# ABSTRACT

Title of Thesis: PERFORMANCE EVALUATION OF NULLSPACE STOPPING CONDITION INCORPORATING NETWORK CODING IN DELAY TOLERANT NETWORKS

Wei Bai, Master of Science, 2015

Thesis directed by: Professor Richard La
Department of Electrical and Computer Engineering

For delay tolerant networks (DTNs), since there is no guarantee of end-to-end path from a source to a destination, routing protocols should make use of opportunistic contacts to deliver files. Although protocols employing network coding have been shown to achieve promising results in DTNs, they still suffer from redundant transmissions. An efficient stopping condition utilizing nullspace has been proposed recently. But more comprehensive studies are needed. In this thesis, a systematic research on effectiveness and efficiency of nullspace stopping condition is explored. We propose a novel algorithm to calculate nullspace. Using comprehensive simulations, we show that the benefits of nullspace stopping condition to network coding depend on scenarios. Moreover, performances may vary even in the same scenario with respect to the number and size of disseminated files. Finally explanations about these phenomena are given out.

# PERFORMANCE EVALUATION OF NULLSPACE STOPPING CONDITION INCORPORATING NETWORK CODING IN DELAY TOLERANT NETWORKS

by

Wei Bai

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2015

Advisory Committee:
Professor Richard La, Chair/Advisor
Professor Gang Qu
Dr. Greg Stein

# Acknowledgments

I owe my gratitude to all the people who have made this thesis possible.

First and foremost, I would like to thank my supervisor, Prof. Richard La for his great scientific guidance. Richard's vast experience in research, combined with his kindness, smile and sincerity, taught me how to produce high-quality work with a positive attitude, always being precise, objective and very self-critical. More importantly, he provided very constructive feedback in sharping my research philosophy: finding a problem, formulating the problem and solving the problem. I would also thank him for revising the thesis with great responsibility, even taking care of grammar details, through which my writing skills improved significantly.

Next, I would like to thank Dr. Greg Stein, formerly in Laboratory for Telecommunication Sciences, and currently in Trusted System Research. As the research scientist for the cutting edge technologies, he broadened my horizon to great extent. Especially, he introduced a lot to me how the research would be applied in real world. Simultaneously, he provided me with many innovative ideas where I benefited from. I would also appreciate him for participating our afternoon meetings in the university.

Then I would like to thank Dr. Ginnah Lee for her previous excellent work, where my research project is based. She had established an excellent simulation framework for my research, and provided many interesting results for use. She also taught me a lot about programming skills.

I would also like to thank Prof. Qu Gang, as my committee member and

providing valuable feedback for my research and thesis. Additionally, I could not forget great support from Staffs in ECE department, especially Mr. Bill Churma and Mrs. Heather Stewart to arrange the defense.

Last but not least, I dedicated this thesis to my parents, for their unconditional love during all these years, especially for their great support for my decision pursuing higher levels of education in the US. I am grateful for everything they sacrificed for my upbringing and education.

# Table of Contents

# List of Figures

# Chapter 1: Introduction

The increasing interest in delay tolerant networks (DTNs) [1] has heightened the need for efficient protocols to transmit data from sources to destinations in challenging environments, such as mobile ad hot networks (MANETs), interplanetary internet, military ad hoc networks and wildlife tracking sensor networks. This kind of networks are disruptive due to sparsity of mobile nodes, constrained energy resources, the limits of wireless radio range, etc. Therefore, traditional network solutions, such as Internet protocols, fail to support DTNs since there is no guarantee of contemporaneous end-to-end connectivity between a source and a destination.

Instead, a different network framework [2] [3] is established to provide solutions for DTNs, and a variety of routing protocols have been proposed based on this framework. Among various techniques, coding schemes have shown promising prospects because of its adaptability to change in network topology and low overhead cost. Coding schemes include source coding, where only source nodes generate encodings, and network coding that allows intermediate nodes to recombine received encodings. A closed-form expression showing the relation between the performance of DTNs and the coding that is used is provided in [19]. The authors in [18] present a coded forwarding protocol, which jointly considers forwarding schemes and the use

of fountain code. The performance of random linear coding for unicast scenarios in DTNs are evaluated in [24]. An analytical model is developed in [25] for network coding in DTNs to demonstrate benefits in resource-constrained situations. Coding schemes are shown to be feasible in [26], where SimpleNC, a network coding router for the DTN2 Reference Implementation, was built. The authors of [27] extended SimpleNC in a way that the encountered nodes exchange encodings while taking into account the ranks of their encoding matrics.

Although coding schemes provide a powerful tool for distributing information, unnecessary coding process and transmissions can lead to a waste of resources. A stopping condition utilizing nullsapce is developed in [32] to reduce redundant transmissions and control data flow, where the proposed protocol is based on [26] and [27]. Aware of the mathematical structure of the underlying code, this stopping condition is based on the nullspace of the space spanned by encoding vectors [33]. Two nodes that meet will exchange nullspace bundles first, and then determine whether to send encoded information to each other or not based on the received nullspace bundle. However, the experimental results provided in [32] are limited. Also, the network model is fairly simple, including only three nodes - a source, a destination and an intermediate node. In an island hopping scenario, very few number of bundles with small sizes are generated during an experiment, and also the group rank may not be a good metric since we are usually concerned with whether or not the bundle is received by the destination node, not collectively by a group of nodes.

Therefore, considering the limitations of previous research, in this thesis, we

evaluate nullspace stopping condition more systematically. The contributions in this thesis are as follows.

- A more realistic DTN simulator, the ONE simulator [47], is used to evaluate performances of nullspace stopping condition in two representative scenarios: urban scenario and island hopping scenario. In each scenario, multiple relatively small files or one large file is generated to simulate different situations.

- In creating nullspace bundle list, a simplified algorithm to calculate nullspace matrix based on the reduced row echelon form of the encoding matrix is proposed to reduce computational complexity.

