Computer Networks 83 (2015) 349–362

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Delay-tolerant networks and network coding: Comparative studies on simulated and real-device experiments

Yuanzhu Chen [a], Xu Liu [a], Jiafen Liu [b,*], Walter Taylor [c], Jason H. Moore [d]

[a] Wireless Networking and Mobile Computing Laboratory, Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, Canada
[b] School of Economic Information Engineering, Southwestern University of Finance and Economics, Chengdu, Sichuan, China
[c] Computational Genetics Laboratory, Giesel School of Medicine, Dartmouth College, Hanover, NH, USA
[d] Institute for Biomedical Informatics, Perelman School of Medicine, University of Pennsylvania, Philadelphia, PA 19104-6021, USA

ARTICLE INFO

Article history:
Received 27 December 2014
Accepted 6 April 2015
Available online 29 April 2015

Keywords:
Delay-tolerant network
Network coding
Mobile devices

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 breakthroughs in mobile and wireless communications in the past decade or so. As reported in this article, we are interested in how network coding interacts with such a challenged networking paradigm as DTN from an experimental perspective. 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.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Data communication networks connect computing devices with wired or wireless links to exchange information. For such networks to scale as the number of devices in it increases, we allow messages to traverse multiple communication links from its source node to the destination. Such a "store and forward" technique is the central idea of how the Internet can support numerous computers. This significantly extends the scope of communication networks spatially. Recently, research on Delay-Tolerant Networking (DTN) [1,15,17,30] has been focusing on how to extend communication networks temporally. As mobile devices and networking technologies become more powerful and efficient, such mobile devices can be used to "store, carry, and forward" data when users roam around. That is, even without cellular or Wi-Fi infrastructure and only relying on short-range radios, e.g. Bluetooth and ZigBee, a number of sparsely deployed mobile devices can be used to transfer data automatically especially when the data are not meant to be time-sensitive. The DTN technology can be useful in many scenarios, such as mobile sensor networks, disaster recovery, and social networking.

The concept of network coding was formulated in the seminal work by Ahlswede et al. [5] in 2000, and the past decade has seen tremendous growth in this area [23]. Its idea breaks away from the principal of traditional multi-hop networking, where intermediate nodes only forward packets but cannot modify their contents, much like cars traveling on a highway. Since bits are not cars anyway, network coding allows intermediate nodes to combine packets mathematically from different input ports just before forwarding them. When treating a packet as a sequence of symbols, even linear network coding defined

* Corresponding author.
E-mail address: ljfwhy@gmail.com (J. Liu).

http://dx.doi.org/10.1016/j.comnet.2015.04.002
1389-1286/© 2015 Elsevier B.V. All rights reserved.

over small Galois fields can introduce a fairly significant throughput gain. The readers are referred to an easy-to-read and yet informative primer by Fragouli et al. [12]. 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 judiciously combine overheard packets and forward them [7,28]. 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 [27] and Lin et al. [19].

In this research, we evaluate how network coding stacks against various conventional message passing techniques in DTNs using both real Apple 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 mobile 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 can behave fairly closely to iOS devices with these parameters tuned properly. As part of a bigger research project, we can be confident that the simulator can work in place of real devices for efficient studies of larger-scale networks.

The rest of this article is organized as follows. Next section, we review relevant experimental research on DTN and network coding in this context, and provide background information about the device API and simulation software used in this research. In Section 3, we describe an array of techniques for message passing without network coding. Next in Section 4, move to detail a generation-based implementation of network coding in DTN. We conducted experiments using both real devices and computer simulation. The experimental settings and results are reported in Section 5. Section 6 concludes this article with discussion and future research issues.

2. Background

Here, we review the most relevant research in DTN and network coding in DTN. We also provide a brief description to the tools used in the experiments, a Bluetooth API to the Apple iOS and a DTN simulation software suite in Java.

2.1. Related research

Research on DTN started from the Interplanetary Networking project at JPL [3]. 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, because the mobility of communication devices can be unpredictable, 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 [15,17,29,30].

The two most important, and yet distinct operations at the Network Layer (Layer III) are data forwarding and routing [18]. Forwarding regulates how packets are taken from one link and put on another. Routing determines which path a data packet should follow from the source node to the destination. The latter essentially feeds control input to the former. Here, we stick to the term of data forwarding although it is also sometimes referred to as routing in literature.

Data forwarding in unpredictable DTN is more or less inspired by Epidemic Routing [26]. There, the authors are interested in transferring messages to their destinations as quickly as possible at the cost of using a considerable amount of network resources consumed by making many copies of the same message. Subsequent work on unicast data, where a message has a sole destination, make more careful tradeoffs between the data transfer performance, in terms of latency and delivery ratio, and resource consumption. For example, Spray and Wait [25] regulates the number of copies a message using a single control parameter.

When assuming that historical contact information would suggest a similar pattern in future, nodes can utilize such observation to construct some sort of utility function to gauge which node in its proximity might help forwarding its messages more effectively. This approach was initially explored in PROPHET [20] and MaxProp [6] most noticeably. When historical contact records are further distilled with social network analysis methods, nodes can make more sophisticated forwarding decisions taking more factors into consideration. This approach is exemplified in BUBBLE-Rap [14], Delegation Fowarding [11], SimBet [10], and CAR [24].

Data forwarding techniques aside, researchers in data communications have also been on a quest for the killer applications of this groundbreaking technology for years [21]. Although such a quest is far from satisfactory, there have been a number of interesting applications in infrastructureless computer networking, such as IPN [3], Haggle [2], ZebraNet [16] and iSNAC [8], to name a few. Among these, iSNAC is a mobile social networking iPad application that focuses on broadcasting messages to help conference attendees to share information effectively.

Subsequent to the seminal work of Ahlswede et al. [5], another important discovery of network coding is its randomized application. As we know, the linear independency of the coefficients used to generate a set of coded packets is a determinant for the receiver to successfully decode for the native packets. This is especially crucial in wireless and mobile networks, where coded packets are subject to

erasure errors. Ho et al. [13] discuss the feasibility of adopting random coefficients in encoding, and prove that as long as the size of the finite field is not trivially small, the set of randomly picked coefficients will be linearly independent with a high probability. This significantly relieves the network nodes from allocating coefficients and only need to concentrate their power on encoding and transmitting the packets.

