Delay-Tolerant Networks with Network Coding: How Well Can We Simulate Real Devices?

by

© Xu Liu

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of Science

Department of Computer Science

Memorial University of Newfoundland

March 2014

St. John's Newfoundland

Abstract

Delay-tolerant networking effectively extends the network connectivity in the time domain, and endows communications devices with enhanced data transfer capabilities. Network coding on the other hand enables us to approach the information capacity of networks by allowing intermediate nodes to process data en route. Both of these were major principal breakthroughs in mobile and wireless communications in the past decade or so. In the first half of this thesis, we consider the problem of disseminating a large number of messages in such networks. With the sparse and intermittently connected topology and with the unreliable and low-rate radios, the strategy of which messages to transfer first and in what order is a determinant of performance here. We compare a few such message prioritization methods using computer simulation and observe their performance in terms how widely and quickly information can be distributed across the network. Next, we are interested in how network coding stacks against conventional epidemic routing variants. We conducted tests with both real smart mobile devices and computer simulation and found conditions where their results match. This would give us confidence of using computer simulation to study larger delay-tolerant networks with and without network coding at a much manageable cost.

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Yuanzhu Chen, for dedicating himself fully into guiding me during my masters program in Canada. He invested a lot of his time not only into training me to be a good researcher, but he also encouraged me to keep going during academic struggles. He was the one who inspired me to enter the world of computer network research and product development. I would not be where I am right now without his all-around support. Furthermore, Memorial University and Department of Computer Science offered me the most precious opportunity of pursuing my post-graduate studies, and I have benefited tremendously from this lovely environment.

I would also like to thank Dr. Jason Moore, Dr. Cheng Li, and Dr. Walter Taylor. They provided me with a lot of support during my research project, including help with defining a research project idea. In addition, they contributed a lot by providing enough experimental data for my research project.

In addition, I would like to thank my friend Esteban Ricalde. He was my first friend I met in Canada and he truly became a friend for life. He has been an inspiration and support for me from the very beginning of my master degree. Also, I would like to thank my friends Anastasia Gurinovich and Wasiq Waqar. Your friendship helped me to get through my master degree in a foreign country.Moreover, I would like to thank my friends Sahand Seifi and Bahar Prar. They organized many gatherings of our friend circle, especially I enjoyed the New Years Eve party.

I would also like to mention that I am very grateful to Dr. Chen's NSERC Discovery Grants and Dr. Moore's NIH grants. This funding relieved the financial burden from me, so I could work on my research more and enjoy my school life without additional financial struggles.

A very special thanks to my wife Chengling Huang and my parents. Their support and understanding gave me all the strength to move forward during my master degree and my life in general. Chengling has spent many days and nights staying up with me and encouraging me to keep going during difficult times.

Moreover, I would like to thank my cousins Yanxiang Su and Zhihua Mao. Their suggestions helped me overcome several difficult moments during my university life.

Also, I want to thank Dr. Qing Li who introduced me to my great supervisor and mentor for life Dr. Yuanzhu Chen.

Finally, I would like to thank all of my friends who supported me in writing, and encouraged me to achieve my goal.

Contents

List of Figures

Chapter 1

Introduction

1.1 Mobile computing today

Since the invention of modern digital computers in 1946, the proliferation of this technology has impacted every aspect of people's life profoundly. The way computers are operated experienced four major paradigms. The initial batch systems would allow a computer user to submit a batch of jobs and retrieve the results at a later time. In the next paradigm, represented by real-time operating systems, the user is able to interact with computers for much higher productivity. Till this time, accesses to computers were mostly limited to university personnel and scientific researchers. Started from the 1980's, as computing hardware became more capable and cheaper to manufacture, computers became household appliances and started the age of Information Technology. With the privatization of the Internet, networked personal

computers fostered disruptive technologies in many sectors, such as telecommunication, e-commerce, media publishing, entertainment, social studies, education, etc. The most recent paradigm shift, however, is happening right now from personal computing to mobile computing.

With further rapid development of computing technologies, a small device such as a smart phone nowadays is packed more computing power than an Apollo Spaceship. In addition, the communication connectivities on such devices are diversified, fast, and energy-efficient. Combined with the high availability of cellular communication and Wi-Fi infrastructure, smart phones and tablet computers allow us to to accomplish many things wherever and whenever we want to. On one hand, these include porting day-to-day applications, such as e-mail, web, video conversation, shopping, multimedia streaming, and gaming to mobile devices and merge them with conventional telephone and short message services on a single device, making these devices a true digital companion of people's life. On the other hand, mobile devices provide other values that are not possible on personal computers. They add a whole array input sources, including location, motion, sound, light, air pressure, and so on. To facilitate innovation on these devices, mobile operating systems, such as the Google Android OS and Apple iOS, often open a large number of programming interfaces to the third-party. The ecological systems thus cultured have not only advanced the state of the art, but also created new business models for economic growth.

Innovations brought about my mobile computers do not stop here. In fact, the

exploration of using the on-board short-range radios, such as Wi-Fi and Bluetooth, to construct *multi-hop* wireless networks [30] has been ongoing for a couple of decades. What used to be only feasible on personal computers is now becoming possible on smart phones and tablet computers. Such a networking technology is also called *mobile ad hoc networking* and *wireless mesh networking*, depending on what aspect we emphasize on. Its application is to fill in the void of infrastructure as a result of natural disaster or to be deployed to a geographical area where there is no such infrastructure at all. Disaster relief, emergency response, personnel rescue, just to name a few, are what mobile devices would prove to be superior to personal computers.

1.2 Mobile computing tomorrow

The research community of mobile computing has been exploring for more creative ways to facilitate cutting-edge or even unconventional ways of data communication. Below are two of such streams of research relevant to this thesis.

1.2.1 Delay-tolerant network

Delay tolerant networking (DTN) is a recent, new data transfer model that expands the network connectivity from the spatial domain to the temporal domain. In contrast to the "store-forward" paradigm of data transfer in the Internet or mobile ad hoc networks, DTN utilizes the mobility of users in a very sparse network to "store-carry-forward" data packets. It effectively connects two nodes that have never had

an end-to-end paths in between during the entire operation of the network, as long as the delay thus introduced is acceptable to the users and the application requirements.

Research on DTN started from the Interplanetary Networking project at JPL [11]. The networking problem in such a scenario considers predictable mobility of space probes and surface stations, where the feedback loop can take a very long time to complete due to both signal propagation delay and obstacles of other celestial bodies. In a more general setting, the mobility of communication devices is unpredictable, so scheduling networking activities in a deterministic fashion is no longer feasible. A great deal of research has been done on data transfer in such a framework to fulfill the simple goal of moving data from the source to its destination. A number of excellent reviews and vision articles have been published on the architecture and protocol aspects of delay-tolerant networks [21, 16, 29, 30].

The roaming of mobile devices their users provides an ideal scenario for studying and applying DTN. It is just a matter of time that such a flexible way of operating a wireless network would become the next wave of innovation.

1.2.2 Network coding

The concept of network coding was formulated in the seminal work of Ahlswede, Cai, Li, and Yeung [19] in 2000, and the past decade has seen a tremendous momentum in this area [17]. Its idea breaks the principal of traditional multi-hop networking, where intermediate nodes only forward packets but do not modify their contents, much like

cars traveling on a highway. Since bits are not cars anyway, network coding allows intermediate nodes to combine packets from different input ports before forwarding them. When treating a packet as a sequence of symbols, even linear network coding defined over small Galois fields can introduce significant throughput gain. The readers are referred to an easy-to-read and yet informative primer by Fragouli, Le Boudec, and Widmer [7]. Other benefits of network coding include improved robustness of network operations, higher energy efficiency in wireless radios, and better security against eavesdroppers. Network coding proves to be especially powerful and flexible, and can be exercised along with other revolutionary networking paradigms. For example, it was shown that opportunistic data forwarding in multi-hop wireless networks can further increase the capacity of these networks when intermediate nodes strategically combine overheard packets and forward them [22, 12]. As another example, the resilience to lost or delayed information brought about by network coding turns out particularly effective in DTNs, as evidenced by computer-simulated experiments in Widmer and Le Boudec [14].

With the computing power packed on current smart phones and tablet computers, we have a choice of trading some of this capability for bandwidth. This is even more promising if we use the graphics hardware on these devices for their highly efficient matrix computation capacity.

1.3 This thesis

This thesis reports scientific research on how network coding improves the performance of a DTN. The primary tools we use are computer simulation and using commodity mobile devices. It attempts to address the following research problems.

1. In a DTN, the opportunity that two nodes are within range of each other can be rare and is often unpredictable. In a short window of contact, a node must decide which messages that it is currently carrying should be exchanged with the other node first. Apparently, this decision affects the network performance significantly, and we are interested in finding what approach of prioritizing messages yields the best results.

2. We rely on computer simulation in validating our proposals, and use the ONE [5] simulator in our study as many of our colleagues in DTN research. However, the ONE needs to be modified extensively to suite our needs. For example, we need link-layer support of packet broadcast, a link model with certain packet-error rates, and implementation of network coding. With such modifications, the simulator is closer to how we would operate a DTN with network coding using real devices. We are curious how well the simulator resembles the real world and under what conditions.

3. With a better tuned simulator, we are more confident in testing new data transfer methods in DTN with it. In particular, we propose a handshake proceeding

the actual coding and transfer of packets so that network coding can be better tailored to the specific needs to a neighborhood of nodes rather than being completely uninformed. We expect that the modified simulator to be able to tell us how such our proposed method improves the vanilla network coding.

The rest of this thesis is organized as follows. Next chapter, we provide necessary background in fundamentals of delay-tolerant networking and network coding, and in the basics of the ONE simulator. In Chapter 3, we study four message prioritization methods in the framework of Epidemic Routing [3]. Results reported therein are obtained with a modified version of the ONE. In Chapter 4, we further investigate the effect of adding network coding to the ONE. More importantly, we compare results with experimenting with real Apple iOS devices. We summarize how the simulator should be fine-tuned for a better match with the real world. Next, we propose an improvement of the original network coding approach in DTN in Chapter 5. That is, we allow nodes to exchange some metadata for an informed decision on what packets are to be coded and exchanged subsequently. Using the modified ONE simulator, we show a noticeable improvement in the data transfer capabilities of the network with a reduced overhead. The thesis is concluded in Chapter 6 with a discussion of how our research can be extended in some interesting ways.

Chapter 2

Background

2.1 DTN Routing

2.1.1 DTN routing overview

Delay tolerant network is an intermittently connectivity environment. In this kind of network, people can exchange non-time-sensitive data between different devices. The characteristics of DTN make us have no stable and fixed routing path to deliver data. Therefore, mobile nodes in such a network must make intelligent decisions to route data with a possibly long delay. Nowadays, many routing protocols are put forward by DTN researchers to address this issue. We summarize them as below.

1. In order to successfully deliver a set of messages to the destination in a delay tolerant network, a naive approach is to use broadcasting. This is a method

that a node forwards messages to all nearby nodes without any restrictions. The representative routing algorithm is called Epidemic Routing [3]. It uses a three-way handshake between two nodes before they decide to exchange real data. After that, based on the knowledge obtained during the handshake, two nodes begin the message exchange process. This routing protocol can effectively distributes messages to all nodes in the network. However, it may have a large operation overhead. Therefore, it is necessary for us to improve Epidemic Routing to reduce the overhead.