Our simulations show that the effectiveness and efficiency of nullspace stopping condition vary depending on scenarios. Even in the same scenario, the performances diverge if different numbers and sizes of files are generated by the source. The reminder of this thesis is structured as follows. Various DTN routing protocols are introduced in chapter 2. In chapter 3, the nullspace stopping condition is described, and the algorithms to create and update nullspace bundle list are detailed. The ONE simulator as the simulation platform for our research is introduced in chapter 4. Simulation results and discussions are shown in chapter 5, and chapter 6 concludes the thesis.

# Chapter 2: DTN Routing Protocols

In this section, various routing protocols in DTNs are introduced. First, different categories of routing protocols are reviewed in general. Then, epidemic routing and coding schemes are described in detail. Finally, an immunity mechanism is described.

## 2.1 Overview

Since there is no guarantee of end-to-end connection from a source to a destination, DTN protocols should make use of opportunistic contacts between encountered nodes to deliver files or messages. DTN routing protocols can be categorized into two classes: forwarding-based protocols and replication-based protocols. The first keeps only one copy of the message, and forwards it to the destination at each contact. The second produces several replicas of each unique message in the network in hopes of increasing the message delivery ratio.

As examples of forwarding-based protocols, traditional routing protocols, such as AODV [4] and IP [5], are not applicable in DTNs since there is no guarantee of end-to-end connection between a source and a destination. Several forwarding-based protocols in DTNs have been investigated [6] [7] [8]. The routing issues in DTN are

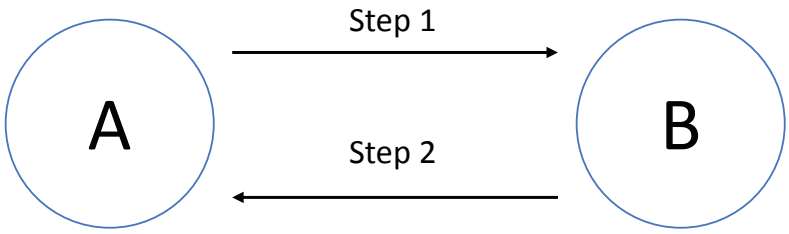
first formulated in [6], where network connectivity patterns are known. A similar reference [8] also proposes a model for nodes to make a series of independent, local forwarding decisions through DTNs. However, their models are based on both current connectivity and predictions of future connectivity information, which may be available only to a certain class of DTNs (e.g., a scheduled bus network). The proposed scheme in [7] mimics routing table construction in DTNs similar to caching in program execution. But, it makes a strong assumption that node movements are recurrent to guarantee bounded worst-case performance, which limits its effectiveness to more general movement models.

Replication-based protocols can be further categorized into two sub-classes: flooding-based and quota-based. Compared to forwarding-based protocols, since several copies of the message are flowing in the network, there is a trade-off for replication-based protocols between improvement of delivery probability and resources consumption (e.g., buffer size, bandwidth). The difference between flooding and quota based routing protocols lies in the number of message replicas. Flooding-based protocols send a new copy of a message following every node contact, while for quota-based protocols, the total number of replicas in the network for each unique message is limited to $L$, where $L$ can be either a fixed number or a discrete variable. Therefore, a replication-based routing protocol is quota-based if and only if $L$ is independent of the number of nodes in the network; otherwise it is flooding-based [9].

## 2.2 Epidemic Routing

In this section, epidemic routing [10] is introduced as a representative of flooding-based protocols. For epidemic routing, ideally every node will have a replica of each unique message created by sources. In this protocol, each node maintains a buffer which consists of messages it has received. To reduce unnecessary resource consumption, only one replica of each unique message is kept at each node. A bit vector, called *summary vector*, is used to indicate messages stored at every node. Also, a Bloom filter [11] can be employed to reduce the number of bytes required to represent the summary vector. A Bloom filter is a space-efficient probabilistic data structure. An empty Bloom filter is an all-zero bit array of $m$ bits. Then $k$ different hash functions are defined, each of which hashes some set element to one of the $m$ array positions with a uniform distribution. An element is added by feeding it to each of the $k$ hash functions to get $k$ array positions, and setting the bits of all these positions to 1. To query for an element, similar to the process of adding element, it is fed to each of the $k$ hash functions to get $k$ array positions. If any of the bits at these positions is 0, the element is definitely not in the set (thus no false negatives [1]). If all positions are 1, then either the element is in the set, or the bits are happened to be set to 1 during insertion of other elements, resulting in false positives. The more elements are added into the set, the larger the probability of false positives.

[1]There are two types of errors: false positives and false negatives (or type I and type II errors). False positives are errors that detect an event that is not present; while false negatives are errors that fail to detect an event that is present.




Step 1: Node A sends its summary vector $SV_A$ to node B;
Step 2: Node B determines what A lacks ($\overline{SV_A} \cap SV_B$), and transmits to node A.
*$SV_A$ represents summary vector of node A

Figure 2.1: Message exchange of epidemic routing

To reduce redundant transmissions, when two nodes meet at each contact, there are three steps to follow in order to exchange messages [10]. First, the two nodes that meet exchange their summary vectors to determine what messages they are missing and should receive from the other node. Second, each node send an acknowledgment to its counterpart to request missing messages. Finally, each node transmits the requested messages to the other node. Based on our observations, step two can be skipped, since from the summary vector received from the other node, a node itself can determine what messages its counterpart lacks, and then transmits these messages. Figure 2.1 depicts an example of message exchange in the epidemic routing protocol.

Epidemic routing is one of the commonly used protocols in DTN research due to its simplicity of implementation and little required knowledge about underlying network topology. We adopt it as one basic protocol for our research.