Network coding is also shown in Widmer and Le Boudec [27] and Lin et al. [19] to be very effective in battling the intermitted connectivity in DTNs while assuming neither do nodes have information about future encounters nor are they able to derive this information from past history. They use a custom computer simulator to compare the packet delivery ratio and delay between non-network-coding DTN forwarding and using random network coding. Their simulation indicates that network coding outperforms DTN forwarding for various network density levels and mobility patterns even when the latter is allowed to use very intelligent beaconing for nodes to be selective in message advertising. In addition to experimental results, the latter [19] also provides some nice performance bounds analytically.

2.2. Multipeer connectivity API

In our real-device experiments, we used the Bluetooth radios on Apple iOS devices to form a DTN. Apple provides the Bluetooth communication capabilities through a set of wrapping classes in the Multipeer Connectivity Framework (formally GameKit). Although this framework was originally designed for infrastructureless peer-to-peer gaming or media sharing, it is essentially an API (Application Program Interface) for devices to send generic data among themselves over the Bluetooth radios on board. The API provides both reliable and unreliable link layer data services between one-to-one and one-to-many devices. In our experiments, we picked the unreliable, one-to-many variant. Each invocation of data transmission is allowed to send up to 90 kilobytes in the payload. If the upper layer modules need to send more data, they must first be segmented into smaller fragments. Since the API was not known to handle rapid connections and disconnections well, there can be a delay in one device detecting another device coming into range, and links may break occasionally when they are busy. Nevertheless, these characteristics of the Multipeer Connectivity Framework provides us a perfect platform to study the effect of network coding in such an extreme and yet practically common scenario.

2.3. The ONE simulator

The ONE (Opportunistic Network Environment) [4] is a discrete-event computer simulator for DTN research. It was written in Java and made open-source for the research community. The ONE implements a fair number of well known message passing methods in DTN, such as Epidemic Routing [26], Spray and Wait [25], and PROPHET [20]. It has a large number of hooks to customize behaviors of these protocols and for us to add network coding to it. The simulator supports mobility in a map-based structure in addition to conventional free-space 2D mobility. This allows us to specify easily how users move around in an experiment area. The particular version of the simulator used in this project is 1.4.1. The ONE uses a simple link layer model with a binary circular transmission coverage area. Given a determined transmission radius, two nodes have reliable communication if they are within range; otherwise, they cannot hear from each other. Apparently, such an idealized link layer model would be unrealistic. In the second set of experiments (Section 5.2), we modified the link layer by adding delays in connection establishment and probabilistic link breaks with parameters set similarly to what we saw in the Multipeer Connectivity Framework so that the ONE could faithfully simulate iOS devices. Below are three of the most important components of the simulator that are often focal points for customization.

- *Movement Management Component*: This component is used for the simulator to move nodes in a given 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 a ONE configuration file, users can choose to 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 Fig. 9.
- *Event Management Component*: In order to handle a series of events in the simulation, ONE offers an Event Management Component. This component processes an event by fetching it from an event queue. We can schedule events, such as message generating, network topology snapshotting, network messages statistics, nodes insertion, 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 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.

3. Message prioritization without network coding

In this section, we are interested in the problem of disseminating messages from a particular source to all other nodes in the network without using network coding. We focus on a DTN, where a set of sparsely deployed nodes roam about without assuming any predictability. Each node periodically injects a message into the network intended to all other nodes. The strategies that a node employs depend on whether network coding is used. Without network coding, two nodes transfer messages via a handshake protocol. With network coding, a node simply randomly mixes the packets it has received so far and send these random combinations to an encountered peer.

When not using network coding, we take a similar approach to Epidemic Routing [26] in that, when two nodes come into transmission range of each other, one node can transfer a number of messages to the other via a 3-way handshake sequence. However, the difference here is that we must pick thoughtfully which messages to include in a short advertisement packet such that the system has a high overall throughput. Specifically, when node A discovers node B in its transmission range, it sends a *summary vector* SV_A. In the original Epidemic Routing, SV_A contains the unique IDs of all the message that A stores in its knowledge base. In our solution, it needs to be a small subset of these messages because there can be simply too many of them after the network has been up running for some time. After receiving SV_A, node B replies with $SV_A - M_B$, where M_B is the set of all messages stored at node B. As such, node B essentially tells A which messages from A would potentially enrich B's knowledge of messages. Next, node A retrieves messages in $SV_A - M_B$ from its storage and sends them to B in a burst to complete the handshake. If the two nodes are still within range τ seconds after the handshake, they will start another round of handshake to transfer more messages. Generally, every node would do the same to its neighbors periodically.

Given that nodes typically store more messages than what can fit in a single handshake packet, the strategy taken as of which messages should be in the advertisement and in what order affects the network performance significantly. We call such a strategy *message prioritization*, and some preliminary research on this scheme is reported in our previous article [22]. To reiterate, 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 ($l = 10$ in our experiments) created out of some of its stored messages.

(1) Round robin — Node A maintains a FIFO queue of the messages that it has received and generated so far, i.e. based on 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 encountered nodes. Note that as the network continues to operate, 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, in a decreasing order of significance. The fewer times it has been forwarded till reaching this node A, the later it was created by its origin, and the shorter its payload is, it is ranked higher in the storage queue. These ranked messages are split into three segments of roughly the same 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. Thus, the system helps newly injected message to spread in the network more quickly.

(3) Oblivious – Node A maintains a FIFO queue of all messages according to 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 network.

In addition to the three above, we also implemented a *random* prioritization method as a comparison baseline, where node A would randomly pick l messages and advertise their digests. All four methods are essentially different ways to allocate opportunities to messages to be advertised in the network.