2. The Spray and Wait routing [23] is an efftective solution to the flooding problem by reducing the copies of a message and the depths of the hops. It includes two phrase: "spray" and "wait". In the spray phrase, the sources is responsible for creating several copies of a message and distribute it to the network. In the wait phrase, the node holds a copy of a message and get ready to delivery it once it meets the destination. By using the binary version of Spray and Wait routing, every node transmits half number of copies of a message. Then, the next node repeats the same process and give half of copies of this message to its future encountering node until there are only one copy left. Finally, the node which holds the single copy deliveries the message only to the destination. As we can see, compared to Epidemic Routing, Spray and Wait controls the copies of messages. It reduces the overhead and workload for the network.

3. Although Spray and Wait routing reduces the flooding effects from epidemic

routing, its decision is still somewhat arbitrary. Therefore, it is necessary for us to look for routing protocols which can delivery packets more selectively.

(a) MaxPro [13] is one of such protocols. Different from epidemic routing, it deliveries packets from an ordered list which is calculated by the likelihoods of a packet reaching a potential destination. Specifically, each node maintains a vector with several probability values for reaching the other nodes. When two nodes meeting each other, they exchange their vectors. Then, during the process of moving, nodes create a matrix for all nodes from these vectors and calculate the possible path from a source to destination. Last, the nodes send packets to the order list based on the least sum of total likelihood of all possible paths. As we can guess, compare to epidemic routing and spray and wait routing, this behavior can potentially increase the average delivery ratio and decrease the average delay time for all the messages due to its selective sending feature.

(b) PRoPHET [4] has a similar idea as MaxPro. It is a also probabilistic routing protocol using history of encounters and transitivity. Rather than assuming a totally random mobility pattern, PRoPHET supposes that in real world, the movement of nodes more or less has a certain profile. By predicting the movement pattern from the history of encounters, PRoPHET delivers messages among each nodes. This protocol allows each node in the network to maintain a table which records the deliv-

ery predictabilities $P(M, D)$ where M is the current node and D is the destination. By adding effect from current delivery predictability to the previous value $P(M, D)$ based on several rules, the node updates this delivery predictabilities table and deliver messages selectively. This protocol efficiently decides productivity paths or nodes for a certain amount messages based on the probability. It performs better than Epidemic Routing and Spray and Wait routing.

(c) DTN routing can be treated as a resource allocation problem. e.g. RAPID [6] RAPID uses a utility function to calculate a utility value U_i to every packet i in terms of the optimization of three metrics: average delay, missed deadlines, and maximum delay. At the beginning, nodes exchange metadata between each other in order to assist each packet to calculate their own utility values. Then, based on the utility value, it orders these packets in the node buffer. Next, a node replicates the packet based on a certain condition in the ordered buffer and injects it in the network. Last, the node stops injecting packets when all packets have been replicated or when the contacts to other node break.

2.1.2 Epidemic routing and message prioritization

The detailed steps of Epidemic Routing [3] are as below. when two nodes (or devices) come into transmission range of each other, they conduct a 3-way handshake for one

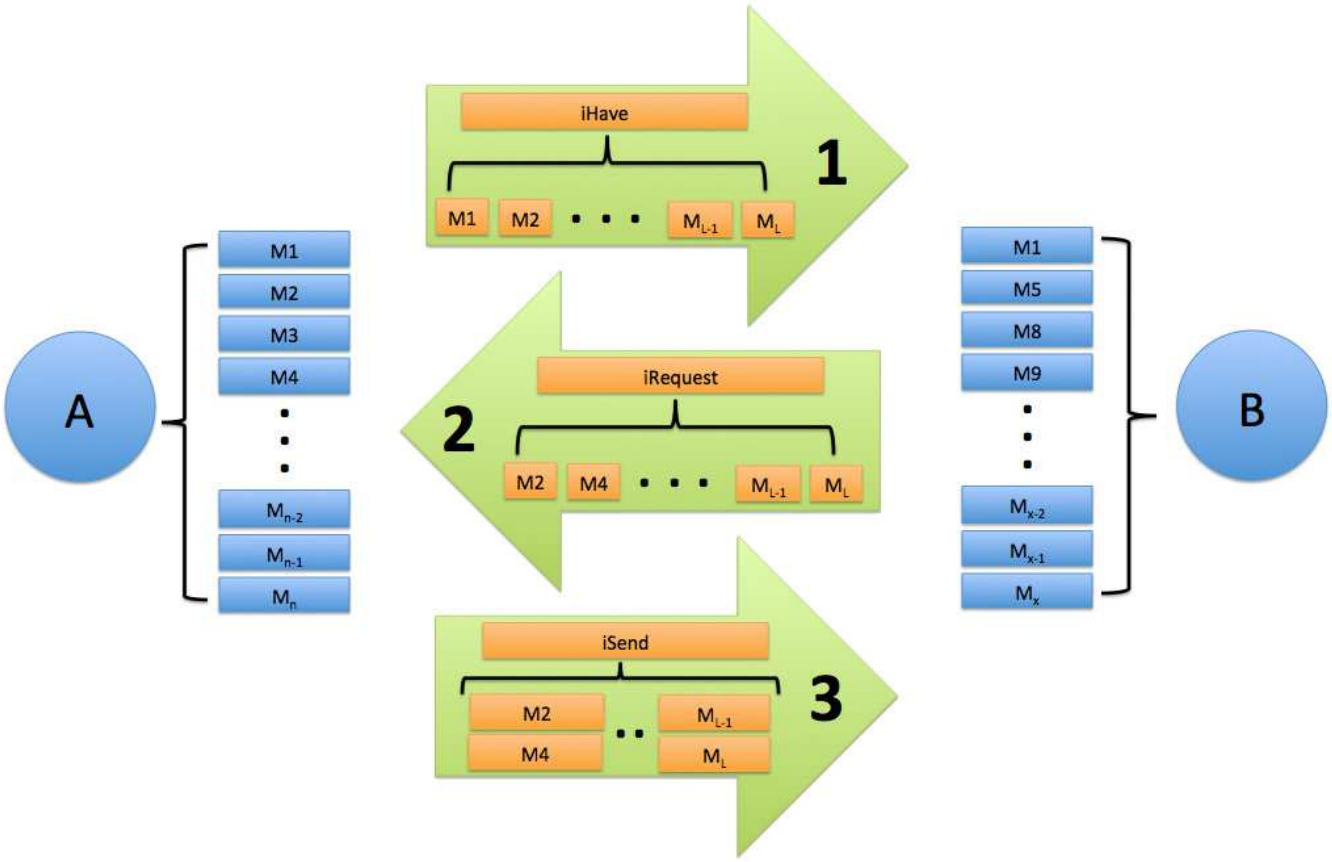

Figure 2.1: Steps for nodes to start to messages exchange

device to send messages to the other (Figure 2.1). In particular, when nodes A and B discover each other, node A sends an array of the IDs of all of its known messages to B in an `iHave` packet. After receiving this array and comparing it to its own set of buffered messages, node B replies with a subset of these message IDs as request indicating that these are the messages that A has but B does not in an `iRequest` packet. In the third message, `iSend`, node A sends the requested message to B. This process is triggered every time two nodes come close to each other. Epidemic routing can be a foundation for fulfilling both unicast and broadcast data transfer services. Apparently, given that nodes typically store more messages than that can fit in a single handshake packet, the strategy taken to include which messages in the advertisement and in what order affects the system performance significantly. We call such a strategy message prioritization, and it is investigated in the next chapter of this thesis.

2.2 Some DTN Projects

Here, we name a few projects using the DTN technology.

1. *Haggle* [8] is one of the most poplar platforms that offers people a handy way to produce applications and conduct experiments. It was funded by the Europe Commission and is treated as a research project in Autonomic Opportunistic Communication. From 2006 to now, five European countries have joined this project to maintain and improve it. Haggle has a great vitality due to the

good technical support from these strong research groups. As for the implementation, Haggle concentrates on data exchange among mobile devices by either Wi-Fi or Bluetooth in challenged networks. Mobile operation systems and platforms (such as Android OS, iOS, and J2ME) are the main supported environments, although it can run in personal computer operation systems such as Windows, Linux, Mac OS and Unix. Moreover, Haggle has a feature, called "Content Sharing". This feature creates a virtual bridge to connect Haggle users who have the similar interests and profiles. Haggle user only needs to set up the personal interest keywords. Then, he/she will automatically receive news, pictures, and messages that match their individual profiles.

2. *DTN2* is another DTN platform for conducting experiments on Linux and Mac OS X. Because it was originally created in 2005 by the Delay Tolerant Networking Research Group (DTNRG) [1, 2], it combines several robust and flexible tools designed especially for DTN research rather than application developers. Moreover, from its official website, we can see that the APIs and program tutorials are not detailed enough for some elementary programmers and DTN amateurs to learn. Potentially, proficiently using this platform requires high programming skill, and deep understanding of some knowledge related to DTN.

3. *PodNet* [24] is also a DTN platform which concentrates on mobile distribution of user-generated contents. It is a both experimental platform and application development platform. A research group in Sweden created this platform

in 2008. The key points and technical details related to this platforms were initially published in International Conference on Mobile System [25]. This platform has similar features to Haggle [8], but without sources code.

4. In contrast, *DoDWAN* [9], Document Dissemination in mobile Wireless Ad hoc Networks, is a relatively open and flexible DTN platform for us to create applications and conduct experiment, which is produced by a research group in France in 2005. It combines two ideas from two papers published in 2007 and 2010 respectively [10, 15]. It is a Java-based platform with some development documents. As with the Haggle project, DoDWAN is a good choice for DTN experiments and application development.

2.3 Network Coding Primer

Network coding [7] is a quite new technique for efficiently exchanging messages in Delay Tolerant Networks among other things. It has inspired a large number of researchers to produce exciting results. Its basic idea is to combine a set of messages in a node storage space,possibly linearly, as a single packet before transmitting it. This process is called "encoding". Once, any device receives such encoded packets, they are inserted into a decoding matrix. Network coding allows information to be diffused in the network guidely and is very resilient to link failure.

Encoding is a process in network coding to combine a set of messages which are

generated by one or several sources before transmitting as a single packet. Let us assume that there are n original messages in the network, denoted$M_1, M_2, \ldots, M_n$. All exchanged packets are generated from these original messages with coefficients $g = \{g_1, g_2, \ldots, g_n\}$ in F_{2^e}, where g is called *encoding vector* and F_{2^e} is a Galois field. Both symbol M^i and coefficient g_i are taken from F_{2^e}. Finally, the generated packet equals to $X = \sum_{i=1}^{n} g_i M^i$, we call it *information vector* [18].

Often is sufficient to use bit-wise operation, so we can set $F_{2^e} = F_2$, namely, $g_k = \{0, 1\}$, where 0 means k^{th} M is not combined in this packet while 1 means k^{th} M is in the combined packet. After that, we include encoding vector and the information vector in one packet and send it to the network.

When a combined packet X arrives at a node, the node decodes for the original messages by inserting this combined packets to a matrix and solves it with Gaussian elimination. This assume that the rank of the decoding matrix is full. Otherwise, the receiving node only needs to maintain a reduced echelon form.