## 2.3 Coding Schemes

Coding-based protocols have shown great potential in DTNs. The basic idea of coding scheme is that when a file is to be sent from a source, it will be first chopped into smaller chunks. If these chunks are sent out through a flooding protocol, the collection of all distinct chunks at the destination suffers from coupon collector's problem [12], i.e., the first few chunks will reach destination fairly quickly, while it will take long time to collect the last few required ones. Suppose a file is chopped into $n$ chunks, the expected number of chunks destination needs to receive all $n$ distinct chunks is $O(n \log(n))$, if all the chunks are received equally likely and with replacement. Thus, to overcome this problem, coding schemes allow sources and intermediate nodes to perform coding operations, combining different chunks before disseminating. The destination only needs to receive $n$ linearly independent encoded chunks (or encodings) to recover the original file. Based on which nodes produce new encodings, coding schemes can be categorized into two sub-classes: (1) source coding, or erasure coding, where only sources generate new encodings; and (2) network coding or recoding, where intermediate nodes generate new combinations of received encodings, which further increases combination of information in the network.

Some terminologies related to coding schemes are listed below.

**File**: a message which is created at source nodes. It will be chopped if it is too big compared to contact time and transmission rates. Each file will be assigned a universally unique identifier (UUID).

**Chunk**: a fragment $\mathbf{f_i}$ of one file. As a part of the file, it is associated with the same UUID as original file. Denote $n$ be the total number of chunks of one file that is chopped.

**Coefficient vector**: a vector $\boldsymbol{\alpha} =< \alpha_1, \ldots, \alpha_n >$ controls what chunks to be used to create new encodings at the source. Elements of $\boldsymbol{\alpha}$ is usually chosen from GF$(2)^n$. The encodings are denoted by $\mathbf{c} = \sum_1^n \alpha_i \mathbf{f_i}$.

**Network-coding vector**: suppose one intermediate node has received $t$ encodings for one UUID. Then network-coding vector $\boldsymbol{\beta} =< \beta_1, \ldots, \beta_t >$ chooses encodings on this intermediate node to perform coding to create a new encoding $\mathbf{d} = \sum_1^t \beta_i \mathbf{c_i}$. Note that only encodings with the same UUID are used in network coding, i.e., two encodings with different UUID cannot be recombined.

### 2.3.1 Source Coding

The concept of source coding is to deliberately add redundancy before forwarding in order to improve performances. Previously, files are chopped into smaller chunks, which are then transmitted independently over network. With source coding, the source node disseminates encodings rather than original chunks, usually more than $n$. Therefore, some of them are redundant. The receiver can recover the original file with high probability if it can receive slightly more than $n$ encodings. Two categories of source codes have been proposed: erasure codes and fountain codes [16]. Compared to the fixed redundancy in erasure codes, fountain codes are rateless, and the original file can be recovered provided that the encoding vectors

of received encodings form a full rank matrix. Several coding algorithms have been investigated, such as linear random codes, tornado codes [13], LT codes [14], raptor codes [15], etc. There is a trade-off between coding/decoding efficiency and the number of encodings to be collected by the receiver. Some surveys and theoretical analysis about encoding/decoding complexity can be found in [16] and [17].

Source coding is first employed in DTN in [18], where a coded forwarding protocol is proposed to integrate fountain codes and optimal probabilistic forwarding together. It is shown that the proposed protocol performs better than epidemic and optimal probabilistic forwarding (OPF) protocols using trace data from UMass-DieselNet. A closed-form expression for the performance in terms of delivery ratio and energy consumption of DTNs as a function of the coding used is provided in [19], and the existence of phase transitions in coding schemes is also found. A fountain-coding based transport protocol DTTP is proposed in [20], and simulations using the ONE simulator validates performance improvement by introducing fountain codes.

The effects of different source codes in DTNs are out of scope of this research. We adopt the random linear codes for this study.

### 2.3.2 Network Coding

One shortcoming of epidemic routing is the high bandwidth consumption during exchange of summary vectors, especially when the total number of encodings is large. Network coding [21], by combining encodings at intermediate nodes, has shown great potential due to its resistance to change of network topology and vari-

ous network attacks [22] [23]. Also, protocols with network coding does not require exchange of summary vectors. In this thesis, we deliberately distinguish network coding from source coding (in which only source nodes perform encoding), which has been discussed in section 2.3.1.

Network coding has been researched extensively in DTN protocols. [24] evaluates the benefits of random linear coding for unicast scenarios in DTNs when bandwidth and buffer space are constrained. The efficiency of network coding is also validated in [25], where information-theoretical analysis is provided. [26] proposes a network coding router for the DTN2 Reference Implementation, SimpleNC, to show that network coding is practically achievable, and the performance of network coding is evaluated in island hopping scenarios. [27] extends the protocol proposed in [26] by considering rank information of encountered nodes, and implemented a Context-Aware Network-Coded (CANC) context agent to control message exchanges.

Recall that in network coding, intermediate nodes choose a random set of stored encodings and combine them to create a new encoding. The new encoding is $\mathbf{d} = \sum_1^t \beta_i \mathbf{c_i}$, where $\mathbf{c_i}\,(i = 1, \ldots, t)$ are stored encodings, and $\boldsymbol{\beta} =< \beta_1, \ldots, \beta_t >$ is network-coding vector. Each $\beta_i$ is chosen from finite field $\mathbf{GF(2)}$. Denote $W_t$ to be the hamming weight of $\boldsymbol{\beta}$, i.e., the number of $\mathbf{1}$s in $\boldsymbol{\beta}$. In the research, the weight of $\boldsymbol{\beta}$ is limited to reduce coding complexity.

For routing protocols employing network coding, apart from the destination node, there are two strategies available for intermediate nodes to receive encodings. One is that intermediate nodes perform rank check every time they receive a new encoding. This encoding will be accepted only if it increases the rank of encoding

matrix of corresponding UUID; otherwise it will be discarded which implies that it is redundant. The other strategy does not need rank check; instead, every node chooses to receive more than $n$ encodings for each UUID. The number of extra encodings a node is willing to accept is denoted by $\epsilon$. In the thesis, both of the strategies will be adopted.