4. With network coding

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 [12]. 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 neighborhood 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 practicality purposes, we divide packets into non-overlapping *generations* and only allow packets of the same generations to be combined as done in Chou et al. [9]. By tuning the size of a generation, we can

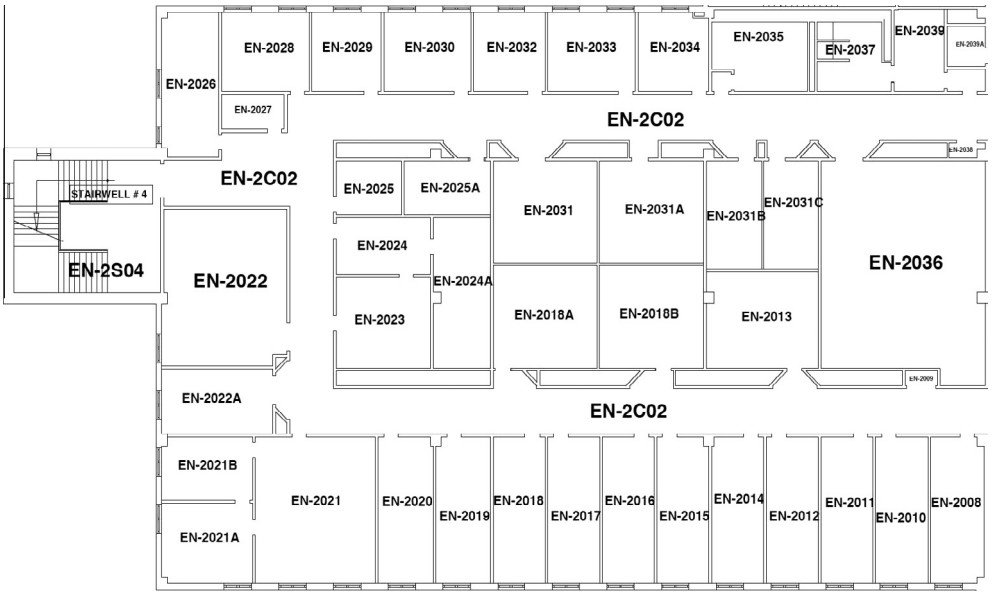


Fig. 1. Engineering building.

Table 1
Simulation parameters.

Parameter	Value
number of nodes in network n	10, 20 or 30
total simulation time T	20,000 s
node movement velocity v	0.5–1.5 m/s
message generate rate per device t	Every 400 s
message length s	2000–5000 bytes
number of digests in advertisement packet l	10 messages
interval of digest advertisement τ	Every 150 s
transmission range r	10 m
maximum packet length S	90 kB
delivery deadline d	3000 s

control the communication and computation overhead. Widmer and Le Boudec [27] show that the generation size is a crucial parameter for the performance in their simulated studies of DTNs.

Here, 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 $i \times (n-1) + (j \bmod 5)$ 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,

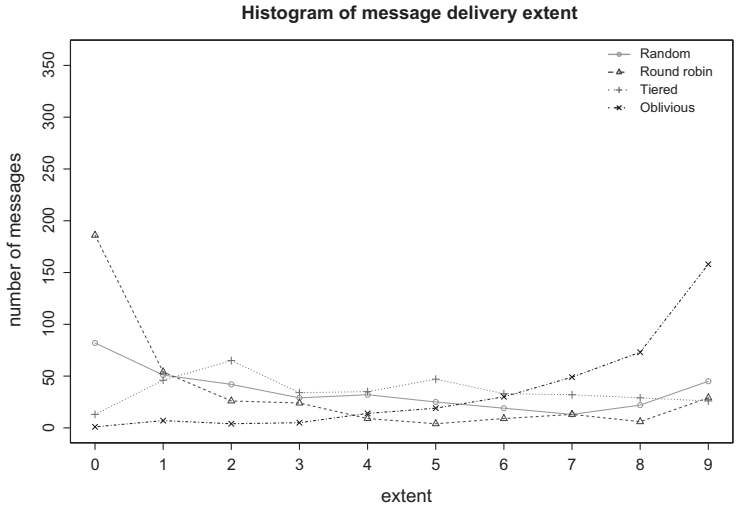


Fig. 2. Message delivery extent in 10-node network.

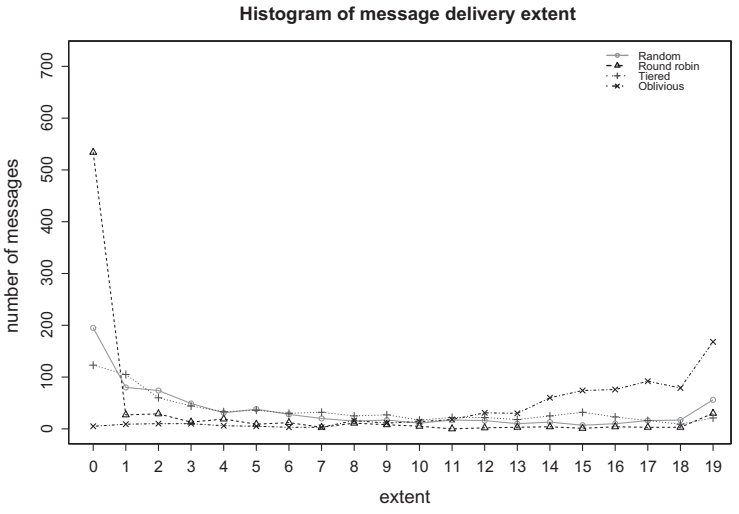


Fig. 3. Message delivery extent in 20-node network.

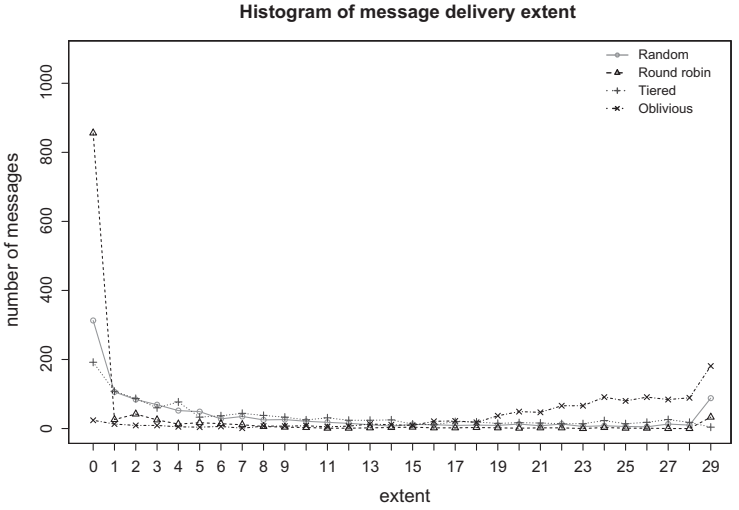


Fig. 4. Message delivery extent in 30-node network.

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 τ seconds) generates a set of random combinations of the packets it has received so far and broadcasts them to its neighbors. (In our real-device experiments, we set $\tau = 15$ to be able to accumulate sufficient traffic in a relatively short experiment time.) 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 so forth. 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 spread.

5. Experiments

We conducted two sets of experiments to study the interplay between DTN and network coding. The first set was done using the vanilla ONE simulator. The second set was a comparison between real devices and ONE modified to mimic real devices.

