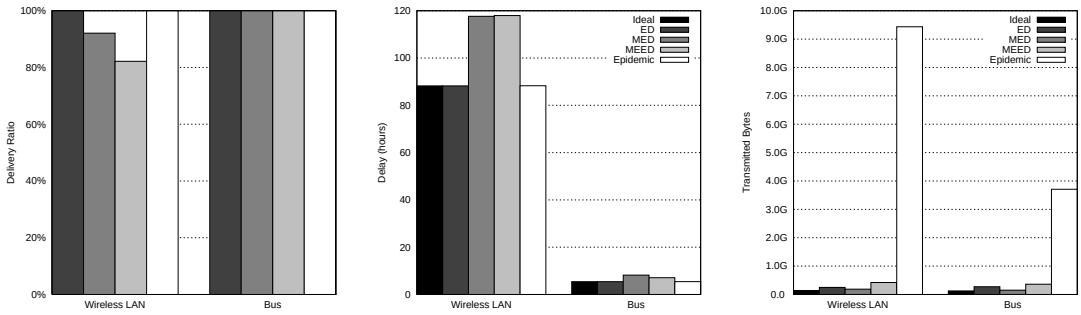
To provide a baseline for comparison, we measure the performance of an "ideal" protocol. This protocol is the ED metric with infinite buffer space and infinite bandwidth. This protocol has two attractive properties. First, if it cannot deliver a message, then it is not possible for that message to be delivered by any protocol. Thus, it provides an upper bound on the delivery ratio. For this reason, we express all delivery ratios as a percentage of the messages delivered by the ideal protocol. Second, when it delivers a message, it is not possible for that message to be delivered earlier. Thus, it provides a lower bound for delay. When computing average delay, we only include delivered messages.

In order to quantify the cost of the protocols, we measure the total number of bytes transmitted. This includes protocol bytes and data bytes, and measures the total amount of bandwidth consumed by the protocol. In order to establish a minimum value, we modify the ideal protocol to select minimum hop-count paths for each message. Thus, it delivers all the messages with the minimum number of transmissions.

In this section, we run the simulator with an ad-hoc email workload. Each node generates 10 messages at regular intervals, sent to a single random destination node. Each message is 10 000 bytes long, which corresponds to users exchanging small files or email messages. For the wireless LAN scenario, the interval is every 6 hours, for a total of 112 message generation times. For the bus scenario, the interval is every hour, for 120 message generations.

4.2.1 Ideal Performance

First, we consider the performance of the four protocols under ideal conditions. We simulate the two scenarios with infinite buffer space and infinite bandwidth. The delivery ratios are shown in Figure 4.2(a). For all scenarios with infinite resources, ED is the same as the ideal protocol, and by definition delivers all the messages. Epidemic delivers 100% of the messages in these scenarios by flooding the network. For the bus scenario, all protocols deliver all the messages because the bus schedule is predictable and repetitive. In the wireless LAN scenario, MEED manages to deliver little more than 80% of the messages. Considering that this protocol has no future knowledge, and the mobility in this scenario is random, this is respectable. It is important to recall that while the ED and MED protocols outperfom MEED, in reality it is not possible to use them in the wireless LAN scenario because a schedule of human mobility cannot be obtained in advance. The MED protocol,



(a) Delivery ratio (b) Delay (c) Transmitted bytes

Figure 4.2: Performance under ideal conditions

which uses the true average instead of MEED's estimated value, delivers around 90% of the messages, simply because its metric is more accurate because of it is computed using future knowledge.

The delay results, shown in Figure 4.2(b), are more interesting. This figure makes the differences between the two scenarios very clear, as the ideal average delivery delay for the wireless LAN scenario is around 85 hours, whereas it is under 10 for the bus scenario. ED and Epidemic match the ideal delay, as expected. MEED's delay is larger than the ideal by a fair margin. However, it compares favorably with MED, matching it in the wireless LAN scenario and slightly beating it in the bus scenario.

We plot the number of transmitted bytes in Figure 4.2(c). This graph shows that Epidemic transmits the most messages, by a very wide margin. This is not surprising, as it floods the network with each message. MEED transmits more messages than the other shortest path protocols. This is because the MEED metric is constantly being recomputed, so it is possible for a message will take a bad path, only to be forced to backtrack later. This problem is emphasized in the wireless LAN scenario because the connectivity is random. This is the reason that MEED requires significantly more transmissions in that scenario, than in the bus scenario. Additionally, ED and MED have no protocol overhead, as they assume that all nodes have the topology information from the start.

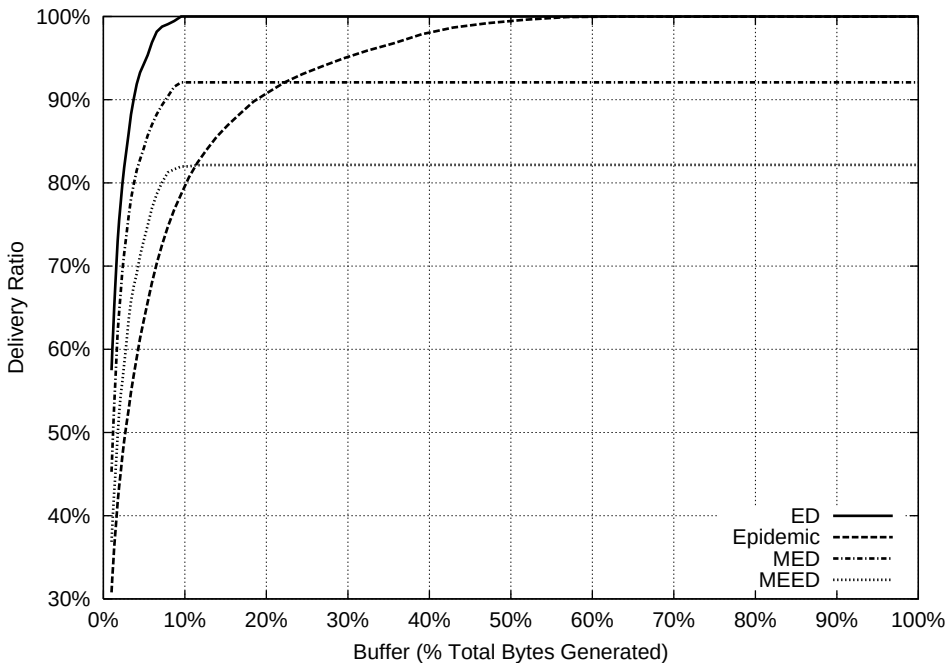


Figure 4.3: Wireless LAN delivery ratio with varying buffer size

4.2.2 Impact of Buffer Size

Next we evaluate the impact of buffer space on the DTN routing protocols. We evaluate each scenario with infinite bandwidth contacts, but limited buffer space. The amount of buffer space is shown as a percentage of the total number of bytes generated. For this experiment, we only consider the wireless LAN scenario because the results for the bus scenario are similar. Looking at the graph of the delivery ratio in Figure 4.3, we can immediately see that buffer size has significant impact on the epidemic protocol. This is because it relies on having sufficient buffer to have a copy of every message at every node. In this particular scenario, it needs buffer for approximately 60% of the data generated, and there is a very predictable relationship between the buffer size and the delivery ratio. The other protocols require much less buffer space because they use a single copy of each message. Only when the buffer size drops below 10% of the total traffic generated does the delivery ratio of the other protocols decrease. Decreasing the available buffer space is equivalent to increasing the amount of data being transferred. Thus, for transferring large amounts of data MEED has better performance than epidemic routing, which is the only other protocol that does not use future information.

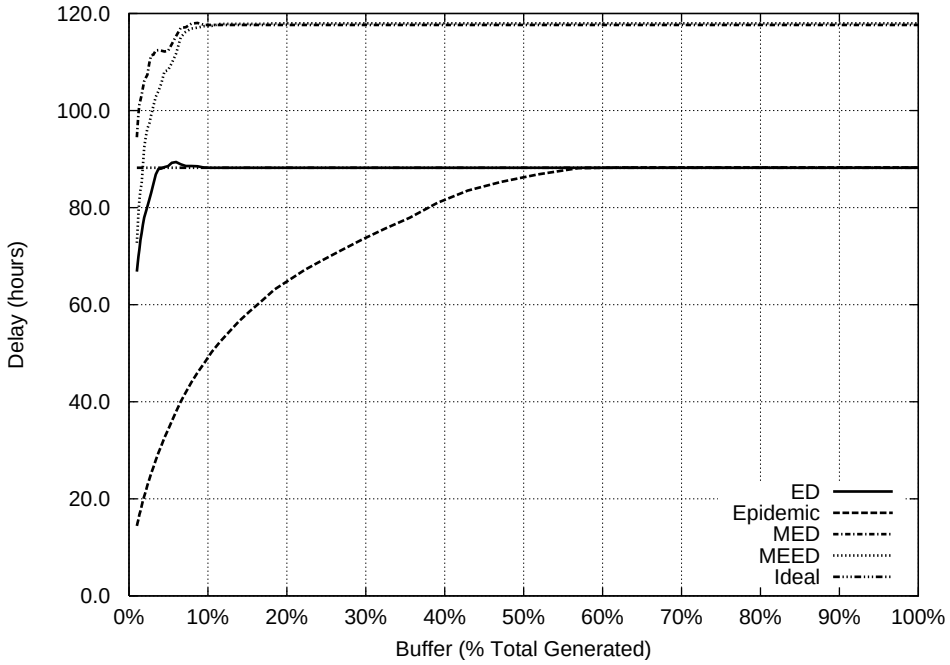


Figure 4.4: Wireless LAN delay with varying buffer size

Looking at the latency results in Figure 4.4, we can see that the delay is not sensitive to the buffer space as the performance does not change much. The average delay for the ED and Epidemic protocols matches the ideal minimum latency, as expected. Epidemic's delay decreases as it drops messages because it drops the oldest messages first. Thus, the delivered messages are only those that are delivered quickly. Interestingly, the latency for MED and MEED are very similar, even though MEED is an estimate of the MED value, and intuitively should be worse. The reason is that it does not deliver as many messages, so it is not valid to compare the two directly, without taking into account the delivery ratios in Figure 4.3.