### 2.3.3 Epidemic routing with coding schemes

Epidemic routing provides a way to prevent unnecessary transmissions between nodes by exchanging their summary vectors first. It can be equipped with source coding. However, exchange of summary vectors is not very helpful when employing network coding. This is because even when two nodes have the equivalent encodings, in the case of network coding, the summary vectors will still say that they are innovative to each other. Network coding is considered to be different from epidemic routing in the following two aspects. On one hand, network coding exempts from exchanging summary vectors, which reduces resource consumption; on the other hand, there is no effective and efficient stopping condition for network coding so far to prevent redundant transmissions.

## 2.4 Immunity Mechanism

Immunity mechanism is considered to further reduce storage requirement and energy consumption. As claimed in epidemic routing in section 2.2, nodes will continue to exchange messages even though these messages are successfully delivered

to the destination, until all the nodes receive a copy of each message or timeout is triggered. Immunity mechanism is another method to prevent unnecessary copying of the message if it has been delivered.

The idea of immunity mechanism is first introduced in network analysis in [30], where a Markov chain model is proposed to evaluate the impact of immunity in sensor networks. In a similar research using Markov chain model [29], (p-q)-epidemic routing in sparsely populated ad hoc networks is investigated, taking into account immunity scheme (the authors used VACCINE in the paper), where it is revealed that the performance is optimal with small value $p$ when $q = 1$ and the VACCINE scheme is employed. [31] explored several performance metrics in epidemic routing, illustrating the differences among various forwarding and recovery schemes considered using ordinary differential equation (ODE) models, yielding closed-form expressions. [28] proposed an immunity based epidemic routing in DTN, and evaluated performances with network simulator ns2. Nodes process transmissions and drop encodings (without network coding) based on m-list and i-list, where m-list is a list resembling a summary vector in epidemic routing, and i-list indicates what encodings are successfully delivered. The two nodes that meet will update their i-lists first by combining the two i-lists into one, and delete all unwanted bundles according to this common i-list. It is discovered that by better utilizing buffer, the fraction of delivered messages at lower delays can be increased.

In this thesis, we adopt an immunity mechanism similar to [28]. The destination node will create a list of UUIDs of delivered files instead of encodings, called "delivered file list", and pass this list to every node it meets. The two nodes upon

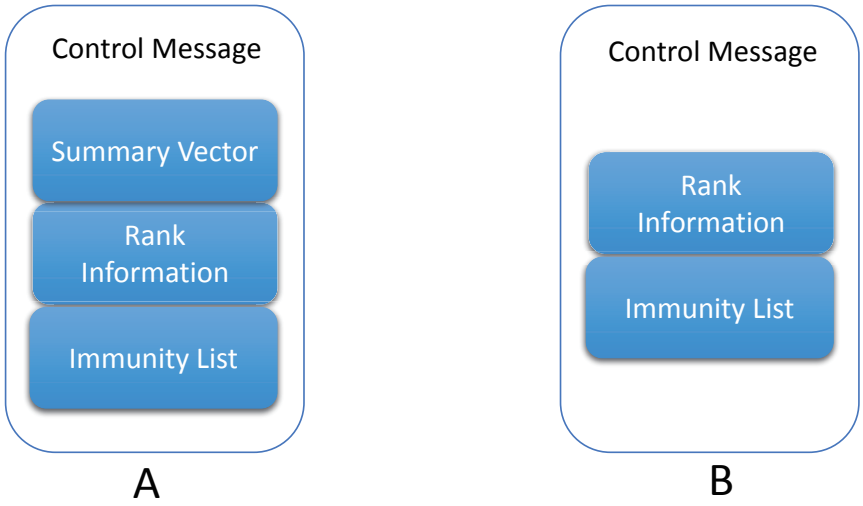



Figure 2.2: Structure of Control Message

a contact merge their delivered file lists first. Then, all the encodings produced from files in the buffer which are on the immunity list will be deleted. Furthermore, they refuse to accept any copy of encodings associated with these UUIDs from other nodes. Therefore, the node storage is better utilized by discarding the "immunized" files in the buffer, and energy is saved by stopping needless transfer of delivered files.

To sum up, the encountered two nodes will exchange control messages first before transmitting encodings. The structures of control message adopting different types of protocols are presented in Figure 2.2.

# Chapter 3: Nullspace Stopping Condition for Network Coding in DTN

In this section, an efficient stopping condition which is based on nullspace structure is described.

## 3.1 Nullspace Stopping Condition

Although network coding has shown potential to improve the performance in DTNs, it still suffers from lacking an efficient stopping condition to prevent exchange of redundant messages. In general, one node is said to be innovative to its neighbor if it can increase the rank of the encoding matrix of that neighbor node. However, in some simple circumstances, the subspaces spanned by encoding matrices of two encountered nodes are identical, Thus neither node is innovative to the other. In this case, transmission of encodings between them is a waste of resources. Therefore, an efficient stopping condition utilizing nullspaces is proposed in [32]. This idea has been used as an analytical tool for network coding gossip protocols in [33], and [32] proposes a mechanism taking consideration of it.

Some notations are described as follows. Denote $\mathbf{Y_{A,x}}$ to be the subspace of $\mathbf{GF(2)^n}$ spanned by the rows of the encoding matrix $\mathbf{E_{A,x}}$ of node A for UUID

x. The nullspace of the encoding matrix is denoted by $\mathbf{Y}^{\perp}_{\mathbf{A},\mathbf{x}}$, which consists of all vectors that are orthogonal to $\mathbf{E}_{\mathbf{A},\mathbf{x}}$. We denote by $\mathbf{N}_{\mathbf{A},\mathbf{x}}$ the matrix consisting of basis of the nullspace $\mathbf{Y}^{\perp}_{\mathbf{A},\mathbf{x}}$. It is known that the dimension of $\mathbf{Y}_{\mathbf{A},\mathbf{x}}$ plus that of $\mathbf{Y}^{\perp}_{\mathbf{A},\mathbf{x}}$ is equal to $n$.