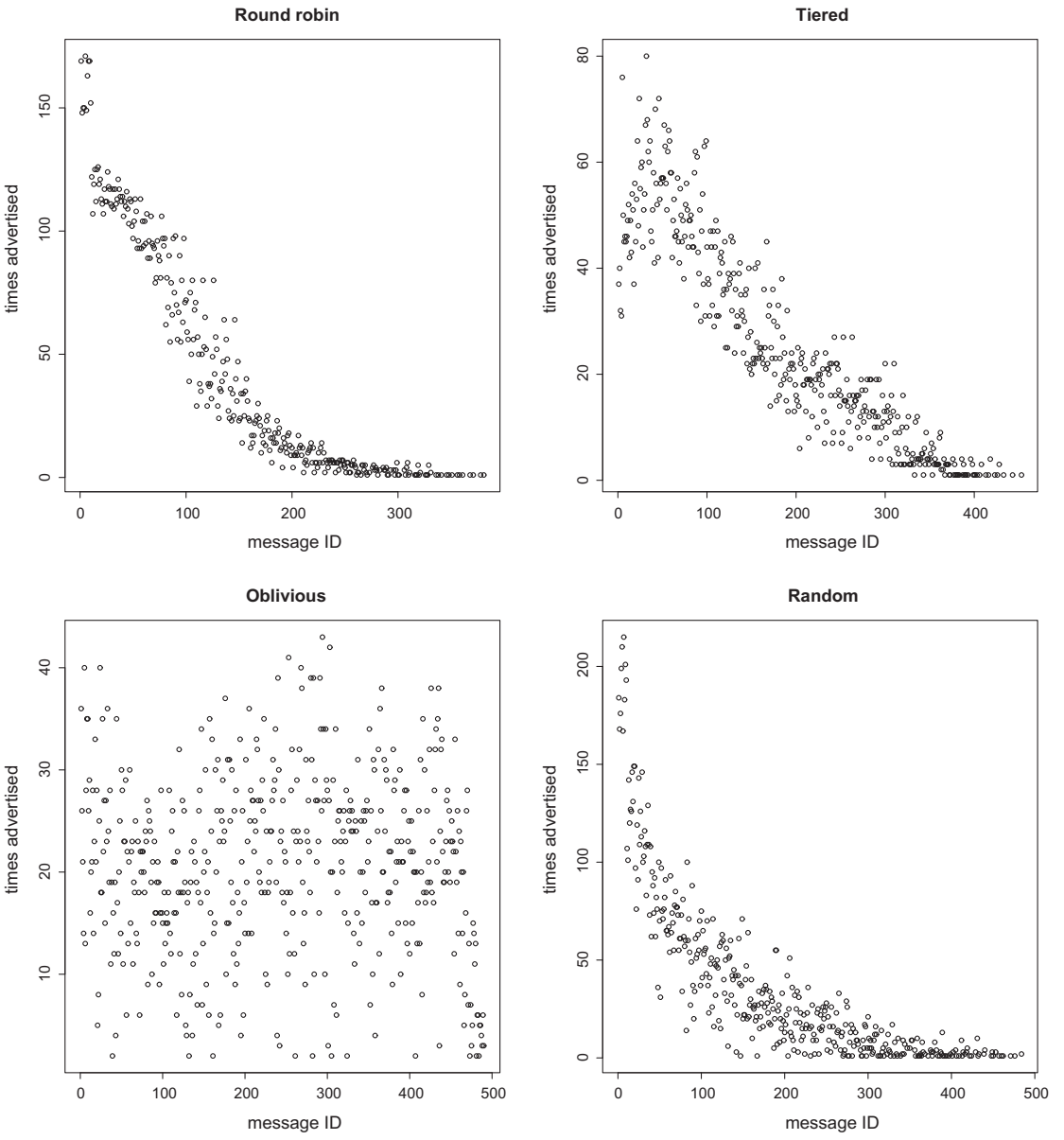


Fig. 5. Number of times a message is advertised in 10-node network.

5.1. Simulated experiments on message prioritization

First, we used ONE [4] to evaluate how different message prioritization methods affect the performance of the system. Here, the simulator was used out-of-the-box without any modification. 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.

We used the map mobility management of the ONE simulator, where a mapped structure of the simulation area is used to specify how nodes can move. During the simulation, a node can decide on a destination position, such as an intersection 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 m in simulation), they discover each other and start to exchange 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 (Fig. 1).

We assumed using the Bluetooth 4.0 radios on the iOS devices to be comparable to real devices later on. As such, the maximum size of a single packet in the handshake is limited to 90 kB. Around every 400 s, a device injects a message of size uniformly distributed in $[2000, 5000]$ bytes at random. Parameter settings in this part of the experiments are summarized in Table 1.