2.4 The ONE Simulator

The ONE (Opportunistic Network Environment) simulator is a research tool for conducting experiments on Delay Tolerant Networking [5]. It gives the users the power to extend movement patterns, routing protocols, rules of generating messages, and so on. Compared to NS-2 [20], the learning curve of ONE is relatively smooth for a beginners, and its configuration is not so difficult. Therefore, in our research,

we chose to use ONE simulator.

2.4.1 Framework

The ONE is written in Java and provides both a graphic user interface and a batch mode interface for user to conduct experiments. Moreover, it offers us source codes of some known DTN routing protocols, such as spray and wait [23], MaxProp [13], PRoPHET [4], and so on. Based on these implementations, researchers can easily understand the main logic of some DTN routing algorithms and improve it. In our research, we use version 1.4.1 of ONE due to no newest version of ONE available when we began this research.

2.4.2 Features

Here, we summarize three main features of this simulator: *Mobility Management Component, Event Management Component, Node Information Management Component* as shown in Figure 2.2.

- *Movement Management Component*: This component is used for the simulator to move nodes in a certain area. It originally includes several well known movement patterns, such as random waypoint, map-based movement, cluster movement, and so on. By changing the value of property "movementModel" in ONE configuration file, users can easily use any movement pattern.

Specifically, the map-based movement offers a very flexible structure for users to dynamically import or create their own maps. It uses a so called *Well-Known Text* (.wkt) file to define a map with different paths. It is defined by the Open Geospatial Consortium (OGC) to render vector geometry objects. In ONE, it has already included many wkt files for us to use, such as roads, main roads, pedestrian paths, shops and so on of various cities in the world. In our experiments, we mainly use the WKT Markup Language – "LineString' or "MultiLineString" to describe our own movement paths based on the geometry of Engineering Building of Memorial University in Figure 4.1.

- *Event Management Component*: In order to handle a series of events in the simulation, ONE offers an Event Management Component. This component processes each event by fetching it in an event queue. We can schedule events, such as message generating, network topology snapshotting, network messages statistics, nodes connections, and so on, to achieve global control in a simulation. Also, we can dynamically change the properties for each node in the component during the simulation. The ONE simulator provides us various types of events in order to help us to customize events for our own research needs.

- *Node Information Management Component*: This is a rather complex component in ONE simulator. It is responsible for the message exchange behaviors of each node. Originally, it gives us two basic approaches to manage messages

– Active Router and Passive Router. These two approaches only allow us to delivery and exchange messages without much control. In order to implement various features of routing protocols, we need to inherit one of the two routers and implement our own routing protocol. In the ONE, it provides us some well-known routing protocols such as epidemic routing, prophet routing, spray and wait routing and so on. By using, learning and modifying these included routing protocols in depth, we can construct our own routing protocol. After that, we can use its report system to generate a set of different formatted trace files for further analysis.

2.4.3 What else we need

- ONE version 1.4.1 does not have a physical layer mode, That is to say, we can not fully simulate packet collision. Therefore, packets loss and retransmission mechanisms for the routing protocols in ONE are not accurate.

- After studying the ONE simulator, it seems that the epidemic router in ONE does not fully implement the mechanism of three way handshake of Epidemic Routing. We need a true broadcast support Epidemic Routing in the simulator.

- In our research, we try to use ONE to conduct some experiments using network coding. However, this feature was not supported by the simulator. We need to implement network coding ourselves

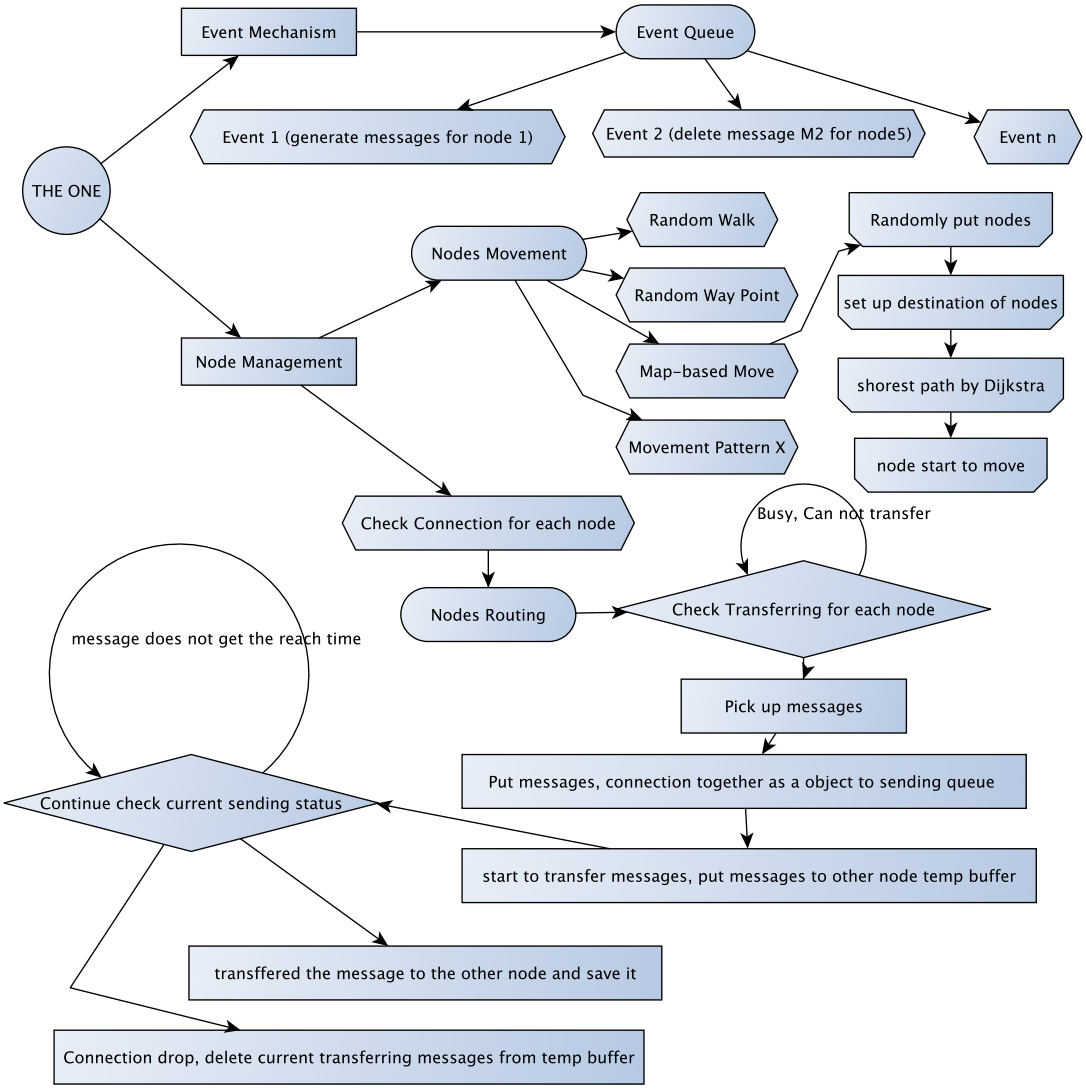

Figure 2.2: Components of ONE simulator

- The reporting system of the ONE simulator needs to be further enhanced for us to analyze protocol execution, such as packet type, outcome of a transmission and reception, state position of a node during a transmission , etc.

Chapter 3

Message Prioritization in Epidemic Routing

This chapter reports computer simulation results on four message prioritization methods of Epidemic Routing. These results are also published in our ICNC conference article [27].

3.1 Methods of Message Prioritization

Recall that, for Epidemic Routing ([3] and Section 2 of this thesis), when two nodes (or devices) come into transmission range of each other, they conduct a 3-way handshake for one device to send messages to the other. In particular, when nodes A and B discover each other, node A sends an array of the IDs of all of its known messages

to B. After receiving this array and comparing it to its own set of buffered messages, node B replies with a subset of these message IDs as request indicating that these are the messages that A has but B does not. In the third message, node A sends the requested message to B. This process is triggered every time two nodes come close to each other. Apparently, epidemic routing can be a foundation for fulfilling both unicast and broadcast data transfer services.

Here, we study the particular operation of information dissemination (i.e. message broadcasting) in DTNs. We do not assume any temporal self-similarity in the mobility of devices, so we do not rely on extrapolating previous contact information. We focus on the case where the number of buffered messages is so large that any form of compact representation of all stored messages, such as digest or ID, would not be accommodated in a single handshake packet. Therefore, when a node advertises about the messages it has received so far, it must pick strategically a subset of them to fit in a single packet. Such a strategy affects how quickly messages can be disseminated to a large number of nodes. We implemented the message dissemination framework with three prioritization approaches in our customized ONE simulator [5], and tested them with one baseline approach. We observe that a well designed message prioritization method can significantly expedite such a broadcast service in DTNs.

Apparently, given that nodes typically store more messages than that can fit in a single handshake packet, the strategy taken to include which messages in the

advertisement and in what order affects the system performance significantly. We call such a strategy message prioritization. In this work,we are interested in a few simple, and yet very different such methods. In all methods, we assume that node A fills the advertisement packet with l digests of some of its stored messages.

1. Round robin — A node maintains a FIFO queue of the messages it has received and generated so far, i.e. by the time it is injected into the network. It circulates through the queue to compile the message digests using a pointer. When it is about to initiate a handshake, it processes l messages and advance the pointer accordingly. Here, the node maintains separate pointers for different nodes. Note that as the system continues, the time it takes to finish a round becomes longer, and when it does, it starts from the head of the queue again.

2. Tiered — Messages stored at a node are ranked according to three quantities to favor new, short messages, i.e. forward history, age, and length. The fewer times it has been forwarded till reaching this node, the later it was created, and the shorter it is in length, it is ranked higher in the storage queue. These ranked messages are split into three segments of equal number of messages, the upper, middle, and lower tiers. Three separate round-robin schedules are executed on the tiers. The upper tier has three opportunities to send an advertisement containing l digests of its own, the middle tier has two, and the lower tier has one. As such, the system helps newly injected message to spread in the network more quickly.

3. Oblivious — A node maintains a FIFO queue of all messages by the time they are injected into the network as in Round robin. When the node needs to create an advertisement packet, it simply takes the last l messages in the queue. In this method, the node never looks back after it has past a message in the queue, thus, always rigidly favoring the latest messages in the system.

In addition to the three above, we also implemented a *random* prioritization method as a comparison baseline. In this approach, a node would randomly pick l messages and advertise their digests. All four methods have different ways to allocate opportunities to messages to be advertised in the network. We tested these different message prioritization methods to see how effective they are in helping messages to spread in the system.

3.2 Revisions to the ONE Simulator

The ONE provides us a great foundation for experimenting message prioritization. It is well-suited for DTN research and has a reasonably smooth learning curve. Nevertheless, we had to revise it in a few ways for our particular needs. These revisions are summarized below.

1. The ONE has an idealized link layer implementation, where a packet is always received as long as the sender and receiver are within range of each other. To make simulation closer to real environments, we introduced a parameter at the

link layer to control the packet error rate. Its default value is 50% regardless how far the two nodes are provided they are within range. Such a high error rate is used account for a harsh operation environment and also the fact that it does take the two nodes some time to create a link after they move close.