4.2.3 Impact of Bandwidth

For this experiment, all nodes have infinite buffer space but the contacts have varying data rates. We test the rates between 100 kbps and 10 Mbps. Again, we only present the results from the wireless LAN scenario as the bus scenario results do not provide any additional insight. It is important to note that because the average connection times are so long for the wireless LAN scenario, a large amount of data can be transferred even at very low data rates. The email workload

generates a small amount of data (336 MB), so assuming that all the contacts are up for the average connection time shown in Table 4.1, a data rate of 1.31 Mbps is sufficient to deliver all the messages that are generated in the scenario over a single hop. Thus, it is important to note that it is only at data rates below 2 Mbps that this scenario begins to become bandwidth limited.

For all four protocols, the delivery ratio decreases slightly as the bandwidth decreases, as shown in Figure 4.5. However, there is no change in their relative positions until the bandwidth is extremely low. It is important to note that ED and MED are more sensitive to the reduced bandwidth than Epidemic and MEED. With infinite bandwidth, ED was able to deliver 100% of the messages, as shown on the right hand side of Figure 4.3. However, even with 10 Mbps links, it drops approximately 2% of the messages, and it drops 15% of the messages with 1 Mbps links. MEED's performance, by contrast, is nearly flat. It only drops an additional 4% of the messages over the same interval. The reason is that ED and MED use source routing and select paths assuming that there is no competing traffic. With the bandwidth restrictions, messages might miss the contact they were supposed to take and be undeliverable. Epidemic, on the other hand, tries all paths. Thus, if one contact is being used, it will try others. Similarly, MEED's per-contact routing allows it to adapt to the varying conditions.

The results for the latency in Figure 4.6 show that the delay for all protocols increases slightly as the bandwidth decreases, due to the longer transfer times. This trend is probably understated, as all the protocols drop messages as the data rate decreases. This graph shows that MEED's performance is slightly worse than MED, as expected. At data rates below 1 Mbps, all the protocols behave erratically. The reason is that this is the point where the protocols really begin to drop significant amounts of messages, so the data points here are all measuring the delay for slightly different sets of messages. MED shows an unusual bump around 2 Mbps, where its delay increases before decreasing again. The increase is due to the bandwidth restrictions increasing the average delay for most of the messages. The decrease is then caused by the delivery ratio decreasing since at that point, MED is only able to deliver the messages which take the least time to arrive at the destination.

4.2.4 MEED Metric Error

The previous results have shown that MEED can achieve reasonable delivery ratios and delays. This implies that it is a reasonable metric for selecting paths. Since this metric estimates a physical quantity, average delay, we wished to test how useful this metric might be for predicting delivery day. To answer this question, we analyzed the performance of the infinite window metric, the sliding window metric

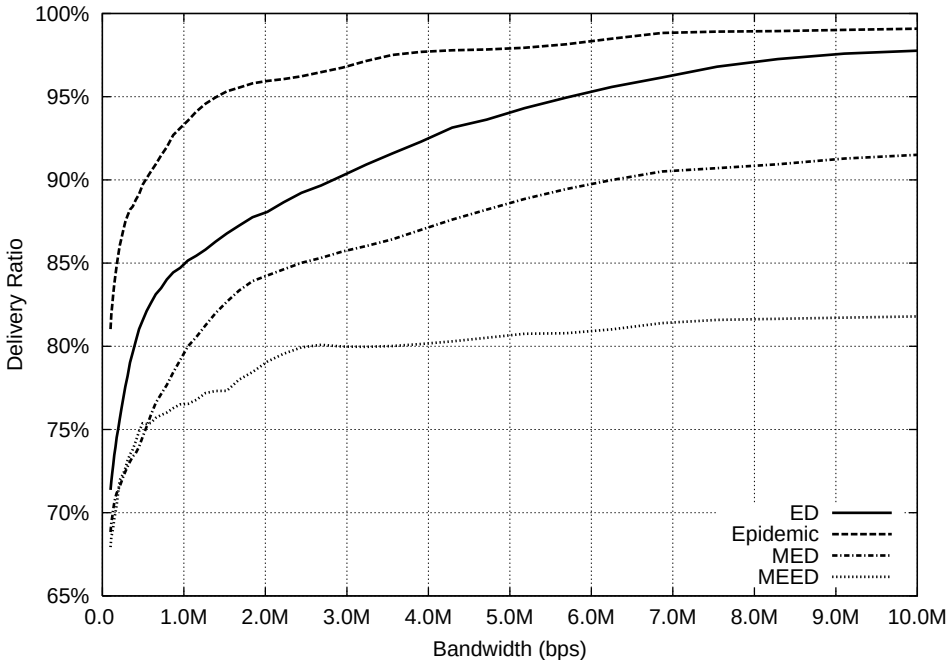


Figure 4.5: Delivery ratio with varying link bandwidth

(window = 2 days), and the EWMA metric ($\alpha = 0.9$) over a single hop. We selected 30 random contacts from the wireless LAN scenario and sampled the metric values at one minute intervals, and compared them to the actual waiting time. Sample metric values for a single contact are shown in Figure 4.7. In order to make the error between different contacts comparable, we divided the absolute error by the average waiting time for each contact. A histogram of the relative errors is shown in Figure 4.8. This histogram shows that most of the samples fell between -4 and 2 average waiting times of the true value. This is a large range of errors, which indicates that none of the metric variants are precise. Additionally, the histogram clearly shows that the infinite window metric has a tendency to overestimate the waiting time, while the other two metrics tend to underestimate. Both the sliding window and EWMA variants have a peak at zero error, which is what we ideally would like to see. However, their distribution clearly underestimates the delay, on average. The infinite window metric peaks a bit above zero, but due to the shape of the distribution, its overall average is somewhat closer to zero.

We also evaluated the metric's performance over complete paths. We instrumented the simulator to record the MEED metric value when the message is forwarded from the source to another node. This is the latest time that the source

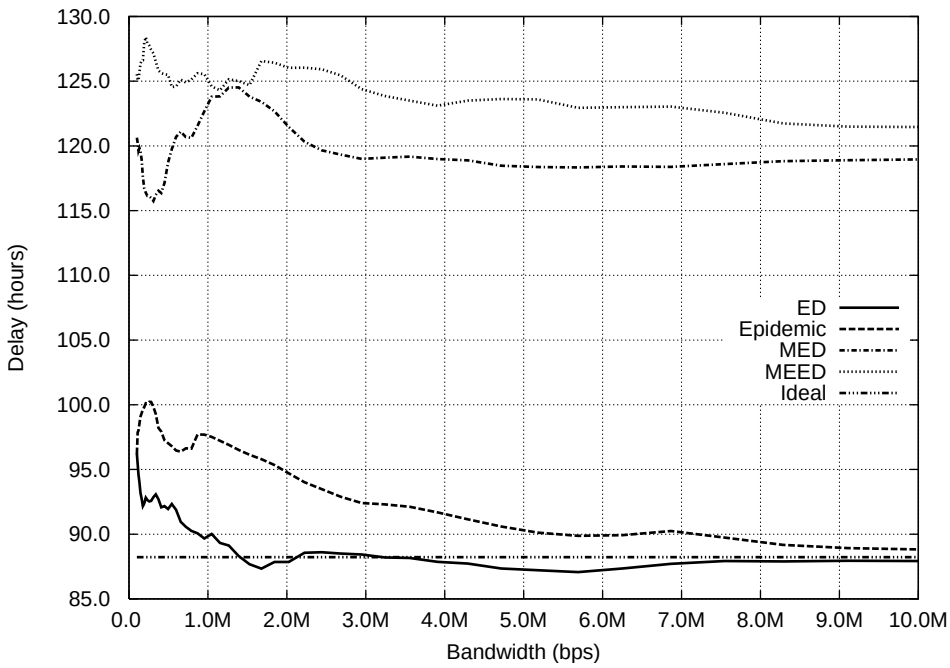


Figure 4.6: Delivery latency with varying link bandwidth

can make an estimate, and thus should be the most accurate estimate available to that node. We plotted the prediction for each message against the actual recorded end-to-end latency. Ideally, these two values would be equal, and all the points on the $y = x$ line. The results for the infinite window metric from the wireless LAN scenarios are shown in Figure 4.9. This figure shows that there is limited correlation between the MEED value and the actual end-to-end delay, as the points appear to be randomly distributed.

The results in this section indicate that the ability for MEED to estimate the end-to-end delay is not ideal. Part of this limitation is simply because it is an average. Consider a contact that has two different behaviours, one where the waiting time is small, and another where the waiting time is very large. The exact average for this contact will be a bad estimate of the end-to-end delay for both sections, although the average errors will be zero. This suggests that more research is required here. It may be possible to estimate a worst-case end-to-end delay by incorporating the variance or a confidence estimate. Alternatively, time-varying statistical or machine learning approaches may be more accurate.