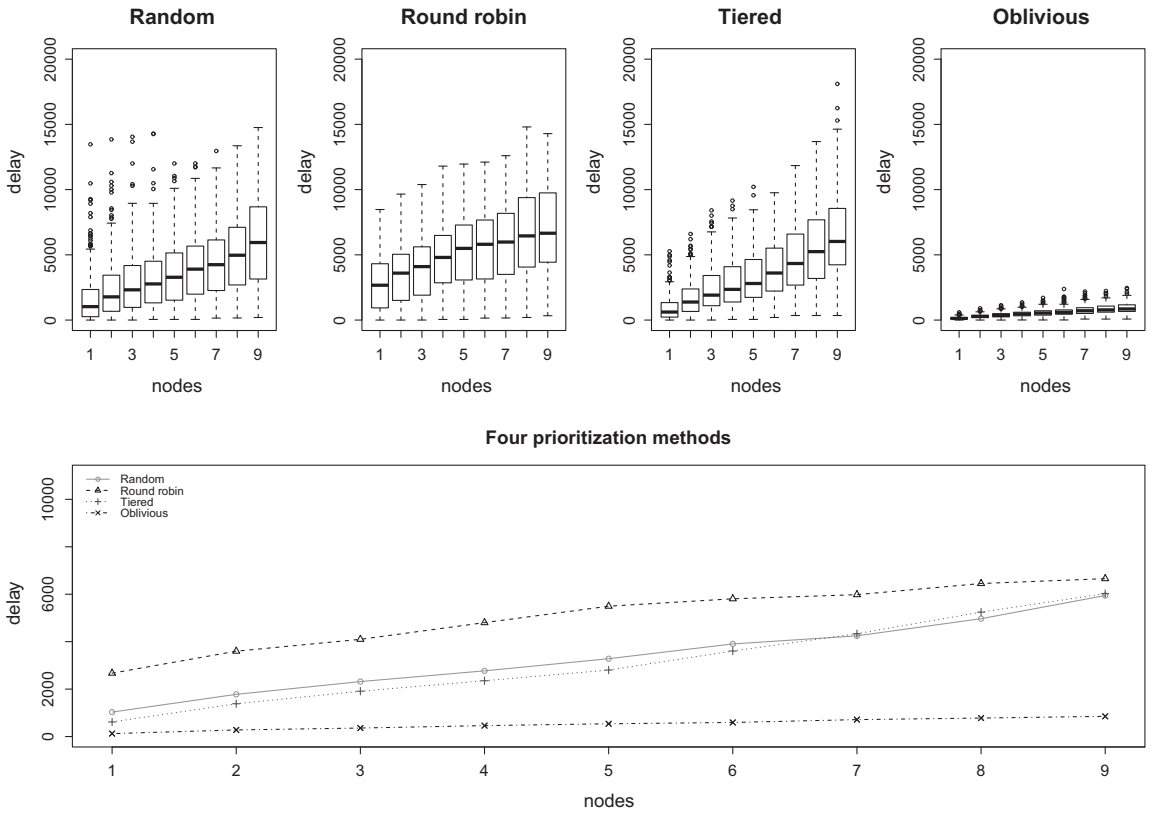


Fig. 6. Message delivery progression in 10-node network.

We are interested in how widely and quickly messages are propagated 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, the *extent* of message m is defined as $|O_{m,d}|$, i.e. how many other nodes the message has reached till the deadline. We consider the messages injected during the first 14,400 s of the entire 20,000 s of simulation so that all messages would have sufficient time to be disseminated. For a network of 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 Figs. 2–4. 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 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 Fig. 5, where we plot the number of times that a message is placed in an advertisement packet in the simulation of 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.

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 Fig. 6. 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. Figs. 7 and 8 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.

5.2. Comparative studies on network coding

We also conducted experiments both on a set of 10 Apple iOS devices and then in the ONE simulator. These experiments were designed for the same scenario in part

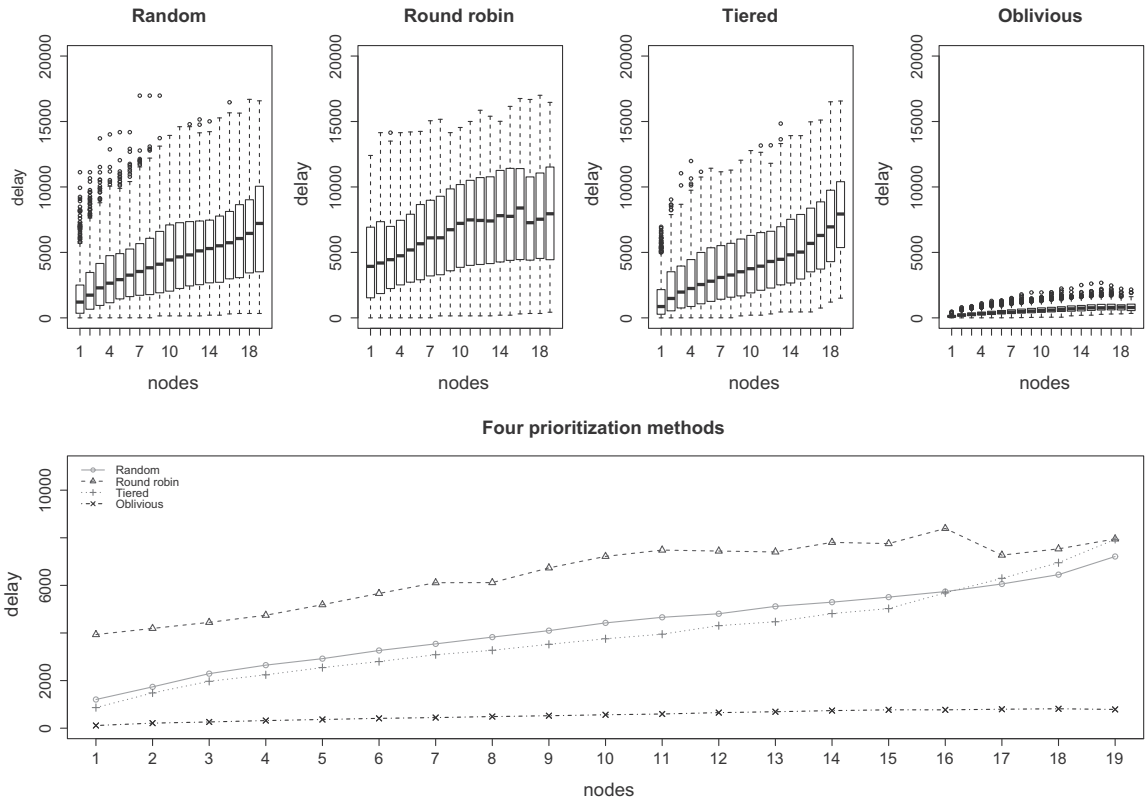


Fig. 7. Message delivery progression in 20-node network.

of the Engineering Building on the university campus. We chose a greater scope of the building than in the previous set of tests for the roaming devices to see more interesting contacting scenarios. Specifically, the 10 mobile users follow some prescribed paths in the building in a 30-min 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 Fig. 9. 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. The simulator was programmed to have the same mobile groups and mobility patterns.

5.2.1. 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 Multipeer Connectivity. Their model, processor clock, Bluetooth version, and quantity are listed in Table 2.

We tested the network-coding-based broadcast against the other four forwarding-based approaches, each in a separate iteration. During an iteration, a devices generates a message every 90 s. The messages are randomly coded and sent every 15 s (Section 4) or selectively advertised as digests every 15 s (Section 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 approximately the same link layer data load across these five different methods, we are particularly interested in setting the generation allocation weight w to 0.5. When $w = 0.5$, we were able to keep the data sending rate at about 25 kbps and receiving rate at about 50 kbps 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 3. 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.