2. The implementation of Epidemic Routing in the ONE does not have the three-way handshake as described in Section 2.4.3 and [3]. Thus, we implemented the three way handshake mechanism by adding two new control packet types to ONE simulator, called `iHave` and `iRequest`.

3. The original simulator implements a broadcast of packet by a sequence of unicasts. We extended this by allowing multiple nodes to receive a packet at the same time in order to faithfully reflect how wireless channels are used on real devices.

4. The ONE has a succinct trace file format for post-test analysis. We enhanced its format extensively for us to be able to reconstruct how messages propagate through the network.

5. Last, we disabled the nodes from dropping packets when the buffer is full so that we can simulate scenarios with significant load and delay.

3.3 Simulation

We used the ONE to evaluate how different message prioritization methods affect the performance of the system. We measured the latency in transferring messages to the destinations and a variant of message delivery ratio. We observe that the Oblivious prioritization method is significantly superior to the other approaches despite its simple nature.

3.3.1 Settings

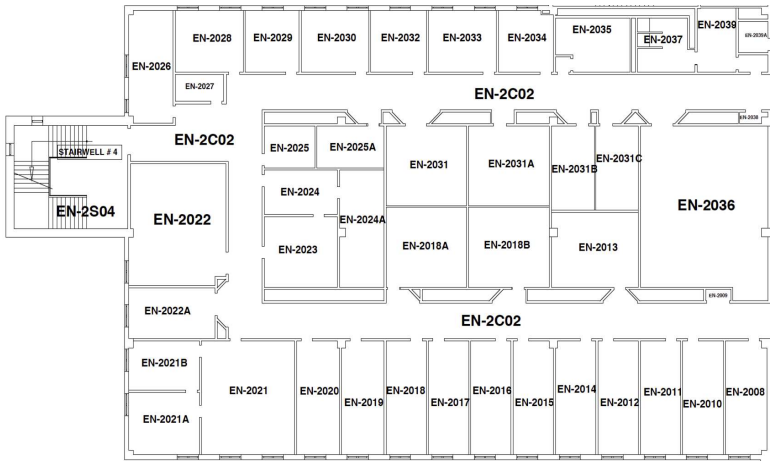

Figure 3.1: Engineering Building

We used the map mobility management of The ONE simulator, where a topological structure of the simulation area is used to specify how nodes can move around. During the simulation, a node can decide a destination position, such as an inter-

section or a specific point on an edge, and moves there via the shortest path at a certain velocity. When two nodes are within transmission range (set to 10 meters in simulation), they discover each other and start to transfer messages. The map that we used in our tests is part of the first and second floors of the Engineering Building at Memorial University of Newfoundland (Figure 3.1). We picked this particular venue because in a parallel project we implemented prototype applications on the iOS and Android OS so that we can compare the real and simulated test results in future.

We assumed using the Bluetooth 4.0 radios on the iOS devices. As such, the maximum size of a single packet in the handshake is limited to 90kB. Around every 400 seconds, a device injects a message of size uniformly distributed in $[2000, 5000]$ bytes. Parameter settings are summarized in Table 3.1.

3.3.2 Results

We are interested in how widely and quickly messages are disseminated in the network, measured in two quantities, i.e. extent and progression. After a message is generated, it is first stored at the originator, and as time goes on, it reaches more and more nodes. We observe how many other nodes a particular message has reached after d seconds, where d is called *delivery deadline* ($d = 3000$ in simulation). For a given message m and delivery deadline d, we denote the set of nodes in the network that m has reached after d other than the message originator itself by $O_{m,d}$. Thus,

Parameter	Value
number of nodes in network n	10, 20 or 30
total simulation time T	20,000 seconds
node movement velocity v	$0.5 \sim 1.5$ m/s
message generate rate per device t	every 400 seconds
message length s	$2,000 \sim 5,000$ bytes
number of digests in advertisement packet l	10 messages
interval of digest advertisement τ	every 150 seconds
transmission range r	10 meters
maximum packet length S	90 kB
delivery deadline d	3000 seconds

Table 3.1: Simulation parameters

the *extent* of message m is defined as $|O_{m,d}|$, i.e. how many other nodes the message has reached up till the deadline. We consider the messages injected during the first $14,400$ seconds of the entire $20,000$ seconds of simulation so that all messages would have sufficient time to be disseminated. For a network n nodes ($n = 10, 20$ or 30), $360 \times n$ messages are injected in total, collectively denoted by M. As such, we plot a histogram of the extent over M, for $n = 10, 20$ or 30 respectively, in Figures 3.6, 3.7 and 3.8. In all three figures, we can see that there is a behavioral difference between Oblivious and the other three. Specifically, Oblivious is able to spread the majority

of the messages to most of the other nodes while the other three have much smaller extents. The reason is that Oblivious outperforms the other three methods is that it persistently advertises the newest messages to boost their initial presence in the system. This is evidenced by Figure 3.2, where we plot the number of times that a message is placed in an advertisement packet in the simulation a 10-node network. We can observe that compared to the other methods, Oblivious is able to distribute the opportunities for messages to be advertised most equally, while the others are more or less skewed towards older messages.

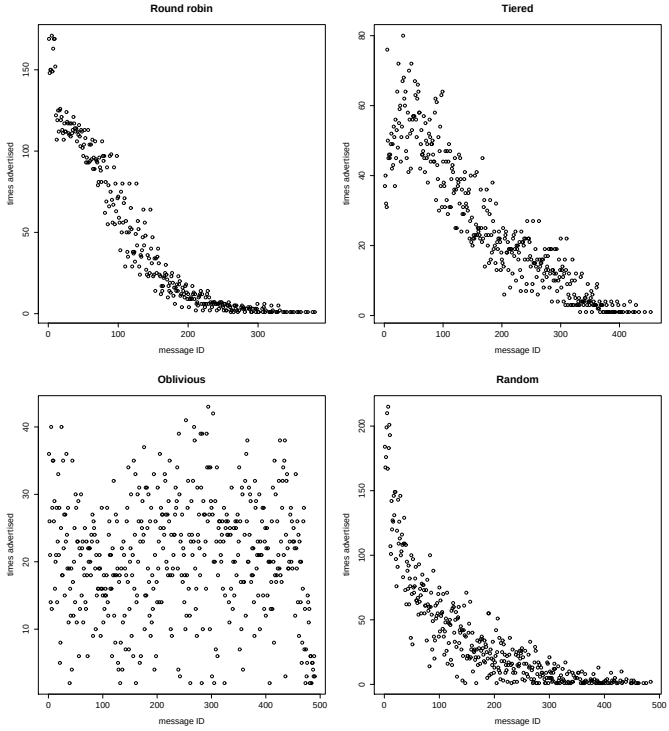

Figure 3.2: Number of times a message is advertised in 10-node network

Next, we turn our attention to how fast messages can be broadcast in the network using a generalized notion of latency, called *progression*. For a given message m in an n-node network, we use the vector $\langle m_1, m_2, \ldots, m_{n-1} \rangle$ to denote the time it took to reach the ith other node ($i = 1, 2, \ldots, n-1$). For the simulation of each of the message prioritization methods in a 10-node network, we summarize the message progression in a separate plot in the top half of Figure 3.3. Statistics shown in these plots include median, 25/75-quantile, 95% confidence, and outliers. In the bottom plot of the figure, we have the medians of the four methods together. Figures 3.4 and 3.5 present the same information for simulation in 20 and 30-node networks. We observe that the message progression rate of Oblivious is about an order of magnitude faster than the other methods, indicating that it is very effective directing messages.

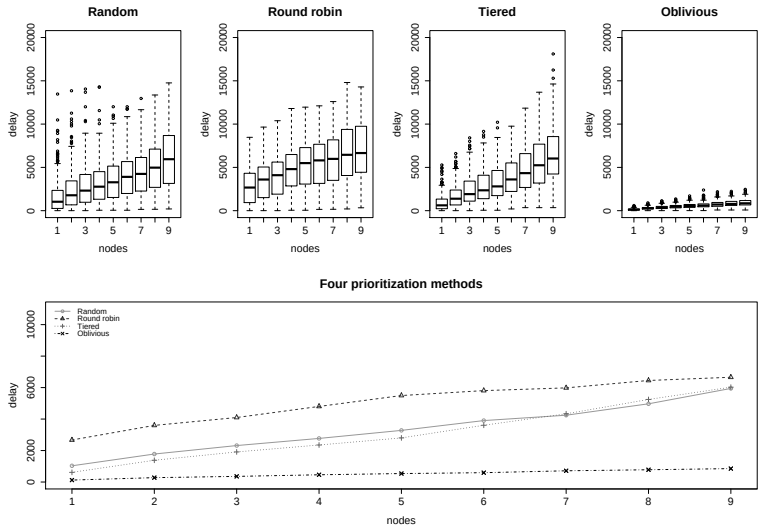

Figure 3.3: Message delivery progression in 10-node network

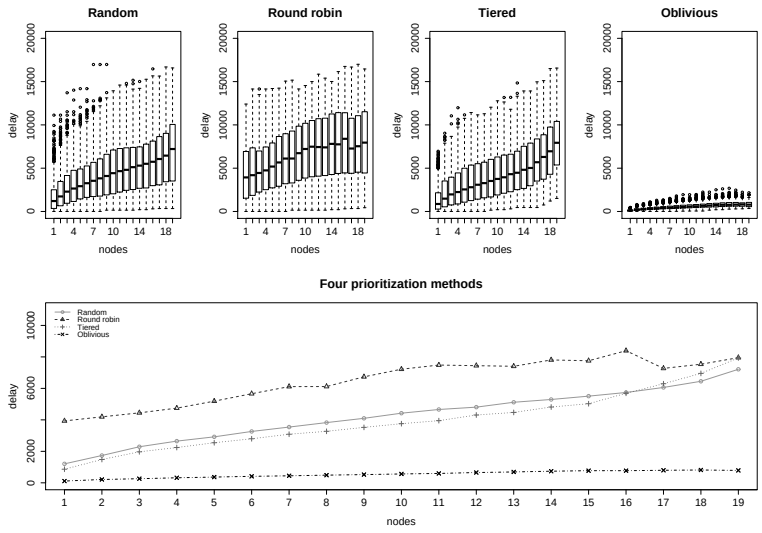

Figure 3.4: Message delivery progression in 20-node network

In this chapter, we reported computer simulation results on a variety of message dissemination methods for a comparative study of their relative performance. With emphasis on newly injected messages, the Oblivious method is able to effectively distribute message across the network quickly. Our next step is to port these methods to actual mobile devices and test them at the same, real venue in the next chapter. By comparing the results to those reported here, we will be able to fine-tune some parameters in the ONE, so that we can use the simulator to test larger networks with better confidence.