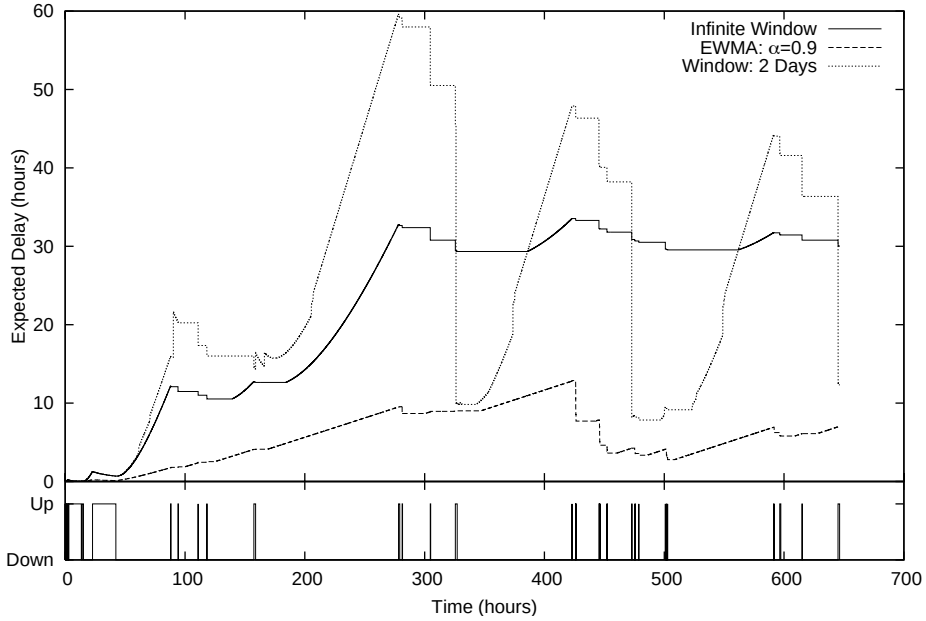


Figure 4.7: Sample expected delay metrics for a single contact

4.2.5 MEED Variants

The previous results all used the infinite window variant of the MEED metric. Here, we will consider the two additional variants presented in Section 3.6. Additionally, in order to determine what impact the choice of metric has on the performance, we also implemented a fourth metric variant that is intentionally a bad choice. This metric uses the time since the contact was last available, and so we call it the "Last Up" metric. Intuitively, this should not help in selecting a good path. In fact, in many scenarios the last available contact will not be seen for a long time, and so this will be a poor choice.

The delivery ratios for all four variants and both scenarios are shown in Figure 4.10(a). Interestingly, in the wireless LAN scenario, the Last Up metric actually delivers more messages than the infinite window and EWMA metric. The reason for this becomes clear when looking at the transmitted bytes, shown in Figure 4.10(c). The Last Up metric transmits far more bytes than the others. Since the metric changes so often, it effectively ends up forwarding the message along nearly every available contact, which explains the large number of transmissions. In the wireless LAN scenario, because the mobility is pseudo-random, this strategy actually works

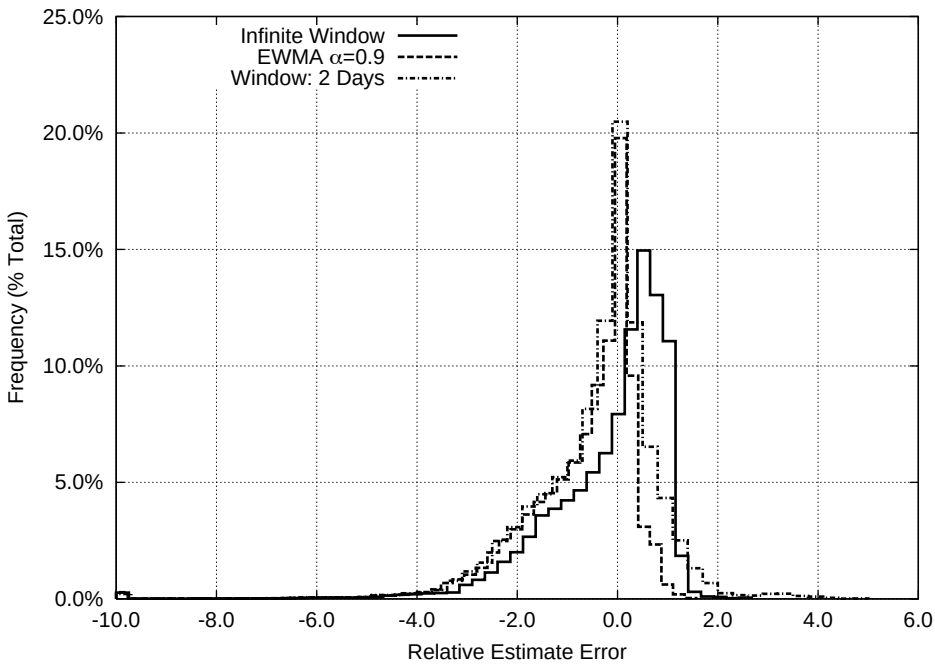


Figure 4.8: Histogram of relative estimation error for 30 contacts

fairly well.

Last Up performs significantly worse in the bus scenario. The reason is that this scenario is one where Last Up is exactly the wrong choice. When a bus passes another one, it will be a long time before it passes it again, as generally they are traveling in opposite directions. Thus, in this case the message ends up getting passed to buses that have just passed the destination, and won't see it for a while. If it remained on one bus, the messages would likely end up being delivered.

For the other three metrics, the performance is very similar. They are each slightly better than the others in some aspect, and worse in another, and thus there is no clear winner. However, it is clear that changing the metric can have a significant impart on the behaviour of the system, as show by the Last Up metric. Thus, it may be possible to improve the performance of our system, simply by substituting a superior metric.

4.2.6 Routing Decision Time

In Section 3.2 we argued that making decisions as late as possible is advantageous in delay-tolerant networks, as it allows the protocol to adapt to changes in both

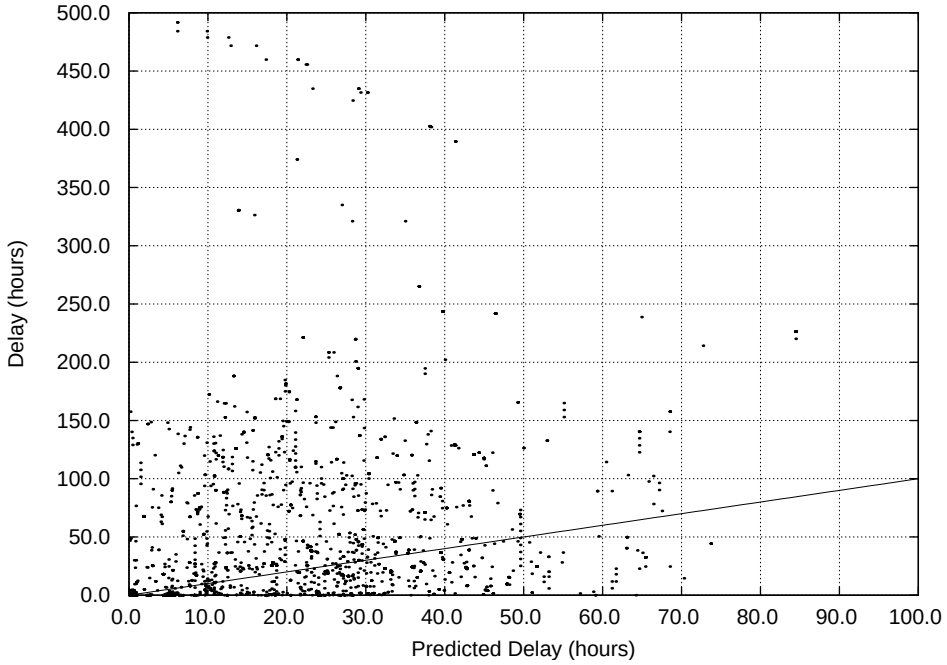
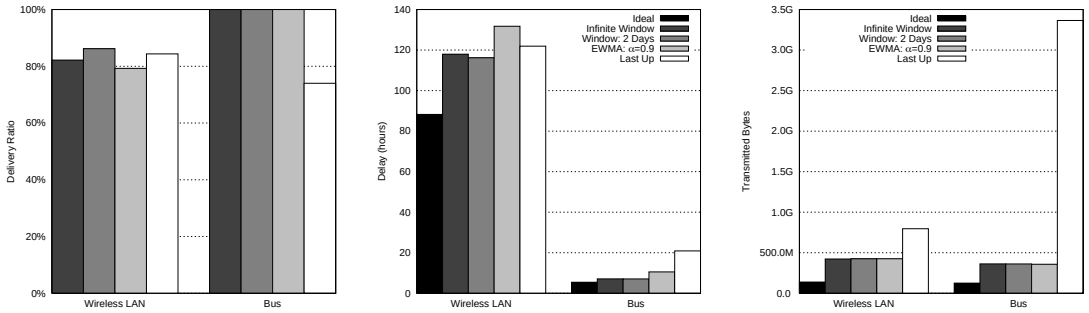
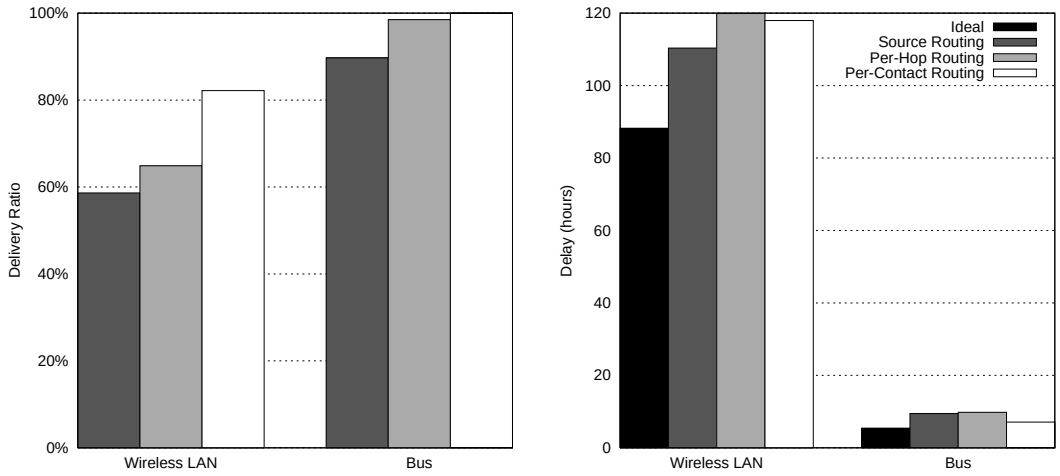