We are interested in how widely messages are disseminated in the network measured in extent as defined previously (Section 5.1), which is somewhat similar to the packet delivery ratio in unicast. Among our 10 devices, 200 messages are injected in total, collectively denoted by $\mathcal{M}$. As such, we plot a histogram of the extent over $\mathcal{M}$ for network coding with $w = 0.5$, 1, and 2, and for the

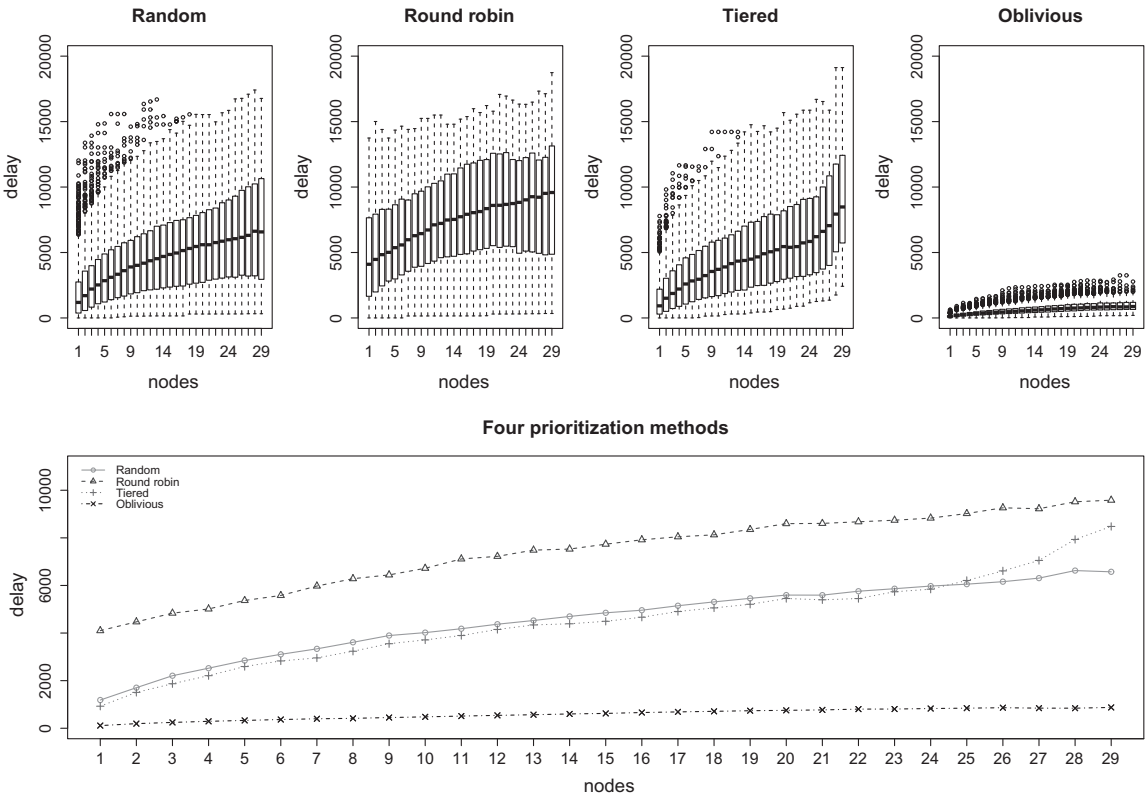


Fig. 8. Message delivery progression in 30-node network.

non-coding approaches in Fig. 10. Note that $w = 0.5$ is the case when network coding has a comparable communication overhead as the non-coding approaches.

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. This would be fine in the computer simulation using the original ONE simulator (Section 5.1) because links are prefect when two nodes are near each other. However, this turned out problematic for real-devices experiments, where links can break or their establishments can be delayed. If either happens, an old message may miss its opportunity of transmission once and for all. 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 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 chance to propagate very far, and all these approaches would have better extent metrics if we gave them more time by allowing a "damping" period at the and of the test.

5.2.2. 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 parameters to control its behavior. First, we added a connection delay d for the time that it takes the Multipeer Connectivity Framework 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 between 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 at