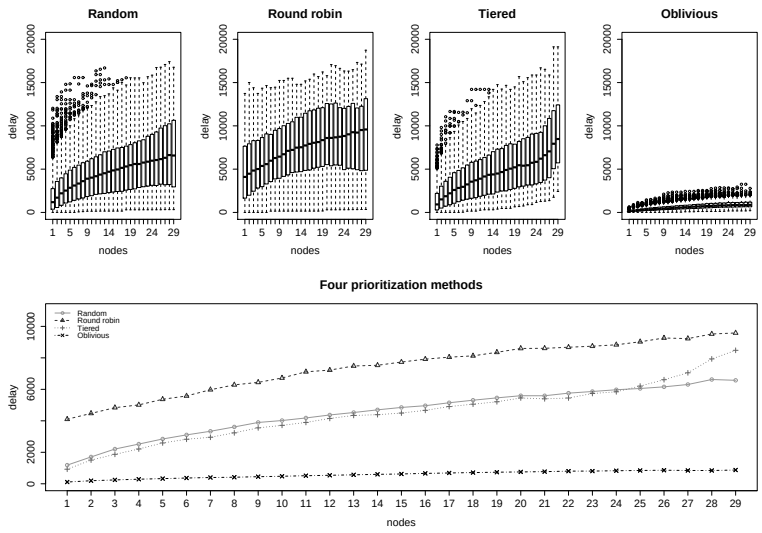

Figure 3.5: Message delivery progression in 30-node network

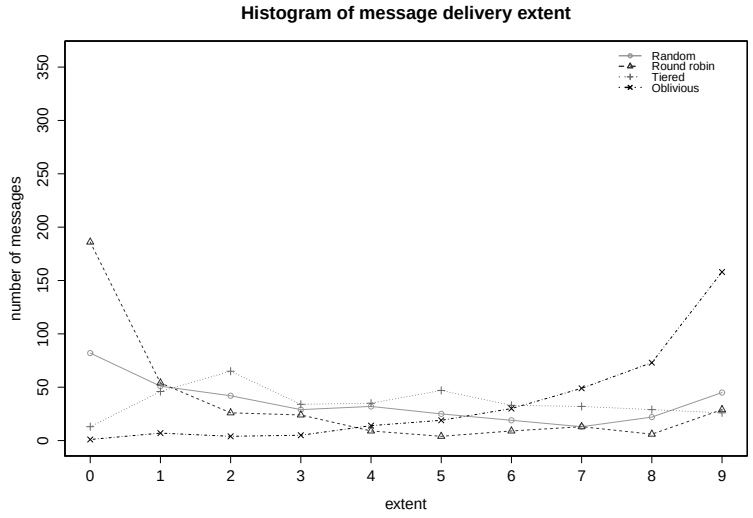

Figure 3.6: Message delivery extent in 10-node network

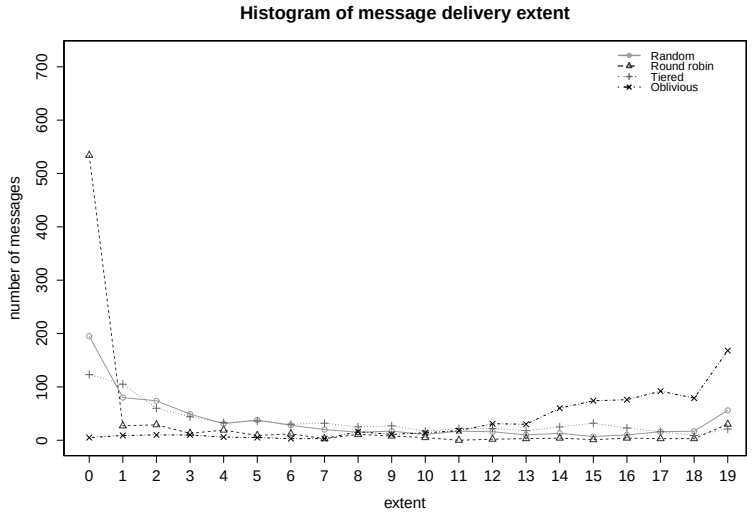

Figure 3.7: Message delivery extent in 20-node network

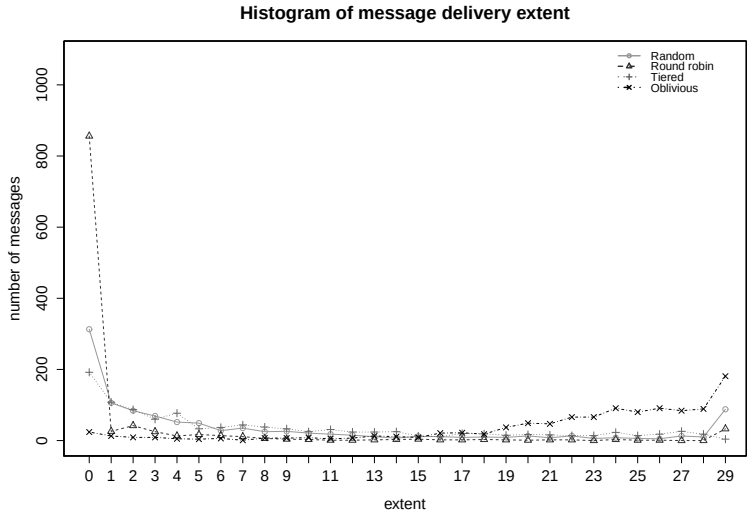

Figure 3.8: Message delivery extent in 30-node network

Chapter 4

Network Coding Using Real Devices

The concept of network coding was formulated in the seminal work of Ahlswede, Cai, Li, and Yeung [19] in 2000, and the past decade has seen a tremendous momentum in this area [17]. Its idea breaks the principal of traditional multi-hop networking, where intermediate nodes only forward packets but do not modify their contents, much like cars traveling on a highway. Since bits are not cars anyway, network coding allows intermediate nodes to combine packets from different input ports before forwarding them. When treating a packet as a sequence of symbols, even linear network coding defined over small Galois fields can introduce significant throughput gain. The readers are referred to an easy-to-read and yet informative primer by Fragouli, Le Boudec, and Widmer [7]. Other benefits of network coding include improved robust-

ness of network operations, higher energy efficiency in wireless radios, and better security against eavesdroppers. Network coding proves to be especially powerful and flexible, and can be exercised along with other revolutionary networking paradigms. For example, it was shown that opportunistic data forwarding in multi-hop wireless networks can further increase the capacity of these networks when intermediate nodes strategically combine overheard packets and forward them [22, 12]. As another example, the resilience to lost or delayed information brought about by network coding turns out particularly effective in DTNs, as evidenced by computer-simulated experiments in Widmer and Le Boudec [14].

In this chapter, we evaluate the how network coding stacks against conventional message passing in DTNs using both real iOS devices and in the ONE simulator in a university building. Our goal is to assess to what extent the ONE as one of the best and most widely used simulators for DTN research can mimic the real world. On one hand, we used real Apple iOS devices to measure how message propagate among roaming users over the built-in Bluetooth radios. On the other hand, we enhanced the ONE with a more realistic link layer by adding a few parameters. We are able to claim that the simulator work fairly closely to iOS devices with these parameters tuned properly. As part of a bigger research project, we may be confident that the simulator can work in place of real devices for efficient studies of larger-scale networks [28].

4.1 Implementation

In random network coding, when a set of packets are combined and sent by an intermediate node, both the combined message (i.e. the *information vector*) and how they are combined (i.e. the *encoding vector*) are to be included as being transmitted [7]. The dimensionality of the information vector is simply the number of symbols in the message content, say M, while the dimensionality of the encoding vector, denoted m, is the maximum number of original messages that can be combined. Apparently, when a packet is sent, $m + M$ symbols are transmitted. The greater m is, a higher percentage of communication capacity is consumed by such a coding overhead. When a packet is received by a node, it is inserted in its decoding matrix. Depending on whether the packet is innovative, the rank of the decoding matrix may or may not increase by 1. In any case, the rank of the matrix can be up to m, and is usually in that ballpark in a stable network. Because matrix operations on general-purpose processors can be expensive, a large value of m also implies a large computational overhead. Thus, for practical purposes, we can divide packets into non-overlapping *generations* and only allow packets of the same generations to be combined as in Chou, Wu, and Jain [18]. By tuning the size of a generation, we can control the communication and computation overhead. Widmer and Le Boudec [14] show that the generation size is a crucial parameter for the performance in their simulated studies of DTNs.

In this research, we set the generation size $G = 50$ globally in our tests. Provided

we have $n = 10$ devices, every device contributes 5 messages to each generation. Specifically at any given time, for device i ($i = 1, 2, \ldots, n$), its jth message ($j = 1, 2, \ldots, g$ for some latest generation g) belongs to generation $\lceil j/5 \rceil$ of the network. In addition, this message takes dimension $i \times (n-1) + (j \mod 5)$ in that generation. During the operation of the network, a node would have generated and received packets of various generations. We use P_k to denote the set of packets of generation k, where $k = 1, 2, \ldots, g$. Thus, collectively, we use $\mathcal{P} = \{P_1, P_2, P_3, \ldots, P_g\}$ to denote the generations of packets stored at said node. When a node is within range of any peer, it periodically (every $\tau = 15$ seconds in our experiments) generates a set of random combinations of the packets it has received so far and broadcasts them to its neighbors. These packets are generated as follows. For generation k ($k = 1, 2, 3, \ldots, g$), it creates $\max\left\{w \times \frac{R_{P_k}}{2^{g-k}}, 1\right\}$ random combinations of all packets in this generation, where w is a parameter to control the overall load on the radio, and R_{P_k} is the rank of the decoding matrix corresponding to the kth-generation packets P_k. That is, each generation contributes at least one random packet combination. In addition, when $w = 1$, the latest generation g contributes random combinations of at most its rank, the second latest generation $g - 1$ contributes up to half of its rank, generation $g - 2$ a quarter, and so on. Once created, these packets are broadcast in the neighborhood with the latest generation first and earliest generation last. Apparently, the greater w is, the more packets are broadcast periodically, the larger the communication overhead of the protocol is, and the higher the network

throughput may be. Essentially, the purpose of such a weight allocation among generations is to boost the late generations with sufficient initial presence in the network for them to propagate through.

4.2 Experiments

We conducted experiments both on a set of 10 Apple iOS devices and in the ONE simulator. These experiments were designed for the same scenario in part of the Engineering Building on campus of Memorial University of Newfoundland. The 10 mobile users follow some prescribed paths in the building in a 30-minute iteration. The users are divided into three groups of 3, 3, and 4 devices, respectively. Each group has a "base" in the building, as numbered in Figure 4.1. During the test, a user from a group leaves his/her base, walks along the path, for example as depicted in the figure, makes a stop at the other two bases for about a minute each, and returns to the base. Subsequently, the next user in the group would repeat the same routine. Users of different groups follow slightly different paths, especially in opposite directions in certain segments, so that they can meet users of other groups en route. These routines were intended to mimic both grouped and individual mobility patterns in an academic setting, and were used both in real-device and simulated tests.