Figure 4.9: Scatter plot comparing estimated and measured end-to-end delay



(a) Delivery ratio (b) Delay (c) Transmitted bytes

Figure 4.10: Performance of MEED variants



(a) Delivery ratio

(b) Delay

Figure 4.11: Comparison of MEED with source, per-hop, and per-contact routing

the estimate of metric values and network topology. In order to investigate the performance impact of this design choice, we implemented two variants of the MEED protocol: source routing and per-hop routing. The source routing variant plans the entire path as soon as possible at the source, after which no changes are made. The per-hop routing variant calculates the next hop when the message arrives at a new node. At that point, the message can only be sent to that next hop. The delivery ratio for these variants is shown in Figure 4.11(a), and the delay is shown in Figure 4.11(b). These graphs show that per-contact routing is significantly better. In the Wireless LAN scenario, per-contact routing delivers a third more messages than source routing. The delay increases because more messages are being delivered. In short, per-contact routing is a key part of the MEED protocol's performance.

We also studied the impact this design decision has on other protocols to show that it is applicable to other protocols. We modified the MED protocol to support per-contact routing, instead of source routing. This means that if a contact becomes available that does not have the best average performance, but is a good choice when it is available, the MED per-contact protocol can take advantage of it. While this changes the performance of MED for all scenarios, it has the most impact when there are bandwidth limits. Thus, we plot the delivery ratio of this protocol when there is infinite buffer space but limited bandwidth in Figure 4.12. This figure shows that per-contact routing improves MED in two ways. First, it increases the

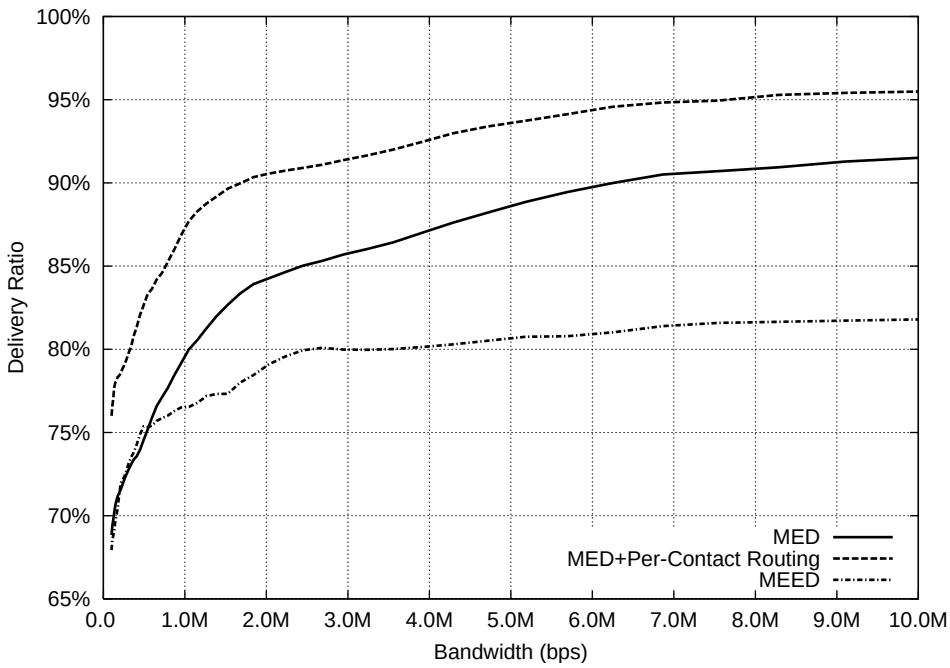


Figure 4.12: Comparison of MED with and without per-contact routing

raw delivery ratio by around 4%. Second, it decreases MED's sensitivity to the low bandwidth, as its delivery ratio decreases less than MED with source routing. For comparison, we have plotted MEED on the same graph. It is essentially the same as MED with per-contact routing, except that its metric is less accurate which causes it to lose between 10% and 15% more messages. This shows that the accuracy of the metric has a significant impact on the delivery ratio. Per-contact routing also decreases the average delay for MED from 119 hours to 111 hours. It achieves these gains with some cost: it increases the number of transmitted bytes from 190 MB to 260 MB.

4.2.7 Hop-by-Hop Flow Control

The shortest-path protocols studied here use a "drop tail" queue policy. If there is insufficient buffer space when a message arrives, it is dropped. This source of loss could be reduced by using hop-by-hop flow control, so that the message is only forwarded if sufficient buffer is available. This scheme has been shown to be effective at dealing with congestion in wireless sensor networks [19].

The performance with and without flow control for the wireless LAN scenario

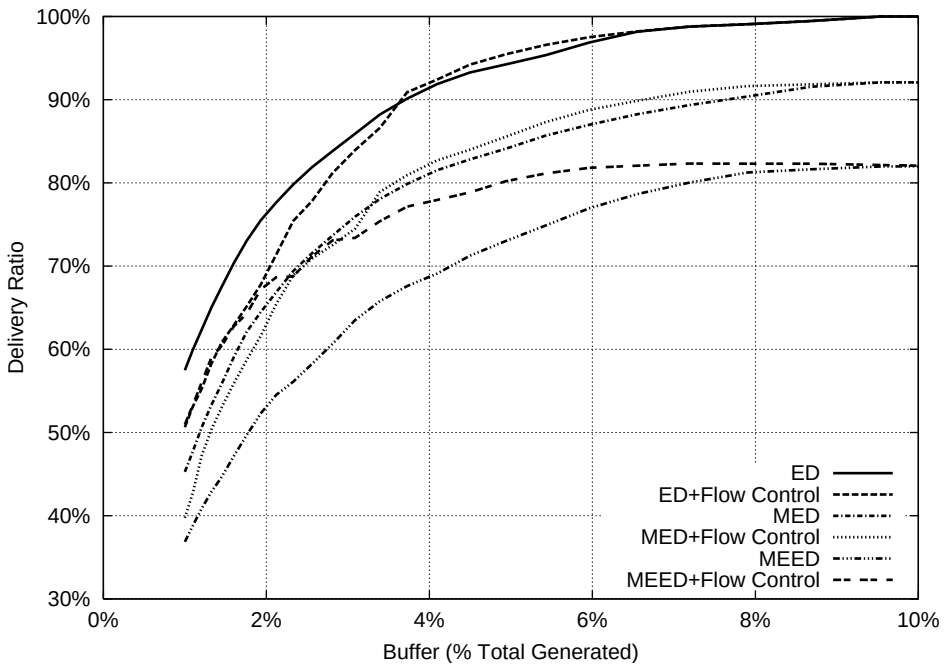


Figure 4.13: Wireless LAN performance with hop-by-hop flow control and limited buffer space

with limited buffer space is shown in Figure 4.13. For ED and MED, flow control increases the delivery ratio by a few percent when the buffer space is between 3-10% of the generated messages. When the buffer is very small, it actually makes things worse. The reason is that with flow control, if the next hop is full, there is now less buffer space available at the previous node. This can in turn cause other nodes to block, which prevents some messages from making progress towards the destination. If the message had been dropped instead, at least one previous message would be able to make forward progress.

For MEED, flow control improves the performance by a significant margin when the buffer space is less than 8% of the generated bytes. The difference is that MEED uses per-contact routing, so when one next hop is blocked because the buffer is full, it is able to attempt delivery via an alternative route. This means that it can spread the load across the network. From these results, it seems that hop-by-hop flow control is a significant improvement for dealing with low-buffer scenarios, although care must be taken when the available buffer space is extremely limited.

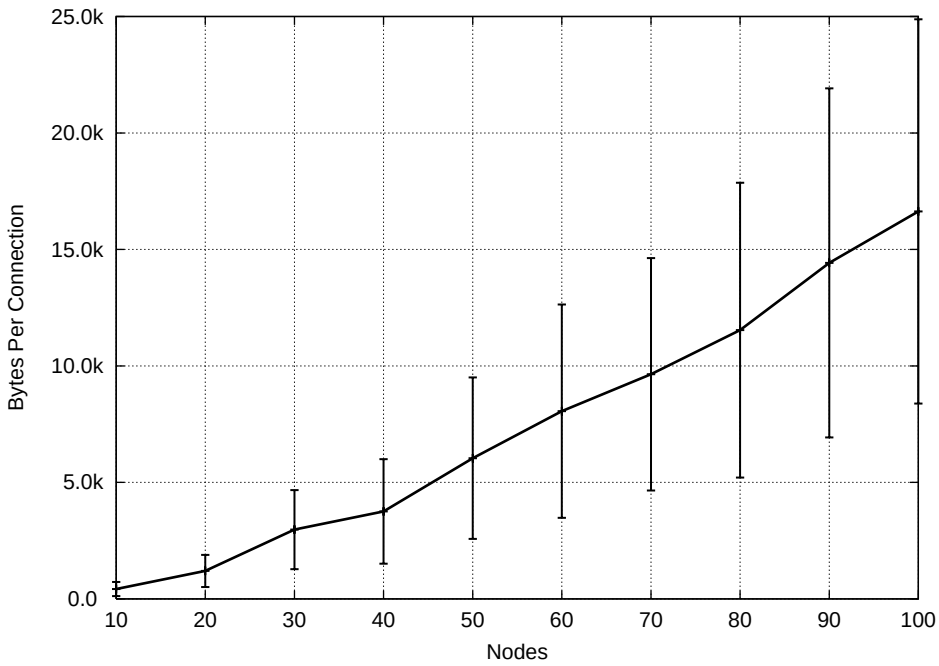


Figure 4.14: Protocol overhead per connection

4.2.8 Protocol Overhead

In order to measure the overhead introduced by the epidemic distribution of the link-state tables, we generated 10 topologies from the Dartmouth data with different sizes. We simulated them for one month without any traffic, and discarded the overhead in the first week. The protocol is initiated each time a connection is established, so if a scenario has more connections it will generate more overhead. To compensate for this, we normalized the overhead by dividing the total protocol bytes by the total number of connections. This gives us the average bytes of overhead that is exchanged each time a connection is established.