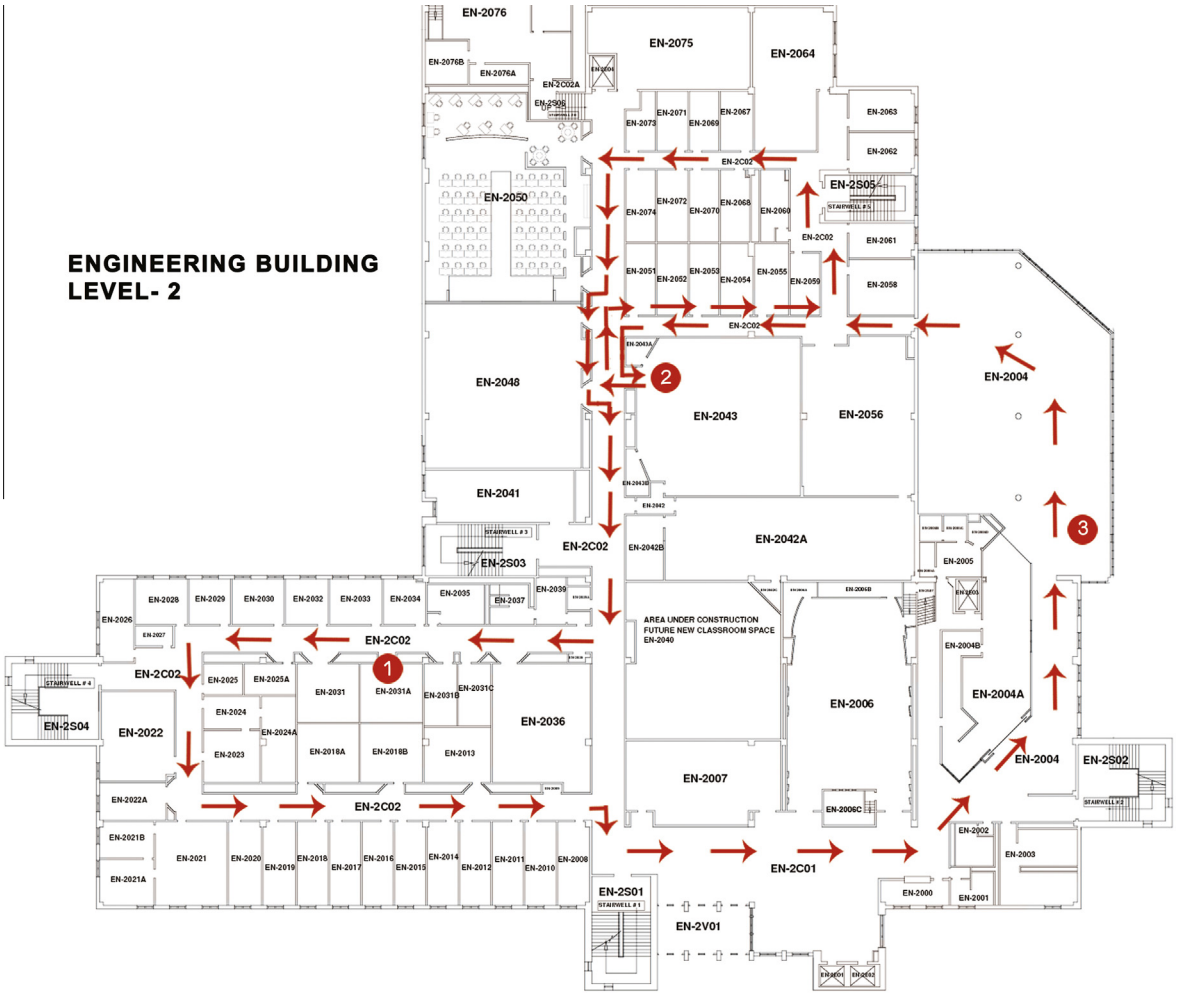


Fig. 9. Path of a mobile user.

Table 2
iOS devices used in experiments.

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

Table 3
Parameters of device tests.

Parameter	Value
Number of nodes in network n	10
Total simulation time T	1800 s
Node mobility model	Walk along prescribed paths
Message generate rate per device t	Every 90 s
Message length s	4000 bytes
Number of digests advertised l	10 messages
Size of network coding generation G	50 messages
Interval of digest advertisement τ	Every 15 s
Interval of coded packets broadcast τ	Every 15 s
Generation allocation weight w	0.5, 1, 2

the higher layers.) We then simulated a 10-node network, where all nodes are identical hardware-wise and the data rate was 2 Mbps for Bluetooth EDR. After testing various combinations of d and p and measuring the message deliver extent, we found that when we gave the connection delay d a uniform value between 0 and 10 s in the experiment and set the link failure probability $p = 50\%$, 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 Fig. 11. Furthermore, we consolidated the data the same way as with real devices and summarize it in Table 5. We can see that the simulator yields very similar relative performance between network-coding and non-coding based approaches in this case.

Fig. 10. Broadcast extent for real devices.

Table 4
Number of messages reaching 1–4 and 5–9 devices.

Method	1–4 (%)	5–9 (%)
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

6. Concluding remarks

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 to the prohibitive cost of using real devices, both in terms of monetary and time, 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. Our first set of simulated experiments focused on an array of message prioritization techniques without network coding. We then compared network coding with these non-coding approaches through real-device experiments. There we observed that the performance gain of network coding over conventional data forwarding was evident. More importantly, we found that, after enhancing the link layer with a few necessary parameters to simulate the iOS Multipeer Connectivity Framework, tests using the ONE simulator would produce very similar performance to using the Apple iOS devices.

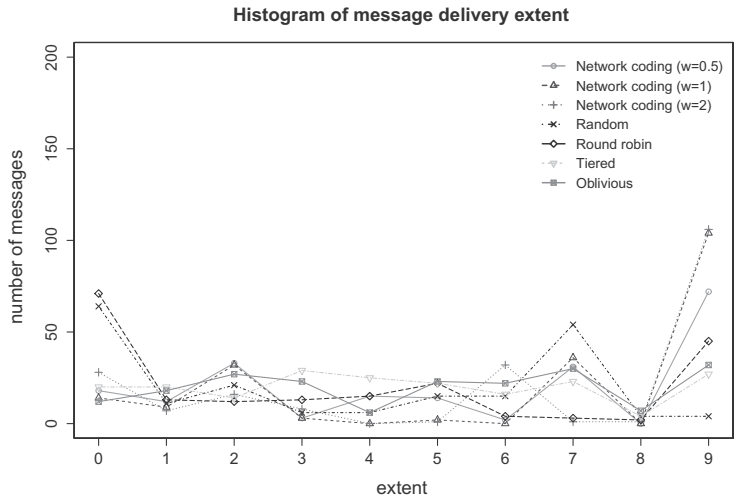


Fig. 11. Broadcast extent in simulation.

Table 5
Number of messages reaching 1–4 and 5–9 nodes in simulation.

Method	1–4 (%)	5–9 (%)
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

We now have more confidence that ONE would yield more reliable results in larger networks with such enhancement.

In future research, we would like to use the enhanced ONE simulator to study larger DTNs. For example, we know that 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 [27], and we intend to further investigate it using real devices. We also plan to test ONE against Android OS devices to find the best parameters in order to bridge the gap between these two as well. Knowing the much higher heterogeneity among devices on this platform, we expect more combination tests to achieve this goal. The benefit would be producing an even more trust-worthy simulator for future experimental research on DTN and network coding.

Acknowledgements

This work was supported by Natural Sciences and Engineering Research Council (NSERC, Canada) Discovery Grant 327667-2010, by National Institutes of Health (USA) Grants R01-LM009012, R01-LM010098, and R01-AI59694, and by National Natural Science Foundation of China Grants 91218301 and 60903201.