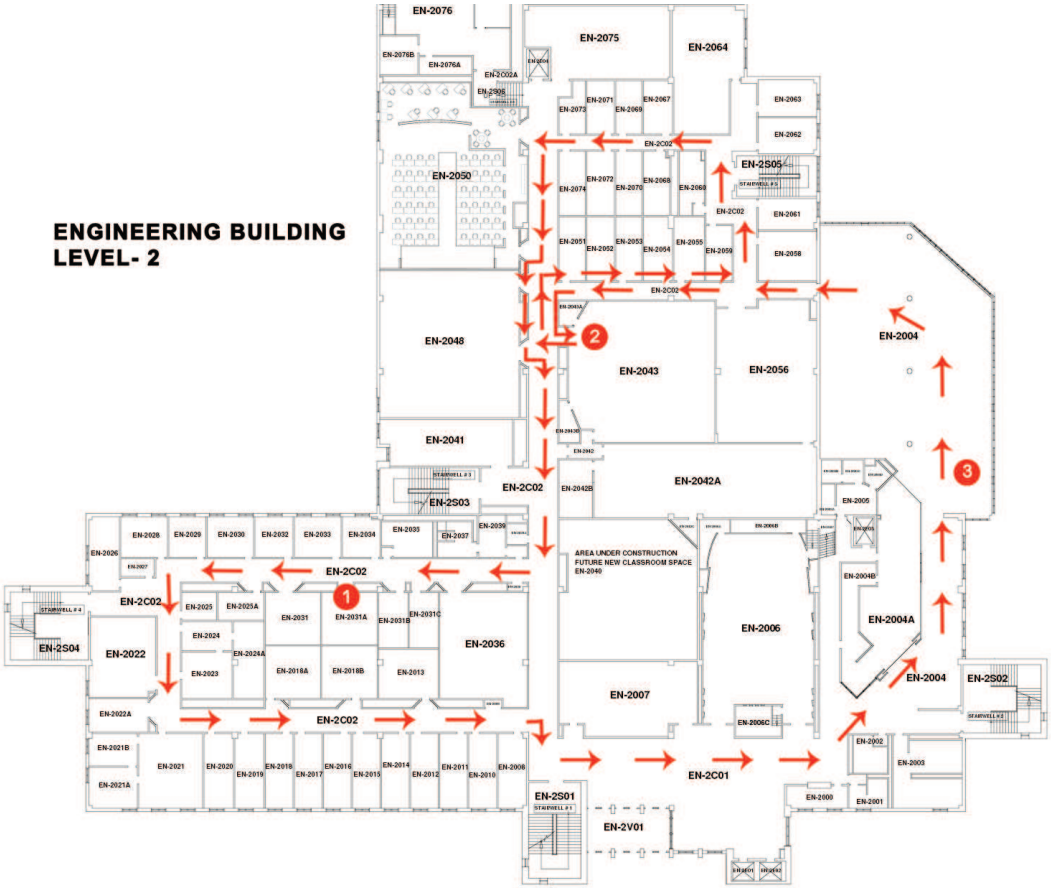

Figure 4.1: Path of a mobile user

4.2.1 Real-device

To have a heterogeneous network, we used a set of different iOS devices because there is no interoperability between iOS and Android OS over Bluetooth with GameKit. Their model, processor clock, Bluetooth version, and quantity are listed in Table 4.1.

We tested the network-coding-based broadcast against the other four forwarding-based approaches, each in a separate iteration. During an iteration, a devices gener-

Model	SoC and CPU cores	Bluetooth version	Quantities
iPod touch 4	A4, 1@0.8GHz	2.1	1
iPhone 4	A4, 1@1.0GHz	2.1	1
iPad 2	A5, 2@1.0GHz	2.1	2
iPod touch 5	A5, 2@1.0GHz	4.0	2
iPad mini	A5, 2@1.0GHz	4.0	3
iPad 4	A6X, 2@1.4GHz	4.0	1

Table 4.1: iOS devices used in experiments

ates a message every 90 seconds. The messages are randomly coded and sent every 15 seconds (Section 4.1) or selectively advertised as digests every 15 seconds (Section 3.3). For the case of network coding, we set the generation size to 50 messages, i.e. 5 messages per device per generation. To have about the same link layer data load across these five different methods, we are particularly interested in the generation allocation weight w to 0.5. When $w = 0.5$, we were able to keep the data sending rate at about 25kbps and receiving rate at about 50kbps across the board. (Note that we are using a broadcast service from the API, so there is no conservation of data flow.) Relevant parameters are summarized in Table 4.2. We recorded when messages were decoded (for network coding) and received (for non network coding) on each device. At the end of the test, these logs were uploaded to a server for centralized synthesis and post-processing.

Parameter	Value
number of nodes in network n	10
total simulation time T	1,800 seconds
node mobility model	walk along prescribed paths
message generate rate per device t	every 90 seconds
message length s	4,000 bytes
number of digests advertised l	10 messages
size of network coding generation G	50 messages
interval of digest advertisement τ	every 15 seconds
interval of coded packets broadcast τ	every 15 seconds
generation allocation weight w	0.5, 1, 2

Table 4.2: Parameters of device tests

We are interested in how widely messages are disseminated in the network, measured in *extent*, which is somewhat similar to the packet delivery ratio in unicast. After a message is generated, it is first stored at the originator, and as time goes on, it reaches more and more nodes. We observe how many other nodes a particular message has reached by the end of the 30-minute test. For a given message m, we denote the set of nodes in the network that m has reached other than the message originator itself by O_m. Thus, the extent of message m is defined as $|O_m|$, i.e. how many other devices the message has reached in the end. Among our 10 devices, 200

messages are injected in total, collectively denoted by M. As such, we plot a histogram of the extent over M for network coding with $w = 0.5$, 1, and 2, and for the non-coding approaches in Figure 4.2. Note that $w = 0.5$ is the case when network coding has comparable communication overhead as the non-coding approaches.

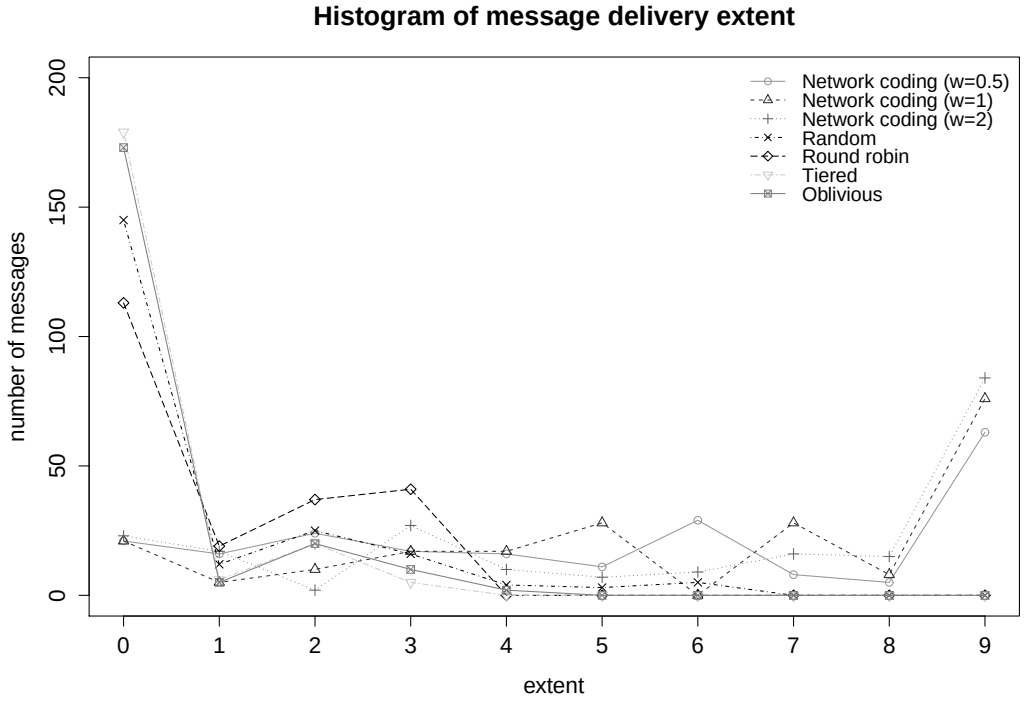

Figure 4.2: Broadcast extent for real devices

In the figure, we can see that the non-coding approaches all end up with many messages not being delivered to any other node, i.e. the case of extent 0, because of the very sporadic connections among devices. Among these methods, when there is more equal opportunities of messages being advertised, as with the cases of Random

and Round robin, the extent is slightly better. The other two approaches, Oblivious and Tiered, would favor newly injected messages but, unfortunately, they can be relentless moving on with new messages and permanently leave certain old messages behind if they miss the window. In stark contrast, the three network coding variants are able to send nearly half of the messages to all 9 other devices. Although for $w = 2$ the number of messages reaching all devices is slightly higher than when $w = 0.5$ or 1, they are fairly comparable, showing that $w = 0.5$ being very effective and efficient. Table 4.3 is a consolidation of the histogram into two cases, messages reaching only the minority in the network (0-4 other devices) and those reaching the majority (5-9 other devices). We can see that network coding was able to utilize the transient links very well while the non-coding forwarding could hardly make any progress. Note that for the messages generated near the end of the experiments, they barely had any time to propagate afar, and all these approaches would have better extent metrics if we gave them more time by allowing a "damping" period bat the and of the test.

4.2.2 Simulated

Although The ONE supports an arbitrary data rate at the Link Layer for nodes within a given range, our preliminary tests show that it could not closely simulate the iOS Bluetooth radios as is. Apparently, we needed the ability to customize other aspects of the link layer. Thus, we modified the simulator by adding two

Method	$1 \sim 4$	$5 \sim 9$
Network coding ($w = 2$)	45.0%	55.0%
Network coding ($w = 1$)	31.5%	68.5%
Network coding ($w = 0.5$)	34.5%	65.5%
Oblivious	96.0%	4.0%
Tiered	100.0%	0.0%
Random	100.0%	0.0%
Round robin	100.0%	0.0%

Table 4.3: Number of messages reaching 1-4 and 5-9 devices

parameters to control its behavior. First, we added a connection delay d for the time that it takes the GameKit to negotiate and connect two devices when they come in range. Second, we also gave a link a failure probability p to accommodate the fact that some transmission may fail when the radio is busy. (Note that these changes are by no means to catch how exactly the Bluetooth radios negotiate among each other, how they create and maintain synchronous or asynchronous links, or how they resolve collisions and provide reliability as it would in ns-2. They are simply to parameterize their synthesized effects on the higher layers.) We then simulated a 10-node network, where all nodes are identical hardware-wise and the data rate was 2Mbps for Bluetooth EDR. After testing various combinations of d and p and measuring the message deliver extent, we found that when we gave d a uniform

value between 0 and 10 in the experiment and set $p = 50\%$, the ONE has a very close behavior as using actual iOS devices. Due to limit of space, we only report results for these particular values in Figure 4.3. Furthermore, we consolidated the data the same way as with real devices and summarize it in Table 4.4. We can see that the simulator yields very similar relative performance between network-coding and non-coding based approaches in this case.