For notational simplicity, we only consider one single file with UUID $x$. Once two nodes $A$ and $B$ meet, $A$ will transmit encodings to its counterpart if and only if it is innovative with respect to node $B$. In terms of nullspaces, it requires $\mathbf{Y}^{\perp}_{\mathbf{B},\mathbf{x}} \not\subseteq \mathbf{Y}^{\perp}_{\mathbf{A},\mathbf{x}}$, or equivalently, $\mathbf{Y}_{\mathbf{A},\mathbf{x}} \not\subseteq \mathbf{Y}_{\mathbf{B},\mathbf{x}}$.

Though each node can exchange a full basis of nullspace for every file to determine the condition, the overhead may be quite high. For each file with $n$ chunks, this may require up to $n^2$ bits. Therefore, a random linear projection of $\mathbf{Y}^{\perp}_{\mathbf{A},\mathbf{x}}$ and $\mathbf{Y}^{\perp}_{\mathbf{B},\mathbf{x}}$ is exchanged instead. Admittedly, there is information loss using this linear projection instead of a full basis, which causes false positives. However, error analysis in section 3.4 shows that this error probability can be made relatively small.

Nullspace bundle list (**NSBL**) is the payload exchanged between two nodes to determine innovation. The structure of **NSBL** from node $B$ to node $A$ is presented in Figure 3.1. It contains a list of nullspace bundles (**NSB**) for every file with UUID $x_i$. Each **NSB** consists of UUID $x_i$, the rank of the encoding matrix $\mathbf{E}_{\mathbf{B},\mathbf{x_i}}$, a fixed number of linear projection vectors in $\mathbf{Y}^{\perp}_{\mathbf{B},\mathbf{x_i}}$, and the number of vectors and vector size.

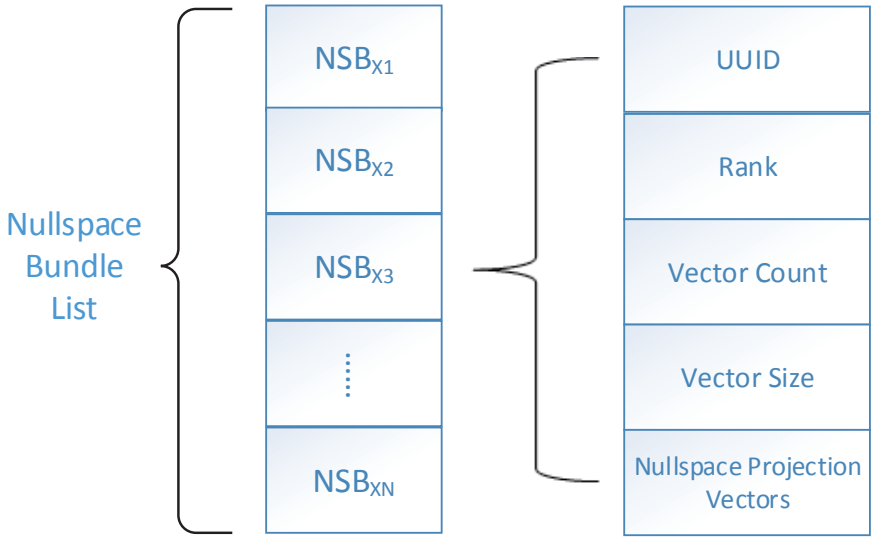



Figure 3.1: Structure of Nullspace Bundle List

## 3.2 Algorithm to Create Nullspace Bundle List

As shown in Figure 3.1, nullspace bundle list is comprised of several nullspace bundles, and creating a nullspace bundle requires calculation of nullspace basis. Define **E** and **N** to be encoding matrix and the matrix consisting a basis of nullspace, respectively. We first provide a recursive algorithm (algorithm 1) to calculate **E** after receiving a new encoding vector $v$, where **E** is in reduced row echelon form (i.e., the leading coefficient is 1 and is the only nonzero entry in its column), but not necessarily in the standard form (i.e., the matrix is not necessarily in upper triangle form).

If **E** is in standard reduced row echelon form, there are various algorithms to calculate **N** (e.g. [34]). However, to revert **E** into standard form requires multiple row swaps, which brings about higher computational complexity. Therefore, we

**Algorithm 1** Algorithm for updating encoding matrix

```
if rank != n then
    E[rank] ← v
    for i = 1 to rank do
        if E(rank, L[i]) == 1 then
            E[rank] ← E[rank]xorE[i]
        end if
        i ← 1 and flag ← false
        while flag is false do
            if i == n+1 then
                flag ← true
            else if E(rank, i) == 1 then
                flag ← true and L[i] ← i
                for j = 1 to (rank) do
                    if E(j, L[rank]) == 1 then
                        E[j] ← E[j]xorE[rank]
                    end if
                end for
                rank ← rank + 1
            end if
            i ← i + 1
        end while
    end for
end if
```

propose a novel but simple way to obtain **N** directly from reduced row echelon form of **E**, which is shown in algorithm 2.

**Algorithm 2** Algorithm for obtaining **N** from **E**

```
if rank == n then
    N ← n × n identitymatrix
else
    for i = 1 to (n − rank) do
        for j = 1 to (rank) do
            if E(j, S[i]) == 1 then
                N(i, L[j]) ← 1
            elseN(i, L[j]) ← 0
            end if
        end for
        N(i, S[i]) ← 1
        for l = 1 to (n − rank) do
            if l! = i then
                N(i, S[l]) ← 0
            end if
        end for
    end for
end if
```

## 3.3 Determining Innovation Using Nullspace Bundles

Once node $A$ receives a nullspace bundle list from node $B$, a function will be called to determine whether or not node $A$ is innovative for each UUID it is carring. For notational simplicity, we only consider one UUID, $x$. Also denote by $v_1, v_2, \ldots v_t$ $t$ random nullspace projection vectors in $\mathbf{NSB_{B,x}}$. Recall that node $A$ concludes that it is innovative to node $B$ if and only if $\mathbf{Y}^{\perp}_{\mathbf{B,x}} \not\subseteq \mathbf{Y}^{\perp}_{\mathbf{A,x}}$. The algorithm for node $A$ to determine if it is innovative to $B$ is presented in algorithm 3.