The average overhead with the 90% confidence interval is shown in Figure 4.14. The theoretical analysis showed that the overhead grows linearly with the size of the network, assuming that the node degree stays constant. The overhead here appears to grow slightly faster than linearly. The reason is that in our scenario, nodes that share an access point form a clique, which means that adding nodes tends to increase the average node degree.

The actual amount of data exchanged is still reasonable. With a network of 50 nodes, less than 10 kB is exchanged each time a connection comes up. This will

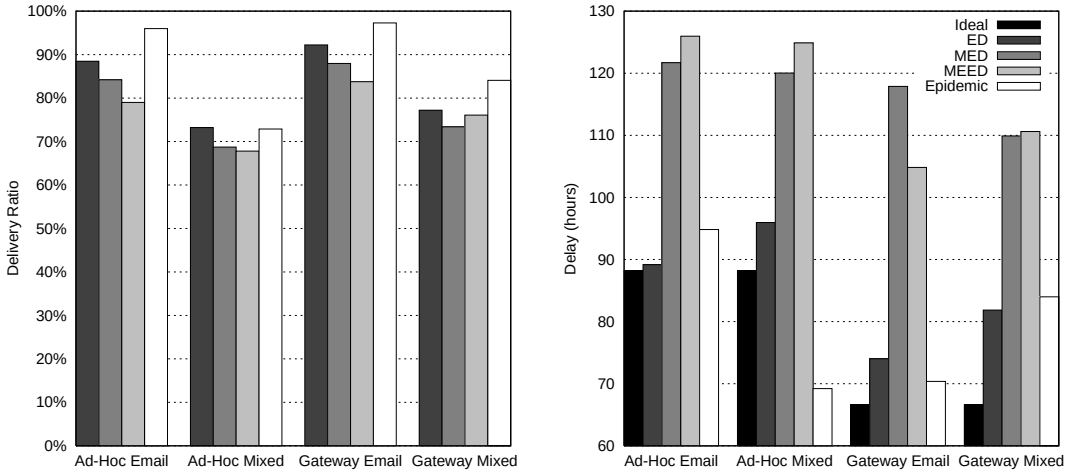
take less than a tenth of a second to transmit at 802.11's base rate of 2 Mbps. This is a tiny portion of the bandwidth, even if the contact is only up for a few seconds. With 100 nodes, the average overhead per connection is less than 25 kB, which is still less than a second of transmission time. However, this overhead could become unmanageable as the network grows, that hierarchical routing may be necessary for very large networks.

4.3 Realistic Scenarios

The previous section looked at the performance of the protocol under various ideal conditions to understand the impact of individual parameters on the performance of our protocol. In this section, we look at the performance of four workloads with plausible combinations of parameters. For each workload, we set the buffer size on each node to 1 GB because the current price of a 1 GB flash disk is under $100 USD, so it is reasonable to assume that even small embedded devices can be equipped with at least this much storage. We set the bandwidth of each contact to 2 Mbps since that is the base rate for 802.11b, and thus represents a reasonable estimate of the bandwidth available between two nodes within wireless range. Experiments using 802.11 in drive-by scenarios report a wide range of average goodputs. The results vary from an average goodput of 5.5 Mbps [12], down to 0.9 Mbps [35, 3], depending on speeds and antenna location. Thus, 2 Mbps seems to be a realistic midpoint. Additionally, it is reasonably safe to assume that future wireless technology improvements will push that data rate even higher.

The scenarios are created by varying two parameters: traffic composition and communication pattern. For traffic composition, we have an email workload and a mixed message workload. The email workload is the one that was used in the previous section, where each node generates 10 messages that are each 10 000 bytes long. The mixed message workload only generates 5 messages of 10 000 bytes, but also generates 2 messages that are 1 000 000 bytes each, representing larger files being exchanged, such as digital photos. The mixed-message workload generates 20 times more traffic than the email workload.

For the communication pattern, we use an ad-hoc pattern and a gateway pattern. The ad-hoc pattern is the same as was used in the previous section, representing users in the network communicating with each other. The gateway pattern represents users communicating with a single Internet gateway. To simplify the scenario, we assume that the gateway node has an infinite bandwidth, zero latency connection to the Internet, so all outbound and inbound messages are delivered instantly. For the wireless LAN scenario, we selected the node with the highest



(a) Delivery ratio

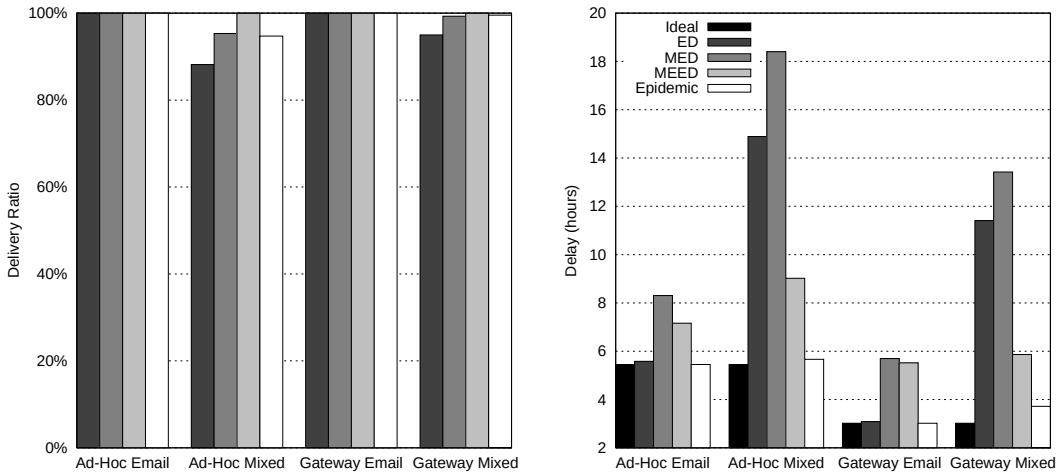
(b) Delay

Figure 4.15: Wireless LAN scenario performance under realistic conditions

degree as the gateway, since it is the node with the best connectivity to the rest of the network. For the bus scenario, we added a fixed gateway node at a centrally located bus stop.

4.3.1 Wireless LAN

The results for the four workloads in the wireless LAN scenario are shown in Figure 4.15. While, the different workloads are generally similar, a few differences can be observed. The epidemic protocol consistently delivers more messages than any other protocol. In the mixed-message workload, the performance of all the protocols decreases due to the increased amount of traffic in the network. However, MEED's performance decreases less than the others, to the point that it outperforms MED in the gateway workload. MEED's ability to adapt to changing network conditions gives it a distinct advantage in this scenario, where there is a significant amount of network congestion. Unfortunately, MEED still has a longer delay than the other protocols, except in the gateway e-mail case where it has lower delay than MED.



(a) Delivery ratio

(b) Delay

Figure 4.16: Bus scenario performance under realistic conditions

4.3.2 Bus

The bus scenario has a higher average node degree than the wireless LAN scenario. Correspondingly, the average path length is also lower. This means that bandwidth and buffer restrictions typically have less impact on the performance in this scenario, as many of the deliveries occur directly from the source to the destination. Thus, the delivery ratios are still 100% for all the protocols with the e-mail workloads, as shown in Figure 4.16(a). However, this scenario also has a very short average connection time, when two buses pass each other on the road. When there are larger messages in the network, as is the case with the mixed workload, the performance changes dramatically. With this workload, MEED outperforms the other protocols. The reason is that ED and MED plan their routes with the assumption that there is no competing traffic. This assumption is violated with this workload, as a single large message can occupy almost all of a contact's time. Epidemic performs poorly in this scenario as there is only 11% buffer space and because the contacts do not last long enough to exchange all the messages.

As for the delay, shown in Figure 4.16(b), MEED outperforms MED for all the bus scenarios. For the gateway scenarios, MEED's delay is second only to the Epidemic protocol. In the other cases, it is only an hour or two more than the ideal minimum delay.

Chapter 5

Performance Enhancements

The previous chapter shows that the performance of the MEED protocol is reasonably good. However in many scenarios the epidemic protocol is able to deliver more messages with less delay, though at much greater cost. In this chapter, we examine ways to try and bridge that performance gap by using a small number of copies of each message, in combination with MEED's knowledge about the network topology. We should be able to intelligently select multiple paths in order to approach the performance of the Epidemic protocol, without incurring its massive and unscalable resource cost. We present two techniques for creating multiple copies of each message: replication and retransmission. In the replication scheme, we explicitly create multiple copies of the message. With the retransmission scheme, we use a timeout to indicate when a copy is needed.