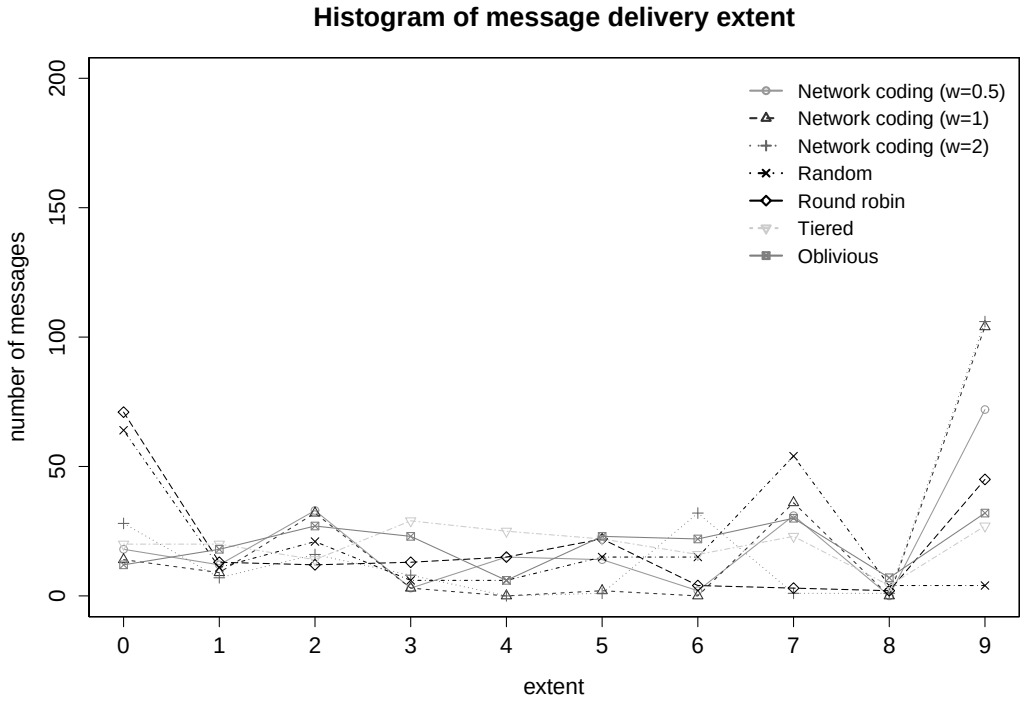

Figure 4.3: Broadcast extent in simulation

We started out with a goal of experimental studies of DTNs of medium-to-large sizes to assess their performance with and with out using network coding. Due

Method	$1 \sim 4$	$5 \sim 9$
Network coding ($w = 2$)	40.5%	59.5%
Network coding ($w = 1$)	29.0%	71.0%
Network coding ($w = 0.5$)	29.5%	70.5%
Oblivious	54.0%	46.0%
Tiered	62.0%	38.0%
Random	54.0%	46.0%
Round robin	43.0%	57.0%

Table 4.4: Number of messages reaching 1-4 and 5-9 nodes in simulation

to the prohibitive cost of using real devices, both in terms of monetary and time senses, the process of directly working with mobile devices can be very laborious and error-prone. We then decided to find out how a simulator widely-used in DTN research, the ONE, would mimic real devices by a side-by-side comparison between the exactly same, real and simulated, scenario of 10 devices. The performance gain of network coding over conventional data forwarding was evident through real-device experiments. More importantly, we found that, after enhancing the link layer with a few necessary parameters to simulate the iOS GameKit, tests using the ONE would produce very similar performance to using the Apple iOS devices. We now have more confidence that the ONE would yield more reliable results in larger networks after the enhancement.

Chapter 5

Enhancement with Handshake

In Chapter 4, we found that random network coding very powerful in message delivery in a DTN compared to the four traditional message prioritization approaches. On the other hand, we wonder if such a completely random method can be improved in anyway. In this chapter, we describe a three-way handshake mechanism that can increase the efficiency of network coding by giving nodes more situational awareness. We are able to show that such a design allows messages to propagate further with comparable or less overhead using the ONE.

5.1 Challenges

In the thee-way handshake operation, when two nodes come into range of each other, the first exchange their knowledge of coded messages, and then decide if and how

to transfer coded packets from one node to the other. In order to do this, we must address the following challenges.

1. As an extra control overhead, the handshake packets must be informative and compact. What information should be included in such control packets?

2. When multiple nodes are within range of each other, preceding handshakes should benefit all nodes in range so that they may not need to repeat some of the information already broadcast. Furthermore, any data packet containing coded messages can potentially eliminate the need of other nodes sending redundant information subsequently. As such, an orderly way for nodes to coordinate the handshaking process is crucial to the design.

3. After exchanging the control packets, if a node decides to go forth and broad cast a data packet, which messages of which generations should be included in order to yield the best performance?

In the next section, we attempt to describe our three way handshake to resolve these challenges.

5.2 Design

The design of the handshake is centered around a crucial data structure that each node maintains. Nodes exchange such a data structure and extract useful information

through the handshake. The handshake has three types of packets as in Chapter 3, `iHave`, `iRequest`, and `iSend`. Nodes periodically broadcast `iHave` packets, and any node within range would reply with an `iRequest`. The node that sent the `iHave` replies with an `iSend` packet, containing coded messages of various generations.

5.2.1 Knowledge base

Besides a decoding matrix for each generation 2.3, each node, v, maintains a knowledge base when hearing from other nodes. It is essentially a table keyed on the generation ID, denoted by $\mathbb{T}$. It has the following attributes for each row.

1. **Generation ID** — the unique identity for identifying a generation of messages in the network, denoted by g.

2. **Rank** — the rank of the decoding matrix of the corresponding generation, denoted by r_g.

3. **Full-rank nodes** — a list of nodes whose decoding matrix for generation g has full rank according to the latest knowledge of v including node v itself, denoted by V_g^F. The purpose of this field is to decide if a neighbor can benefit from a coded message any more.

4. **Reached nodes** — a list of nodes who have received any coded messages of this generation including node v itself, denoted by V_g^R. This field indicates the presence of a generation within the proximity of v.

5. *Penetration rate* — a fraction of how many nodes are full-rank for this generation to the best v's knowledge. It is defined as $\frac{|V_g^F|}{|V_g^R|}$.

5.2.2 Handshake procedure

Assume that node u's timer triggers at a specific point of time. It broadcasts an `iHave`, which is received by node v and other nodes.

1. The `iHave` carries the knowledge base of u, denoted $\mathbb{T}(u)$. We also use $|\mathbb{T}(u)|$ to denote the number of entries in the table, i.e. the number of generations messages that node u is aware of.

2. After receiving the `iHave`, node v first updates its knowledge base. Specifically, assume that for a given generation g, if the list of full-rank nodes from u is $V_g^F(u)$, node v updates its own with $V_g^F(u) \cup V_g^F(v)$. Similarly, v updates its list of reached nodes with $V_g^R(u) \cup V_g^R(v)$

3. Next, node v prepares its `iRequest` packet by extracting a subset of the rows in $\mathbb{T}(u)$. This subset has two parts. First, it has all generations contained in $\mathbb{T}(u)$ but not $\mathbb{T}(v)$. Second, for each of the remaining generation g in $\mathbb{T}(u)$, we calculate the difference $r_g(u) - r_g(v)$. (If node v is not aware of generation g, we let $r_g(v) = 0$.) This second part of the subset is sorted by the difference above, from high to low. The `iRequest` packet constructed as such signifies which generations from node u would benefit the growth of node v's decoding

matrix of the different generations more as a priority list. We denoted it by $\mathbb{T}'(v)$.

4. Because multiple nodes can be within range of node u, as with node v, and would send an `iRequest`, we want to give nodes that request more generations of messages a higher priority. This is achieved by enforcing a waiting time before the transmission of `iRequest`. Specifically, we make node v wait for $\beta \times \left(1 - \frac{|\mathbb{T}'(v)|}{|\mathbb{T}(u)|}\right)$ seconds, where β is a control parameter in the experiments. As a result, if node v requests everything, it waits for no time, while if it requests just a few generations, it would wait for nearly β seconds. Thus, we give nodes with "different levels of needs" different priorities. (Note that we are not dealing with possible collisions caused by the *Hidden Terminal Problem* [26] because the link layer may well have taken that into consideration.)

5. When any node w, which may or may not be node u, receives an `iRequest` $\mathbb{T}'(v)$ intended to u, it first updates its own knowledge base as in Step 2. Next, it responds according to the following cases.

   - If $w \neq u$ and does not have a pending `iRequest` packet for u, it exits the handshake.
   - If $w \neq u$ and has its own pending `iRequest` packet for u, it removes generations in $\mathbb{T}'(w)$ that are requested in $\mathbb{T}'(v)$, and continues to wait for the time to transmit its own `iRequest` as long as $\mathbb{T}'(w)$ is not empty.

(While it waits, it may receive additional `iRequest` packets.)

- if $w = u$, it goes to the next step to transmit an `iSend` packet.

6. For node u to prepare the `iSend` packet, it needs to determine 1) which generations of coded messages should be included, 2) in what order, and 3) how many messages in a generation. The first two questions can be answered by the composition of the `iRequest` just received. Assume that the request $\mathbb{T}'$ carries message generations $\{g_1, g_2, \ldots, g_{|\mathbb{T}'|}\}$. Because this is an ordered set to indicate the "importance" of generations being requested, the `iSend` will include these generations in the exact same order. For the third question, our ideas are to let generations with rapidly changing penetration rate $\frac{|V_g^F|}{|V_g^R|}$ include more coded packets. Specifically, for generation g_k ($k = 1, 2, 3, \ldots, |\mathbb{T}'|$), the number messages to include is

$$p_{g_k} = \begin{cases} r_{g_k} & \text{if change of } \frac{|V_{g_k}^F|}{|V_{g_k}^R|} \text{ is } 0 \\ r_{g_k} + \frac{G}{n} \times \left(|V_{g_k}^R| - |V_{g_k}^F|\right) & \text{otherwise,} \end{cases}$$

   where G is the number of messages in a generation and n is the number of nodes in the network as defined in Table 5.1.

   Node u then proceeds to transmit the `iSend` packet prepared as such.

7. Any node receiving the `iSend` packet should process it, extract the coded messages, and insert them into the decoding matrices of the corresponding generations to utilize the broadcasting nature of wireless channels as much as possible.

5.3 Experiments

To evaluate the performance improvement using handshake, we used the ONE to compare it with three variants of the network coding in the same map-based scenario as in Chapter 4). Moreover, in order to make the experiment results contrastive, we set the period of generating a message from 90 seconds in the previous experiments to 30 seconds in this experiment, so as to simulate a large number of messages in the network. Other parameters are detailed in Table 5.1.