**Algorithm 3** Algorithm for node $A$ determine innovation to node $B$

```
if rank(E_{A,x}) > NSB_{B,x}.rank then
    return true
else
    for all v_i ∈ NSB_{B,x} do
        if E_{A,x} · v_i ≠ 0 then
            return true
        end if
    end for
    return false
end if
```

## 3.4 Error Analysis of Nullspace Stopping Condition

Suppose node $A$ receives a nullspace bundle, $\mathbf{NSB}_{B,x}$ from node $B$, and will determine whether or not it is innovative to $B$. As mentioned in section 3.1, random linear projection vectors of $\mathbf{Y}^{\perp}_{\mathbf{B},\mathbf{x}}$ are chosen instead of the full basis, which will bring about erroneous decisions. We will analyze this error probability in this section.

Recall that there are two types of errors: false positives and false negatives (or type I and type II errors). Specifically in our analysis, false positive is the probability that node $A$ concludes it is not innovative to $B$ while in fact it is, and false negative is the probability that node $A$ concludes it is innovative to $B$ while in fact it is not.

Note that there is no false negatives. If node $A$ is not innovative to $B$, i.e., $\mathbf{Y}_{\mathbf{A},\mathbf{x}} \subseteq \mathbf{Y}_{\mathbf{B},\mathbf{x}}$, or equivalently $\mathbf{Y}^{\perp}_{\mathbf{B},\mathbf{x}} \subseteq \mathbf{Y}^{\perp}_{\mathbf{A},\mathbf{x}}$, then for every $v \in \mathbf{Y}^{\perp}_{\mathbf{B},\mathbf{x}}$, we all have $\mathbf{E}_{A,x} \cdot v_i = 0$. Algorithm 3 will correctly halt transmission from $A$ to $B$.

Therefore, only false positives will occur, i.e., in fact $\mathbf{Y}^{\perp}_{\mathbf{B},\mathbf{x}} \not\subseteq \mathbf{Y}^{\perp}_{\mathbf{A},\mathbf{x}}$, but for all linear projection vectors $v_i, \forall i = 1, \cdots, t$ independently chosen from $\mathbf{Y}^{\perp}_{\mathbf{B},\mathbf{x}}$, $\mathbf{E}_{A,x} \cdot v_i = 0, \forall i = 1, \cdots, t$. This will happen if $v_i \in \mathbf{Y}^{\perp}_{\mathbf{A},\mathbf{x}} \cap \mathbf{Y}^{\perp}_{\mathbf{B},\mathbf{x}}, \forall i = 1, \cdots, t$. Suppose $n_{a,b}$ is

the dimension of $\mathbf{Y}_{\mathbf{A},\mathbf{x}}^{\perp} \cap \mathbf{Y}_{\mathbf{B},\mathbf{x}}^{\perp}$, and $n_a$ and $n_b$ are the dimensions of $\mathbf{Y}_{\mathbf{A},\mathbf{x}}^{\perp}$ and $\mathbf{Y}_{\mathbf{B},\mathbf{x}}^{\perp}$, respectively. Then, the false positive probability is

$$\begin{aligned} P_e &= P\left(v_1, \ldots, v_t \in \mathbf{Y}_{\mathbf{A},\mathbf{x}}^{\perp} \cap \mathbf{Y}_{\mathbf{B},\mathbf{x}}^{\perp} | v_1, \ldots, v_t \in \mathbf{Y}_{\mathbf{B},\mathbf{x}}^{\perp}\right) \\ &= \prod_{1}^{t} P\left(v_i \in \mathbf{Y}_{\mathbf{A},\mathbf{x}}^{\perp} \cap \mathbf{Y}_{\mathbf{B},\mathbf{x}}^{\perp} | v_i \in \mathbf{Y}_{\mathbf{B},\mathbf{x}}^{\perp}\right) \\ &= \left(\frac{dim\left(\mathbf{Y}_{\mathbf{A},\mathbf{x}}^{\perp} \cap \mathbf{Y}_{\mathbf{B},\mathbf{x}}^{\perp}\right)}{dim\left(\mathbf{Y}_{\mathbf{B},\mathbf{x}}^{\perp}\right)}\right)^t \\ &= \left(\frac{n_{a,b}}{n_b}\right)^t \end{aligned} \tag{3.1}$$

The worst case occurs when the nullspaces of nodes $A$ and $B$ are almost identical, i.e, they have the property of $n_b - n_{a,b} = 1$. In this case, however, $P_e = \left(\frac{1}{2}\right)^t$, which decays exponentially with the number of projection vectors. The error probability is relatively low even if $t = 3$ is adopted.

## 3.5 Update Implementation

In network coding with nullspace stopping condition, the structure of control message is presented in Figure 3.2, where a control message includes NSBL. When nodes decide to send control messages, they should ensure that every file has an up-to-date encoding matrix as well as a nullspace basis. Using algorithms 1 and 2, however, nullspace basis matrix $\mathbf{N}$ for each UUID has to be calculated every time a new encoding arrives. This is unnecessary, though, since nullspace basis matrix $\mathbf{N}$ needs to be updated only when the rank of encoding matrix $\mathbf{E}$ is changed; otherwise, $\mathbf{N}$ remains the same. Therefore, algorithm 1 is revised to introduce an


Figure 3.2: Structure of Control Message with Nullspace Stopping Condition

indicator **isInno** to indicate rank update. This is shown in algorithm 4 below.

After creating nullspace bundles, the indicator **isInno** is set to be false again, waiting for next update.