5.1 Replicated MEED

With the Replicated MEED approach, we simply make multiple copies of the message as it propagates through the network. There are three design decisions to be made here. The first is when and where to create copies of a message, the second is which path to use for each copy, and the third is how to organize the messages in the buffer. Our design philosophy is that adding replication should only improve the performance of the protocol, insofar as we can guarantee it. Thus, any decisions we make should not change how the original message is handled.

5.1.1 Copying Decisions

The intention of the Replicated MEED protocol is to use as few additional copies as necessary in order to improve the delivery ratio and decrease the delay. Thus, there is a clear cost/performance trade-off. This problem is very similar to limiting the cost of flooding protocols. Any of the techniques surveyed in Section 2.3.1 could be applied here.

Techniques for deciding when to make copies can be divided into two broad categories. The first category includes schemes which explicitly limit the replication. These schemes maintain some sort of counters to stop the replication once the desired number of copies have been made. The second category is for schemes that make a copy when certain conditions are met. These strategies attempt to make copies only when there is a good reason. Hybrid strategies, which only make copies when there the conditions are met, and that have a fixed maximum number of copies, are also possible. We investigate one strategy of each type.

Source Replication

In order to limit the cost, we use a simple scheme where the source makes a limited number of copies. This is similar to Two-Hop Relay presented in Section 2.3.1. This strategy presents a very easy way to adjust how much replication is acceptable. However, if the source only has a single next hop, it cannot take advantage of alternate routes. With this scheme, we always deliver a copy along the best path, then deliver a copy to the next best path to the destination, as explained in Section 5.1.2.

Backtracking Replication

The MEED protocol was designed to permit messages to backtrack. When this occurs, it indicates that the metrics have changed and the routing now computes a different best path. However, the original path was at one point considered the best, and is likely still a good choice. Thus, this seems like a logical opportunity to exploit some redundancy. Instead of selecting only the current best path, we can select both paths. Thus, whenever a message will be sent back to the last node it came from, we will also make a copy of the message, which will be forwarded using an alternate route.

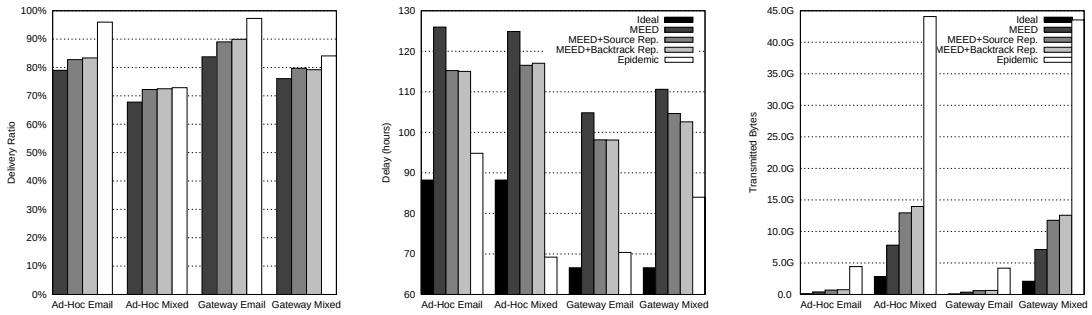
5.1.2 Selecting Alternate Paths

The concept of adding replication to the routing protocol is to try multiple routes simultaneously in order to avoid problems and to take advantage of unforeseen events. However, for this benefit to be realized, the paths must be at least partially independent. This means the paths must not share the exact same nodes. It is not necessary for the paths to be completely disjoint. In other words, it is acceptable if they have some nodes in common. The reason is that as long as there is a chance that the two messages will take different amounts of time to traverse the independent sections of their paths, then there is a chance that using both paths will be better than using the single best path.

In order to not change the minimum performance of the MEED protocol, we make no changes to how the path is selected for the first copy of a message. Thus, we need to select an alternate path for the copies. Since the MEED protocol allows nodes to backtrack, sending a message along an alternate path is more difficult than it may appear at first glance. We cannot simply give a message to a node that is not on the best path, as it will determine that it should immediately send the message back along the best path. Thus, we must mark messages in such a way that they are forced to travel along an alternate route to the destination. To do this, we include a list of forbidden nodes in the message header. To determine an alternate path, a node recomputes its routing table without any of the forbidden nodes. This generates a good path that will be independent for at least the immediate next node, although it will be possible for two messages to meet later along the path. The allows the original message to follow whichever path the routing system determines is best, and to backtrack as needed, while forcing the copy to not follow the same path.

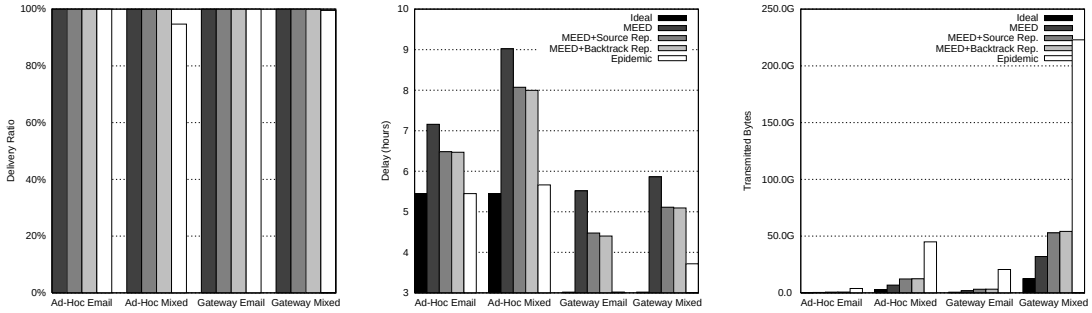
5.1.3 Buffer Management

The final design question is how to manage the replicas in the buffer. Following our philosophy of not changing the performance of the original protocol, the replicas are always lower priority than the original messages for both forwarding and buffer management purposes. Thus, if an unmarked message arrives, we will delete replicas until there is sufficient space for it. We also need to handle the case where we have a copy in the buffer, and receive another copy, potentially with a different priority. This complicates buffer management, as we essentially need to maintain two independent buffers that share the same storage. Similarly, we only forward replicas once we have finished processing all the original messages.



(a) Delivery ratio (b) Delay (c) Transmitted bytes

Figure 5.1: Performance of MEED with replication in the wireless LAN scenario



(a) Delivery ratio (b) Delay (c) Transmitted bytes

Figure 5.2: Performance of MEED with replication in the bus scenario

5.1.4 Evaluation

We implemented both the simple source replicated MEED variant, where the source sends one additional copy of the message, and the backtracking replicated MEED. We tested the performance of these two protocols on the realistic scenarios from Section 4.3. The results for the wireless LAN scenario are shown in Figure 5.1, and for the bus scenario in Figure 5.2.

These results indicate that adding replication increases the delivery ratio and decreases the delay in most scenarios. In the wireless LAN scenarios, replication increases the delivery ratio in all scenarios, as shown in Figure 5.1(a). In the ad-hoc mixed workload, the performance of the replicated variants even meets that of the

epidemic protocol. In addition, the average delay is reduced across all the scenarios, in both Figures 5.1(b) and 5.2(b). Interestingly, the two replication variants perform nearly identically, even though they use very different techniques for selecting when to make copies of a message. This suggests that either naive schemes are sufficient for extracting the performance improvement available from replication, or that the backtracking scheme is not very effective. More investigation is required to gain a better understanding of the performance trade-offs here.

The disadvantage is that adding more copies of the messages results in more transmissions. However, Figures 5.1(c) and 5.2(c) show that MEED with replication still uses less than a third the number of transmissions that the Epidemic protocol does. Thus, adding replication is a very viable way of improving the performance without an excessive increase in cost.

5.2 MEED with Retransmissions

Many applications for delay-tolerant networks will require reliable delivery. A traditional approach to achieving this is to use acknowledgments and retransmissions. This idea may seem ridiculous in a DTN, as the round-trip time could be extremely large and difficult to estimate. However, if we consider that this is simply another way of adding redundancy, we realize that setting the timer to an accurate value is not critical. If the timer is set too low, we will have extra copies in the network. If the timer is set too high, then the application may have to wait longer before receiving an acknowledgment.

Using acknowledgments and retransmissions has two advantages over the simple replication scheme. First, the source receives a confirmation that the data was in fact received. For the very large class of applications that need reliable delivery, this is a required feature. The end-to-end principle applies here: Applications should not trust users' critical data, such as digital photos, to a network that has any chance of losing it. If applications are going to be using acknowledgments anyway, it seems natural to investigate how the network layer can either use this additional information, or optimize the process. Harras and Almeroth have made a similar argument [15]. Second, this scheme adaptively adjusts the level of replication as needed. If an acknowledgment is received very quickly, no additional copies are injected. Conversely, if a message is lost or is taking a very long time, multiple copies will be sent which should naturally lead to a higher delivery ratio and potentially also to lower delay.

There is, however, one significant disadvantage. For retransmissions to occur, the source must keep a copy of the data in its buffer until it receives an acknowl-

edgment. This means that any retransmission scheme effectively has less buffer space available for forwarding, as it must reserve buffer space for local messages. There are two design decisions we investigate further. The first is how to send acknowledgments and the second is how to set the retransmission timer.

5.2.1 Acknowledgments

The naive implementation of acknowledgments in a delay-tolerant network is to create a tiny message with the acknowledgment and send it back toward the source using the same routing system as the original message. This works, and has the attractive properties that it is simple and it requires no modifications to the network layer. Unfortunately, this means that the acknowledgment will experience similar performance as the actual message, and thus it may be dropped or experience variable delays. We can easily do better.