References

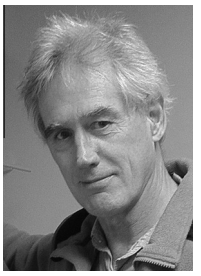
[1] Delay tolerant networking research group. <http://www.dtnrg.org/wiki/Home>.
[2] Haggle project. <http://code.google.com/p/haggle/>.
[3] The interplanetary network (IPN). <http://tmo.jpl.nasa.gov/>.
[4] The opportunistic network environment (ONE) simulator. <http://www.netlab.tkk.fi/tutkimus/dtn/theone/>.
[5] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, Raymond W. Yeung, Network information flow, IEEE Trans. Inform. Theory 46 (4) (2000) 1204–1216.
[6] John Burgess, Brian Gallagher, David Jensen, Brian Neil Levine, MaxProp: routing for vehicle-based disruption-tolerant networks, in: Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM), 2006, pp. 1–11.
[7] Szymon Chachulski, Michael Jennings, Sachin Katti, Dina Katabi, Trading structure for randomness in wireless trading structure for randomness in wireless trading structure for randomness in wireless opportunistic routing, in: Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2007, pp. 169–180.
[8] Yuanchu Chen, Walter Taylor, Sam Coxon, Jason H. Moore, isnac: Infrastructureless social networking at academic conferences, in: Demo at the 31st IEEE International Conference on Computer Communications (INFOCOM), IEEE, 2012.
[9] Philip A. Chou, Yunnan Wu, Kamal Jain, Practical network coding, in: Proceedings of Allerton Conference on Communication, Control, and Computing, 2003.
[10] Elizabeth M. Daly, Mads Haahr, Social network analysis for routing in disconnected delay-tolerant manets, in: Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), ACM, 2007, pp. 32–40.
[11] Vijay Erramilli, Mark Corvella, Augustin Chaintreau, Christophe Diot, Delegation forwarding, in: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), ACM, 2008, pp. 251–260.
[12] Christina Fragouli, Jean-Yves Le Boudec, Jorg Widmer, Network coding: an instant primer, SIGCOMM Comput. Commun. Rev. 36 (1) (2006) 63–68.
[13] Tracey Ho, Ralf Koetter, Muriel Medard, David R. Karger, Michelle Effros, The benefits of coding over routing in a randomized setting, in: Proceedings of IEEE International Symposium on Information Theory, IEEE, 2003.
[14] Pan Hui, Jon Crowcroft, Eiko Yoneki, BUBBLE Rap: social-based forwarding in delay tolerant networks, in: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, 2008, pp. 241–250.
[15] Sushant Jain, Kevin Fall, Rabin Patra, Routing in a delay tolerant network, in: Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM, 2004, pp. 145–158.
[16] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, Daniel Rubenstein, Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet, in: Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X), 2002, pp. 96–107.
[17] Maurice J. Khabbaz, Chadi M. Assi, Wissam F. Fawaz, Disruption-tolerant networking: a comprehensive survey on recent developments and persisting challenges, IEEE Commun. Surv. Tutorials 14 (2) (2012) 607–640.
[18] James F. Kurose, Keith W. Ross, Computer Networking: A Top-Down Approach, sixth ed., Pearson, Upper Saddle River, NJ, USA, 2012.
[19] Yunfeng Lin, Baochun Li, Ben Liang, Efficient network coded data transmissions efficient network coded data transmissions in disruption tolerant networks, in: Proceedings of the 27th IEEE International Conference on Computer Communications (INFOCOM), 2008.
[20] Anders Lindgren, Avri Doria, Olov Schelén, Probabilistic routing in intermittently connected networks, SIGMOBILE Mob. Comput. Commun. Rev. 7 (3) (2003) 19–20.
[21] Anders Lindgren, Pan Hui, The quest for a killer app for opportunistic and delay tolerant networks, in: Proceedings of the 4th ACM Workshop on Challenged Networks (CHANTS), ACM, 2009, pp. 59–66.
[22] Xu Liu, Yuanzhu Chen, Cheng Li, Walter Taylor, Jason H. Moore, Message prioritization of epidemic forwarding in delay-tolerant networks, in: Proceedings of International Conference on Computer Networking & Communications (ICNC), 2014.
[23] Muriel Medard, Alex Sprintson, Network Coding: Fundamentals and Applications, first ed., Academic Press, 2011.
[24] Mirco Musolesi, Cecillia Mascolo, CAR: context-aware adaptive routing for delay-tolerant mobile networks, IEEE Trans. Mob. Comput. 8 (2) (2009) 246–260.
[25] Thrasyvoulos Spyropoulos, Konstantinos Psounis, Cauligi S. Raghavendra, Spray and wait: an efficient routing scheme for intermittently connected mobile networks, in: Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking, ACM, 2005, pp. 252–259.
[26] Amin Vahdat, David Becker, Epidemic Routing for Partially-Connected Ad hoc Networks. Technical report CS-200006, Duke University, 2000.
[27] Jorg Widmer, Jean-Yves Le Boudec, Network coding for efficient communication in extreme networks, in: Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking, 2005, pp. 284–291.
[28] Jian Zhang, Yuanzhu Chen, Ivan Marsic, MAC-layer proactive mixing for network coding in multi-hop wireless networks, Comput. Networks 54 (2) (2010) 196–207.
[29] Zhensheng Zhang, Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges, IEEE Commun. Surv. Tutorials 8 (1) (2006) 24–37.

[30] Zhensheng Zhang, Qian Zhang, Delay/disruption tolerant mobile ad hoc networks: latest developments, Wireless Commun. Mob. Comput. 7 (10) (2007) 1219–1232.

Yuanzhu Chen is an Associate Professor and Deputy Head in the Department of Computer Science at Memorial University of Newfoundland, St. John's, Newfoundland. He received his Ph.D. from Simon Fraser University in 2004 and B.Sc. from Peking University in 1999. Between 2004 and 2005 he was a post-doctoral researcher at Simon Fraser University. He was also a Visiting Professor at Dartmouth College in 2011–2012. His research interests include computer networking, mobile computing, graph theory, Web information retrieval, evolutionary computation, and bioinformatics.

Xu Liu received the B.Eng. degree from Chengdu University of Information Technology, Sichuan, China, in 2010 and the M.Sc. degree from the Memorial University of Newfoundland, St John's, NL, Canada, in 2014. He is currently an iOS Lead Developer of GreenOwl Mobile in Toronto, Canada.

Jiafen Liu graduated from Department of Computer Science, University of Electronic Science and Technology of China, and was awarded Doctor's degree in 2008. She now works as associate professor in School of Economic Information Engineering, Southwestern University of Finance and Economics (SWUFE), Chengdu, China. Her research interests includes information security, wireless network and mobile commerce.



Walter Taylor holds a BA degree from Washington University, St. Louis, MO, and Ph.D. degree from the University of Michigan, Ann Arbor Michigan. He has held a number of research positions from 1979 through 1997 that focused on the molecular mechanism of circadian rhythms. During that time, developed a strong interest in software development for controlling various types of biological instrumentation, which eventually led to stints as a software engineer at 3 companies well-known for their advances in DNA sequencing technology, including Applied Biosystems, 454 Life Sciences, and Helicose. From 2009 through the present, he has focused on bioinformatics support for the sequencing core facility at the Dartmouth Medical School.

Jason Moore is the Edward Rose Professor of Informatics and Director of the Institute for Biomedical Informatics at the Perelman School of Medicine at the University of Pennsylvania. His research interests include artificial intelligence, bioinformatics, complex adaptive systems, machine learning, and network science. He serves as Editor-in-Chief of the journal BioData Mining. He is an elected Fellow of the American Association for the Advancement of Science and a Kavli Fellow of the National Academy of Sciences.