**Algorithm 4** Algorithm for updating encoding matrix with update implementation

```
if rank != n then
    E[rank] ← v
    for i = 1 to rank do
        if E(rank, L[i]) == 1 then
            E[rank] ← E[rank]xorE[i]
        end if
        i ← 1 and flag ← false
        while flag is false do
            if i == n+1 then
                flag ← true
            else if E(rank, i) == 1 then
                flag ← true and L[i] ← i
                for j = 1 to (rank) do
                    if E(j, L[rank]) == 1 then
                        E[j] ← E[j]xorE[rank]
                    end if
                end for
                rank ← rank + 1
                isInno ← true
            end if
            i ← i + 1
        end while
    end for
end if
```

# Chapter 4: The ONE Simulator

In this chapter, the Opportunistic Networking Environment (ONE) simulator is introduced, which is used as the tool for DTN performance evaluation.

## 4.1 Overview for DTN Simulators

For research on DTN, simulation plays an important role in analyzing routing protocol behaviors. Performances vary significantly according to how the nodes move, how many nodes are in the simulation world, how nodes communicate and transmit messages, etc. Therefore, the closer the settings under which protocols are evaluated to real-world scenarios, the more reliable simulation results may be.

Various network simulators have been proposed so far. For example, ns-3 [35] and OMNet++ [36] provide open simulation platforms for packet-based communications, specifically for MANETs. So do some other tools such as JANE [37]. However, their generic support for DTN is relatively limited. Although ns-3 also holds some openly available DTN simulators (dtnsim [38] and dtnsim2 [39]), only DTN routing protocols are available, while other important features such as data input and event generation are omitted.

Another essential aspect for DTN simulation is mobility modeling, which de-

fines nodes' movement, their population density, their contact times, etc. The mobility data can be generated by collecting real-world traces, e.g., CRAWDAD project [40]. However, there still exist some limitations for DTN simulation. First, the population analyzed in these traces is naturally fixed and limited. Once the trace is gathered, the number of participants cannot be adjusted, while DTN is often scalable, and actual population can be much larger than provided. Furthermore, the time granularity is often limited in order to save battery power on mobile devices. [41], for example, uses sensing intervals of 5 mins. Although it can to some extent reflect energy constrains, many contact opportunities may be missed, and only coarse levels of contact times can be recorded. Also, some traces are highly specialized, i.e., collected by a group of people in a certain situation (e.g., people attending a conference). The behavior of one group may not be applicable for other situations, thus making the simulation scenarios quite limited. So, real-world traces do not offer a wide range of mobility scenarios.

Therefore, the only option for deriving flexible and scalable mobility data is by establishing model-based mobility. The mobility models have been researched extensively, from the simplest models such as the Random Waypoint (RWP) and Markov-Gaussian Mobility Model [42], to group mobility models such as Reference Point Group Mobility Model (RPGM), to urban network mobility model considering real street maps [43]. In these models, the number of nodes as well as their behaviors can be changed according to different simulations. For example, [44] proposed a mobility model in which node velocity, wait time, etc. could be adjusted to match vehicles, pedestrians and other node types, and also moving features such as smooth

turns, speed variation could be added, which makes simulation closer to real world scenario.

It would be inconvenient for users to get intuitive sense about node mobility and message transfer without visualization. A Graphical User Interface (GUI) is an efficient tool for this purpose. iNSpect [45] can be used for ns-2 simulations. For DTN routing simulators, however, there are no similar tools available.

Due to integrated support for DTN routing, capabilities for mobility modeling and realization of visualization, the ONE simulator is a popular tool for DTN simulations. Some of the features of the ONE simulator are listed below:

- It is specifically designed for evaluating DTN routing algorithms and application protocols.
- Scenarios based on different synthetic movement models and real-world traces are supported.
- Messages can be generated through event generator, and post-processing is also supported.
- A GUI is provided to users for instant sanity checks, deeper inspection, or simply observing node movements in real time.

Details about the ONE simulator will be presented in the following sections.