A simple improvement is to make intermediate nodes aware of acknowledgments. While an acknowledgment is propagating back to the source, it may encounter a retransmission directed toward the destination. At each intermediate node, if we compare the acknowledgments to the messages in the buffer, we can remove any of the messages that have already been received. This is the same as the "death certificate" concept in epidemic algorithms [8], and it has been shown to reduce the buffer space requirements in delay-tolerant networks [41, 15].

A second improvement is to use epidemic replication to propagate the acknowledgments. Our performance results show that Epidemic Routing is very robust when there are sufficient resources. The acknowledgments will be very small, so even if there are a large number of them, they will not consume too many resources. Using an epidemic algorithm has three benefits. First, the acknowledgments will be delivered if there is some way to deliver them. Second, they will be delivered with very low delay. Finally, they will ensure that any retransmissions that are still in the network will be deleted as quickly as possible.

5.2.2 Retransmission Timer

In an ideal world, we would have an excellent estimate of the round-trip delay, and we would be able to compute a reasonable worst-case estimate, as TCP does. Unfortunately, this seems unlikely to be possible in delay-tolerant networks without hints from the network layer. If this layer has very accurate information, like the ED metric, it is very feasible to estimate the round-trip delay with very good accuracy. However, if it uses a metric like MEED, its estimates will not be useful for predicting delay, as shown in Section 4.2.4. This is certainly an open research

problem. Conversely, if we consider that this is simply a technique for adding redundancy, then we simple need to tune the retransmission timer until we reach the desired number of additional transmissions. Many adaptive approaches could be possible, but we do not investigate this idea further.

Chapter 6

Conclusions and Future Work

This thesis presented a design for a shortest path routing protocol for delay-tolerant networks. The protocol is a link-state routing protocol, where the link-state tables are distributed by an epidemic routing algorithm. This provides robust performance and permits a node to obtain the entire topology after a single contact. To estimate the cost of links, we designed the Minimum Estimated Expected Delay (MEED) metric, which estimates the average waiting time for a message to go from one node to another. In order to adapt to changes while messages are in transit, we use what we call per-contact routing. We recompute the routing table each time a contact becomes available, and then reroute messages accordingly. While this recomputes the routing table frequently, which may be expensive, our simulation results show that it significant decreases delay and increases the delivery ratio.

We presented techniques for further improving the performance of the protocol by using multiple copies of each message. This increases the cost to deliver each message, but increases the delivery ratio and decreasing the delay. We argued that many applications will need acknowledgments, so potentially routing protocols should take advantage of them in order to improve performance.

Our experimental performance evaluation shows that in many realistic scenarios, we are able to meet and even exceed protocols that either know the future contact schedule, or flood the network with messages. Since our protocol requires no configuration, it represents an important development for building real delay-tolerant networks.

6.1 Future Work

Our protocol uses an epidemic protocol for distributing link-state tables, and we proposed using an epidemic protocol for acknowledgments. These applications leverage its strong performance, and limits the cost by only sending very small messages. This suggests that perhaps epidemic routing could serve as a control plane, available for small high-priority messages. An important portion of user traffic could fit into this category, such as e-mail messages. This approach would provide fast, reliable delivery for small messages, while still supporting efficient delivery of larger data items. This hybrid strategy could be a powerful way of building reliable and performant routing infrastructure for delay-tolerant networks.

We briefly described some of the issues involved in selecting alternate paths in Section 5.1.2. However, we essentially ignored the problem. There is significant room for further investigation along both theoretical and practical lines. From a theoretical perspective, there is a question of how to select alternate paths, and how disjoint they must be before there is any benefit. Intuitively, it seems as though related problems in operations research might shed some light on these questions, as they may be similar to problems that arise in transportation. From a practical perspective, we need adaptive algorithms that select alternate paths that are the most useful. Our simple "copy when backtracking" policy is effective, but it seems as though better options probably exist. After all, the fact that backtracking is occurring indicates that a replica should have been sent along an alternate path much sooner. This is a difficult optimization problem that needs to balance the cost against the incremental benefit of making more copies.

The papers surveyed in Chapter 2 cover a large range of the mobility spectrum shown in Figure 2.1. Precise schedules are covered by the forwarding strategies, random networks are covered by epidemic and tree-based strategies, and the networks with implicit schedules are covered by some of the variants. However, one type of network in this spectrum has not been well-examined: networks with imprecise schedules. These networks have fairly predictable contact schedules, which should be leveraged to improve performance, but it is not clear if the techniques pioneered for precise schedules can be used without modifications. The bus scenario presented here is a perfect example for this type of work, as both a published schedule and measured data are available.

DTNs must be able to integrate multiple types of networks together. This means that techniques will be required that allow messages to be exchanged between DTNs that have different properties and possibly use different routing protocols. None of the current work in this area addresses this issue. The protocol presented here was designed with a single, moderately-sized DTN in mind. It will not be sufficient

for extremely large networks, or networks with widely varying requirements and performance characteristics. Additionally, there is an extremely important network that most DTNs will want to communicate with: The Internet. One proposal that enables communication between DTNs and the Internet is the Tetherless Communication Architecture [38]. Any widely adopted DTN routing protocol will need to address these issues.

Finally, it is impossible to determine what approach is the right one when there are no delay-tolerant networks being actively used. Perhaps the most important future work is to deploy DTN applications, and then see what routing problems occur in practice. At the moment, there are a number of prototype DTNs that are being used to measure connectivity properties [3, 18, 6]. These projects are an extremely important first step. However these networks are not yet being used. It is difficult to predict the requirements for DTN routing strategies and to evaluate their performance without any information about what traffic patterns would be relevant. Building more experimental deployments and applications is critical for to be able to determine where DTN routing strategies need improvement. Protocols that require no configuration, such as the one presented here, can help make this task feasible.

Appendix A

Derivation of the Expected Delay

The expected delay for a contact is computed assuming that all arrival times are equally likely. When the contact is up, the waiting time is zero. When the contact is down, the waiting time is the time until the contact comes back up again, as shown in Figure A.1. Since the arrival time probability distribution is uniform, to compute the expected value of the waiting time we can compute the area under the curve, and then divide by the length of the time interval. For a single disconnected interval, d_i, the area under the curve is given by $\frac{1}{2}d_i^2$. The area under a connected interval is 0. Thus, the final metric is given by:

$$\text{MEED} = \frac{\sum_{i=1}^{n} \frac{1}{2} d_i^2}{t} \tag{A.1}$$

$$= \frac{\sum_{i=1}^{n} d_i^2}{2t} \tag{A.2}$$

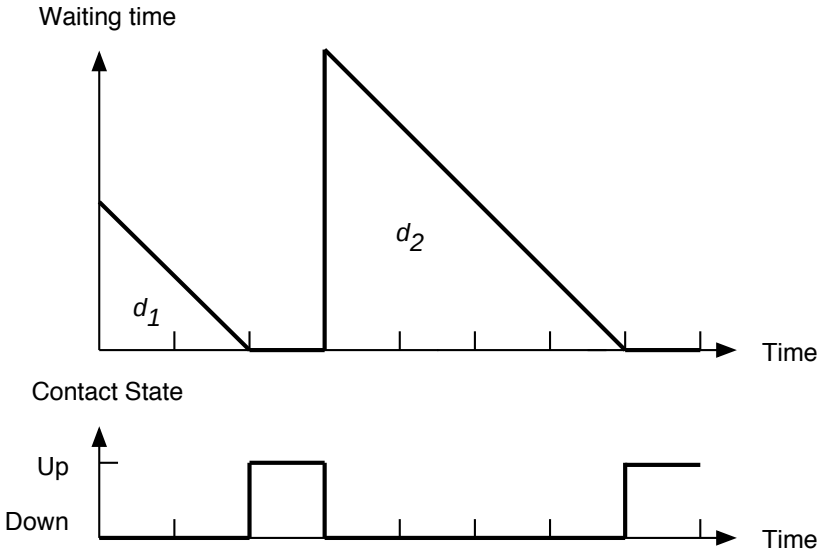


Figure A.1: Example contact waiting time and state

Bibliography

[1] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6–28, 2004. [Online]. Available: http://dx.doi.org/10.1109/MWC.2004.1368893

[2] E. Brewer, M. Demmer, B. Du, M. Ho, M. Kam, S. Nedevschi, J. Pal, R. Patra, S. Surana, and K. Fall, "The case for technology in developing regions," *IEEE Computer*, vol. 38, no. 6, pp. 25–38, May 2005. [Online]. Available: http://dx.doi.org/10.1109/MC.2005.204

[3] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networking," in *Proceedings of IEEE INFOCOM*, April 2006.