Parameter	Value
number of nodes in network n	10
total simulation time T	1,800 seconds
node mobility model	walk along prescribed paths
message generate rate per device t	every 30 seconds
message length s	4,000 bytes
number of digests advertised l	10 messages
size of network coding generation G	50 messages
interval of digest advertisement τ	every 15 seconds
interval of coded packets broadcast τ_w	every 15 seconds
interval of `iHave` broadcast τ_{hs}	every 100 seconds
generation allocation weight w	0.5, 1, 2

Table 5.1: Parameters of simulation

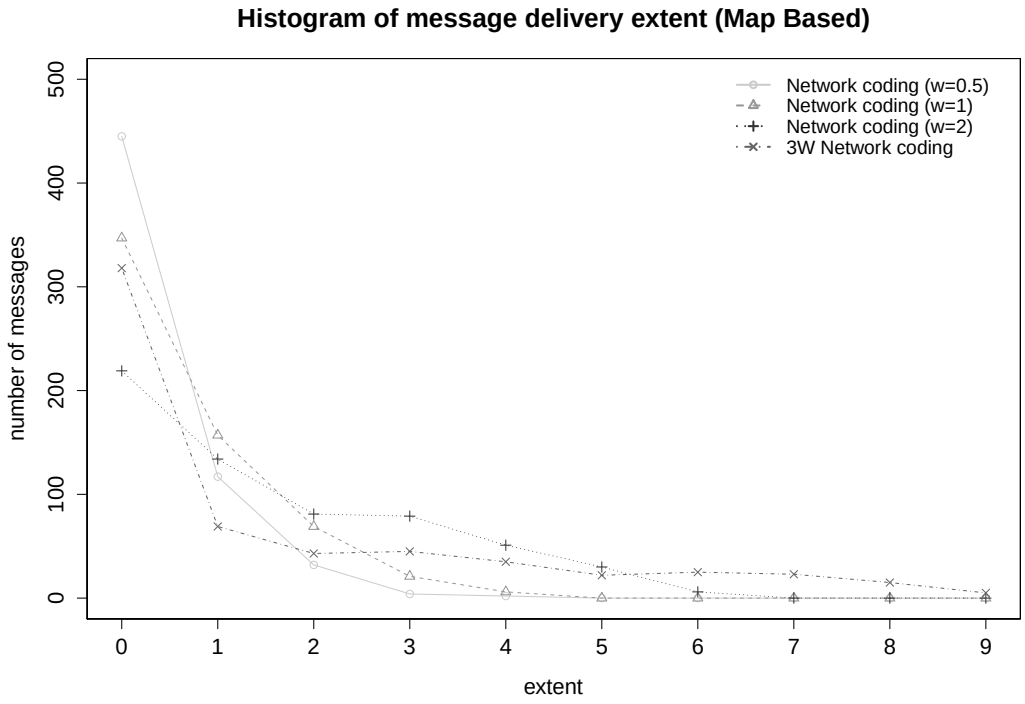

Figure 5.1: Message delivery extent

First, we plot the number of devices that messages reach at the end of the simulation in Figure 5.1. The same data are also summarized in Table 5.2. We observe that the handshake mechanism is able to boost message dissemination so that many more can reach at least 5 other nodes in the network compared to without using handshake.

To see how much communication overhead (i.e. control and data) these different methods incur, we summarize the number of bytes sent and received in Table 5.3. (Note that the network may send many more bytes than receiving because once an

Methods	$1 \sim 4$	$5 \sim 9$
Network coding ($w = 0.5$)	100%	0%
Network coding ($w = 1$)	100%	0%
Network coding ($w = 2$)	94%	6%
Network coding with handshake	85%	15%

Table 5.2: Number of messages reaching 1-4 and 5-9 devices

`iSend` is prepared, it will be transmitted regardless whether any other is in range or not. The simulator could be further optimized by aborting the handshake in this case but this is beyond the goal of this thesis.)

Methods	sent	received
Network coding ($w = 0.5$)	3,750,392	1,995,939
Network coding ($w = 1$)	5,448,862	2,594,400
Network coding ($w = 2$)	9,371,103	4,365,113
Network coding with handshake	8,994,722	5,737,374

Table 5.3: Overhead

In this chapter, we are able to show that, by introducing handshaking to guide network-coding based message passing, messages can be further propagated in the network at no extra communication overhead. We do not believe that the way we

implemented the handshaking is the only option, so we expect that the performance can be improved even more by fine-tuning it for the future.

Chapter 6

Conclusion and Future Research

6.1 Concluding remarks

This thesis was motivated by the need to communication in extreme networking conditions of DTN, where an end-to-end path connecting two nodes rarely exists in the network but nodes are highly mobile. We started with modifying the ONE simulator to truly support epidemic routing with a 3-way handshake. We implemented 4 variants of message prioritization to study how they affect the rate messages are disseminated in the network. The general observation was that favoring recently generated messages would boost their initial presence in the network for them to have equal opportunities to reach the destination as older messages.

We implemented network coding in the ONE simulator and on real iOS devices to study 1) how network coding improves the network performance of DTN, and 2)

in what conditions the simulator would produce matching results as real devices. We found that with proper modifications, the simulator can be a trustworthy tool to produce satisfactory simulation results, with and without network coding, in certain conditions.

We further revised epidemic routing using network coding by transmitting more targeted coded messages than being completely random. This is inspired by the 3-way handshake from epidemic routing. To do this, nodes convey summaries of their decoding matrices to their neighbors in order to compose packets of coded messages that can better serve the neighborhood. We were able to show with computer simulation that network coding can indeed be further enhanced with such a guidance.

6.2 Future research

The research reported in this thesis can be extended in a few very interesting ways. Some of these are related to the proposed methods while others are able the ONE simulator itself.

- In network coding with handshake (Chapter 5), an `iSend` packet usually contains a large number of rows. The current implementation in ONE is to use a queue to accommodate all such `iSend` packets as soon as they are prepared. Once in this queue, they will be sent by the antenna event if the requester has left the transmission range. It would be more efficient if a node can detect the departure or arrival of another with the help of the Link Layer so that

transmission of an `iSend` packet can be aborted. In a future extension, this can be achieved by maintaining the `iSend` packets in a queue that is managed by our user space code. With an intelligent dropping and reordering capability, such a queue management can potentially increase the network performance significantly.

- The generation size G is an effective parameter to control network coding. If the number of nodes in the network is known a priori, setting this parameter to a fixed value may serve this purpose. However, a more flexible and scalable approach would be to obtain a good value of it as the network operates and to allow it to adjust to the network conditions dynamically. This issue was studied in the simulated tests of Widmer and Le Boudec [14], and we intend to further investigate it using real devices. This would turn out to be much harder problem if the number of nodes in the network is unknown. A node must find a way to determine the index of each message of a generation using local information.

- Throughout the thesis, we assumed that each node has a message for all other nodes in the network. Therefore, we resorted to epidemic routing, with our without network coding and handshaking. It would interesting to test how to network coding may help if a message only needs to get to a single destination. How would the number of coded copies of a message affect the performance.

- One of the central issues of message passing in a DTN is the performance-resource trade-off. Whether it is 1-to-all or 1-to-1 communication, it may be helpful to incorporate a mechanism to stop propagating a certain message and clear it out of the network.

- When we work on 1-to-1 communication, a crucial issue is striking the balance between efficacy and cost by controlling the number of copies of the same message in the network.

- We used the Apple iOS mobiles for real-device tests. It would be interesting to see how much our observation from these devices can be consistent with other mobile platforms, such as the Android OS.

- The original ONE simulator has a simple Link Layer model of a fixed circular transmission range. In our experiments, we enhanced it by introducing a packet error rate for better mimicking the real world. It would be useful to include an 802.11 MAC model and possibly a PHY propagation model for even more realistic simulation.

Bibliography

[1] Kevin Fall, Stephen Farrell and Jörg Ott. Delay tolerant networking research group. http://www.dtnrg.org/wiki/Home.

[2] Kevin Fall, Stephen Farrell and Jörg Ott. DTN2 source code. `http://sourceforge.net/projects/dtn/files/`. Last accessed December 2nd, 2012.

[3] Amin Vahdat and David Becker. Epidemic routing for partially-connected ad hoc networks. Technical Report CS-200006, Duke University, 2000.

[4] Anders Lindgren, Avri Doria and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mobile Computing and Communications Review*, 7(3):19–20, July 2003.

[5] Ari Keränen, Jörg Ott and Teemu Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. New York, NY, USA, 2009. ICST.

[6] Aruna Balasubramanian, Brian Levine and Arun Venkataramani. DTN routing as a resource allocation problem. In *Proceedings of the 2007 Conference on*

Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), pages 373–384, New York, NY, USA, 8 2007. ACM.

[7] Christina Fragouli, Jean-Yves Le Boudec and Jorg Widmer. Network coding: an instant primer. *SIGCOMM Computer Communication Review*, 36(1):63–68, January 2006.

[8] Erik Nordström, Christian Rohner and Daniel Aldman. Haggle project. http://code.google.com/p/haggle/.

[9] Frédéric Guidec and Yves Mahéo. Document dissemination in mobile wireless ad hoc networks. `http://www-irisa.univ-ubs.fr/CASA/DoDWAN/index-en.html`, 2012. Last accessed December 2nd, 2012.

[10] Frédéric Guidec and Yves Mahéo. Opportunistic content-based dissemination in disconnected mobile ad hoc networks. *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM)*, pages 49–54, November 2007.

[11] Jet Propulsion Laboratory, California Institue of Technology. The interplanetary network (IPN). http://tmo.jpl.nasa.gov/.

[12] Jian Zhang, Yuanzhu Chen and Ivan Marsic. MAC-layer proactive mixing for network coding in multi-hop wireless networks. *Computer Networks*, 54(2):196–207, 2010.

[13] John Burgess, Brian Gallagher, David Jensen and Brian Neil Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[14] Jorg Widmer and Jean-Yves Le Boudec. Network coding for efficient communication in extreme networks. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking*, pages 284–291, 2005.

[15] Julien Haillot and Frédéric Guidec. A protocol for content-based communication in disconnected mobile ad hoc networks. *Journal of Mobile Information Systems*, 6(2):123–154, 2010.

[16] Maurice J. Khabbaz, Chadi M. Assi and Wissam F. Fawaz. Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges. *IEEE Communications Surveys & Tutorials*, 14(2):607–640, 2012.

[17] Muriel Medard and Alex Sprintson. *Network Coding: Fundamentals and Applications*. Academic Press, 1st edition, 2011.

[18] Philip A. Chou, Yunnan Wu and Kamal Jain. Practical network coding. In *Proceedings of Allerton Conference on Communication, Control, and Computing*, 2003.

[19] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li and Raymond W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[20] Sally Floyd, Craig Leres, Vern Paxson, Van Jacobson, Kevin Fall and Steven McCanne . ns2 (network simulator 2). http://www-nrg.ee.lbl.gov/ns/.

[21] Sushant Jain, Kevin Fall and Rabin Patra. Routing in a delay tolerant network. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 145–158. ACM, 2004.

[22] Szymon Chachulski, Michael Jennings, Sachin Katti and Dina Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 169–180, 2007.

[23] Thrasyvoulos Spyropoulos, Konstantinos Psounis and Cauligi S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking*, pages 252–259. ACM, 2005.

[24] Vincent Lenders, Franck Legendre, Martin May, Gunnar Karlsson, Bernhard Distl, Dominik Schatzmann, Theus Hossmann and Bernhard Plattner. PodNet

Official Website. http://podnet.ee.ethz.ch/. Last accessed December 2nd, 2012.

[25] Vincent Lenders, Franck Legendre, Martin May, Gunnar Karlsson, Bernhard Distl, Dominik Schatzmann, Theus Hossmann and Bernhard Plattner. Podnet - mobile distribution of user-generated content. *The International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2008.

[26] Wikipedia. Hidden node problem. http://en.wikipedia.org/wiki/Hidden_node_problem. Last accessed March 29th, 2014.

[27] Xu Liu, Yuanzhu Chen, Cheng Li, Walter Taylor and Jason H. Moore. Message prioritization of epidemic forwarding in delay-tolerant networks. International Conference on Computing, Networking and Communications (ICNC), Honolulu, Hawaii, USA, February 2014.

[28] Yuanzhu Chen, Xu Liu, Walter Taylor and Jason H. Moore. Delay-tolerant networks with network coding: How well can we simulate real devices? To appear in IEEE International Conference on Communications (ICC), Sydney, Australia, June 2014.

[29] Zhensheng Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. *IEEE Communications Surveys & Tutorials*, 8(1):24–37, 2006.

[30] Zhensheng Zhang and Qian Zhang. Delay/disruption tolerant mobile ad hoc networks: latest developments. *Wireless Communications and Mobile Computing*, 7(10):1219–1232, December 2007.