[4] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-tolerant networking: an approach to interplanetary internet," *IEEE Communications Magazine*, vol. 41, no. 6, pp. 128–136, June 2003. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2003.1204759

[5] F. Cristian and C. Fetzer, "The timed asynchronous distributed system model," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 10, no. 6, pp. 642–657, June 1999. [Online]. Available: http://dx.doi.org/10.1109/71.774912

[6] The dartmouth wireless trace archive. Dartmouth College. [Online]. Available: http://crawdad.cs.dartmouth.edu/

[7] J. A. Davis, A. H. Fagg, and B. N. Levine, "Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks," in *Proceedings of Int'l Symp. on Wearable Computers (ISWC'01)*, October 2001, pp. 141–148. [Online]. Available: http://dx.doi.org/10.1109/ISWC.2001.962117

[8] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," *SIGOPS Operating Systems Review*, vol. 22, no. 1, pp. 8–32, January 1988. [Online]. Available: http://dx.doi.org/10.1145/43921.43922

[9] K. Fall, "A delay–tolerant network architecture for challenged internets," in *Proceedings of ACM SIGCOMM*, August 2003, pp. 27–34. [Online]. Available: http://doi.acm.org/10.1145/863955.863960

[10] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, pp. 374–382, April 1985. [Online]. Available: http://dx.doi.org/10.1145/3149.214121

[11] R. H. Frenkiel, B. R. Badrinath, J. Borres, and R. D. Yates, "The infostations challenge: balancing cost and ubiquity in delivering wireless data," *IEEE Personal Communications*, vol. 7, no. 2, pp. 66–71, 2000. [Online]. Available: http://dx.doi.org/10.1109/98.839333

[12] R. Gass, J. Scott, and C. Diot, "Measurements of 802.11 in-motion networking," in *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, April 2006. [Online]. Available: http://www.cambridge.intel-research.net/haggle/pubs/irc-tr05050-inmotion.pdf

[13] M. Grossglauser and D. N. C. Tse, "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 477–486, August 2002. [Online]. Available: http://dx.doi.org/10.1109/TNET.2002.801403

[14] R. Handorean, C. Gill, and G.-C. Roman, "Accommodating transient connectivity in ad hoc and mobile settings," *Lecture Notes in Computer Science*, vol. 3001, pp. 305–322, January 2004. [Online]. Available: http://springerlink.metapress.com/link.asp?id=y049w219f7hwpldy

[15] K. A. Harras and K. C. Almeroth, "Transport layer issues in delay tolerant mobile networks," in *Proceedings of IFIP-TC6 Networking*, May 2006.

[16] K. A. Harras, K. C. Almeroth, and E. M. Belding-Royer, "Delay tolerant mobile networks (DTMNs): Controlled flooding schemes in sparse mobile networks," in *Proceedings of IFIP-TC6 Networking*, vol. 3462, May 2005, pp. 1180–1192. [Online]. Available: http://dx.doi.org/10.1007/11422778_95

[17] A. A. Hasson, D. R. Fletcher, and D. A. Pentland, "A road to universal broadband connectivity," in *Proceedings of Development by Design (dyd02)*, December 2002. [Online]. Available: http://www.thinkcycle.org/tc-filesystem/?folder_id=37675

[18] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket switched networks and human mobility in conference environments," in *Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN'05)*, August 2005, pp. 244–251. [Online]. Available: http://dx.doi.org/10.1145/1080139.1080142

[19] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks," in *Proceedings of ACM SenSys*, November 2004. [Online]. Available: http://dx.doi.org/10.1145/1031495.1031512

[20] S. Jain, M. Demmer, R. Patra, and K. Fall, "Using redundancy to cope with failures in a delay tolerant network," in *Proceedings of ACM SIGCOMM*, October 2005, pp. 109–120. [Online]. Available: http://dx.doi.org/10.1145/1080091.1080106

[21] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *Proceedings of ACM SIGCOMM*, vol. 34. ACM Press, October 2004, pp. 145–158. [Online]. Available: http://dx.doi.org/10.1145/1015467.1015484

[22] E. P. C. Jones, L. Li, and P. A. S. Ward, "Practical routing in delay-tolerant networks," in *Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN'05)*, August 2005, pp. 237–243. [Online]. Available: http://portal.acm.org/citation.cfm?id=1080139.1080141

[23] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet," in *Proceedings of ASPLOS-X*, October 2002, pp. 96–107. [Online]. Available: http://dx.doi.org/10.1145/605397.605408

[24] H. Jun, M. H. Ammar, and E. W. Zegura, "Power management in delay tolerant networks: A framework and knowledge-based mechanisms," in *Proceedings of IEEE SECON*, September 2005. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1557095

[25] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proceedings of ACM MobiCom*, August 2000, pp. 243–254. [Online]. Available: http://dx.doi.org/10.1145/345910.345953

[26] J. Lebrun, C.-N. Chuah, D. Ghosal, and M. Zhang, "Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks," in *Proceedings of IEEE Vehicular Technology Conference (VTC)*, vol. 4, May 2005, pp. 2289–2293. [Online]. Available: http://dx.doi.org/10.1109/VETECS.2005.1543743

[27] J. Leguay, T. Friedman, and V. Conan, "Evaluating mobility pattern space routing for DTNs," in *Proceedings of IEEE INFOCOM*, April 2006.

[28] Q. Li and D. Rus, "Sending messages to mobile users in disconnected ad-hoc wireless networks," in *Proceedings of ACM MobiCom*, August 2000, pp. 44–55. [Online]. Available: http://dx.doi.org/10.1145/345910.345918

[29] A. Lindgren, A. Doria, and O. Scheln, "Probabilistic routing in intermittently connected networks," *Lecture Notes in Computer Science*, vol. 3126, pp. 239–254, January 2004. [Online]. Available: http://springerlink.metapress.com/link.asp?id=9xt3904hd05fxmjf

[30] D. Marasigan and P. Rommel, "MV routing and capacity building in disruption tolerant networks," in *Proceedings of IEEE INFOCOM*, vol. 1, March 2005, pp. 398–408. [Online]. Available: http://dx.doi.org/10.1109/INFCOM.2005.1497909

[31] M. Mauve, A. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *IEEE Network*, vol. 15, no. 6, pp. 30–39, 2001. [Online]. Available: http://dx.doi.org/10.1109/65.967595

[32] M. Musolesi, S. Hailes, and C. Mascolo, "Adaptive routing for intermittently connected mobile ad hoc networks," in *Proceedings of IEEE WoWMoM*, June 2005, pp. 183–189. [Online]. Available: http://dx.doi.org/10.1109/WOWMOM.2005.17

[33] D. Nain, N. Petigara, and H. Balakrishnan, "Integrated routing and storage for messaging applications in mobile ad hoc networks," *Mobile Networks and Applications (MONET)*, vol. 9, no. 6, pp. 595–604, December 2004. [Online]. Available: http://dx.doi.org/10.1145/1035715.1035720

[34] T. S. E. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *Proceedings of IEEE INFOCOM*, vol. 1, June 2002, pp. 170–179 vol.1. [Online]. Available: http://dx.doi.org/10.1109/INFCOM.2002.1019258

[35] J. Ott and D. Kutscher, "Drive-thru internet: IEEE 802.11b for "automobile" users," in *Proceedings of IEEE INFOCOM*, March 2004, pp. 362–373. [Online]. Available: http://dx.doi.org/10.1109/INFCOM.2004.1354509

[36] R. D. Poor, "Gradient routing in ad hoc networks," 2000, MIT Media Laboratory, *unpublished manuscript*. [Online]. Available: http://www.media.mit.edu/pia/Research/ESP/texts/poorieeepaper.pdf

[37] A. Rabagliati. (2004, April) Wizzy digital courier – how it works. [Online]. Available: http://www.wizzy.org.za/article/articleprint/19/1/2/

[38] A. Seth, P. Darragh, S. Liang, and S. Keshav, "An architecture for tetherless communication," June 2005, University of Waterloo, *unpublished manuscript*. [Online]. Available: http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/05/tca.pdf

[39] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: modeling a three-tier architecture for sparse sensor networks," in *Proceedings of Sensor Network Protocols and Applications*, May 2003, pp. 30–41. [Online]. Available: http://dx.doi.org/10.1109/SNPA.2003.1203354

[40] 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)," in *Proceedings of ACM MobiHoc*, June 2003, pp. 233–244. [Online]. Available: http://doi.acm.org/10.1145/778415.778443

[41] ——, "Resource and performance tradeoffs in delay-tolerant wireless networks," in *Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN'05)*, August 2005, pp. 260–267. [Online]. Available: http://dx.doi.org/10.1145/1080139.1080144

[42] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Single-copy routing in intermittently connected mobile networks," in *Proceedings of Sensor and Ad Hoc Communications and Networks (SECON)*, October 2004, pp. 235–244. [Online]. Available: http://dx.doi.org/10.1109/SAHCN.2004.1381922

[43] ——, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN'05)*, August 2005, pp. 252–259. [Online]. Available: http://dx.doi.org/10.1145/1080139.1080143

[44] K. Tan, Q. Zhang, and W. Zhu, "Shortest path routing in partially connected ad hoc networks," in *Proceedings of Global Telecommunications Conference (GLOBECOM)*, vol. 2, December 2003, pp. 1038–1042. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1258396

[45] Intelligent transportations systems. University of Washington. [Online]. Available: http://www.its.washington.edu/

[46] Dtnsim2 dtn simulator. University of Waterloo. [Online]. Available: http://shoshin.uwaterloo.ca/dtnsim2/

[47] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Duke University, Tech. Rep. CS-2000-06, July 2000. [Online]. Available: http://issg.cs.duke.edu/epidemic/epidemic.pdf

[48] R. Y. Wang, S. Sobti, N. Garg, E. Ziskind, J. Lai, and A. Krishnamurthy, "Turning the postal system into a generic digital communication mechanism," in *Proceedings of ACM SIGCOMM*, 2004, pp. 159–166. [Online]. Available: http://dx.doi.org/10.1145/1030194.1015485

[49] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure-coding based routing for opportunistic networks," in *Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN'05)*, August 2005, pp. 229–236. [Online]. Available: http://dx.doi.org/10.1145/1080139.1080140

[50] J. Widmer and J.-Y. Le Boudec, "Network coding for efficient communication in extreme networks," in *Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN'05)*, August 2005, pp. 284–291. [Online]. Available: http://dx.doi.org/10.1145/1080139.1080147

[51] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *Proceedings of ACM MobiHoc*. New York, NY, USA: ACM Press, May 2004, pp. 187–198. [Online]. Available: http://dx.doi.org/10.1145/989459.989483

[52] ——, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in *Proceedings of IEEE INFOCOM*, vol. 2, April 2005, pp. 1407–1418 vol. 2. [Online]. Available: http://dx.doi.org/10.1109/INFCOM.2005.1498365