## 4.2 Structure of the ONE Simulator

The structure of the ONE simulator is shown in Figure 4.1. The core of

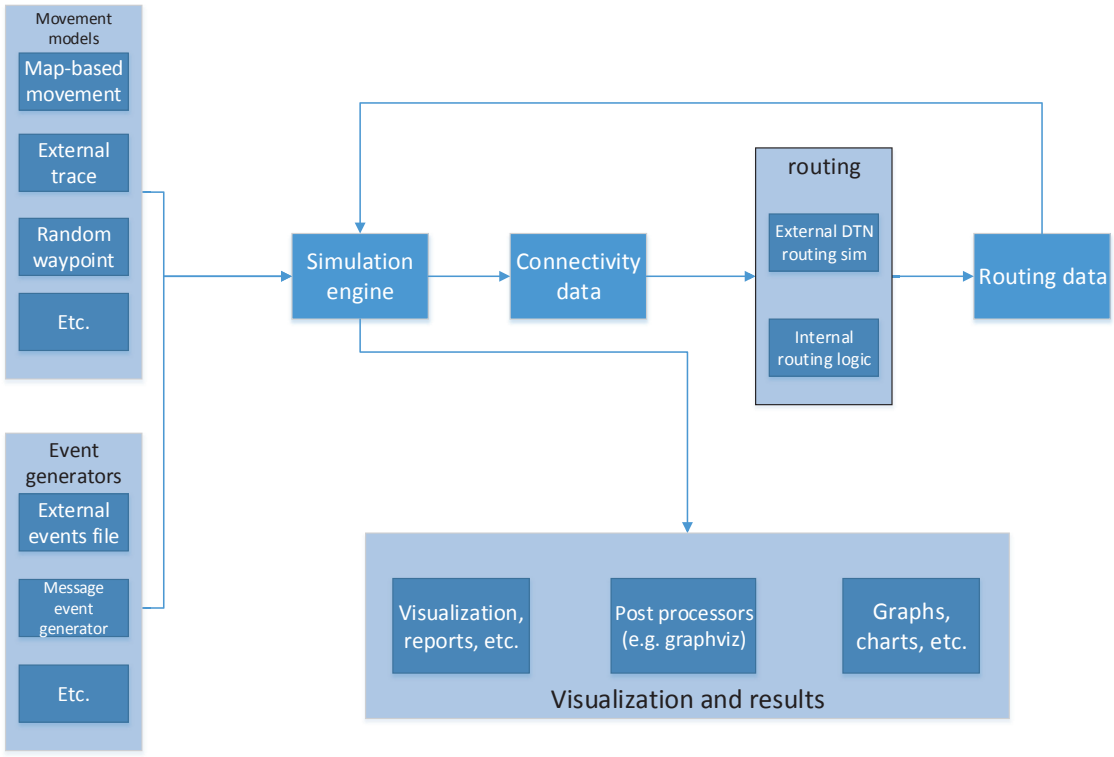



Figure 4.1: Structure of the ONE simulator

the ONE simulator is an agent-based discrete event simulation engine. A number of modules implementing the main simulation functions are updated at each simulation step. These main functions include modeling of node movement and connection, message processing, routing, etc.

Movement models govern the way nodes move in the simulation. Five basic installed models are provided in the ONE simulator: random waypoint, map based movement, shortest path map based movement, map route movement and external movement. The movement speed and pause time are drawn from a uniform distribution, where the minimum and maximum values can be configured, except for the external model where the speed and pause time are interpreted from the given data.

Routing modules define how the messages are handled in the simulation. There

are two sorts of routing modules: active and passive. Passive router is made especially for interacting with other (DTN) routing simulators or running simulations that do not require any routing functionality. Active routing modules are implemented using well-known routing algorithms, and six modules are provided in the ONE simulator: First Contact, Epidemic, Spray and Wait, Direct delivery, PRoPHET and MaxProp.

In event generating module, two classes, ExternalEventQueue and MessageEventGenerator, can be used as a source of message events. In the first class, users can create scripts by hand, or convert other output (e.g., dtnsim2) for its use. The second class creates uniformly distributed message creation patterns with configurable message creation interval, message size, and source/destination host ranges.

The ONE simulator uses report module to generate simulation results. The reports can be logs of events (e.g., node connectivity, message transmissions) that can be further processed after simulations, or the aggregate statistics calculated by the simulator. GUI provides simulation states showing node locations, connectivity, message transmission etc. Figure 4.2 presents the GUI displaying the simulation running with the Istanbul map.

A detailed description of the ONE simulator is available in [47] and [48]. The source code of the ONE simulator can be downloaded from [46].

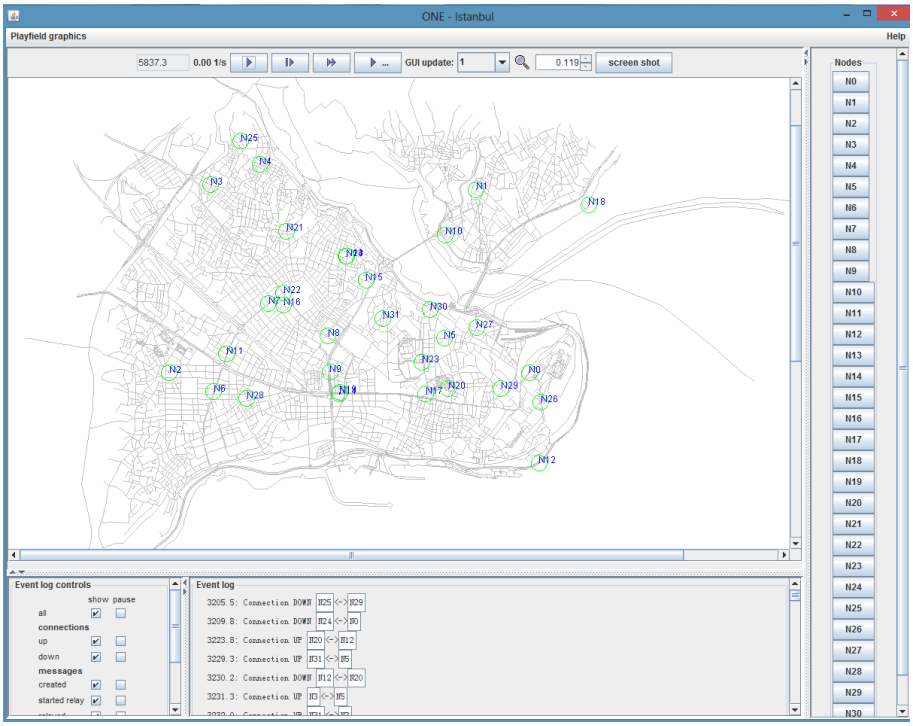



Figure 4.2: Screenshot of the ONE simulator running on Istanbul map

## 4.3 Configuration of the ONE simulator

In this section, some key configuration parameters are discussed to explain how they affect simulation performances. The detailed instructions on how to use the ONE simulator can be found in the ReadMe file included in the source code package [46]. All simulation parameters are given using configuration files. These files are normal text files that contain key-value pairs with the form

$$\text{Namespace.key} = \text{value}$$

Any number of different types of nodes can be created in the simulation. Each node group shares common configuration parameters, such as movement speed, transmission range, etc., so that it is possible to have pedestrians, cars, and buses

in one simulation.

**Transmission speed** determines how fast a message can be transferred to a neighbor during one contact. Therefore, the number of transferred and delivered encodings and thus files will be affected, and so will delivery latency.

**Transmission range** determines connectivity range of nodes. The larger the transmission range is, the more contacts nodes can have, and the longer contact times are. Thus, message delivery ratio will increase and delivery latency will decrease.

**The number of nodes** reflects population density in the simulation. By influencing node connectivity, almost all performance metrics will be affected. On one hand, a larger number of nodes can create more contacts between nodes; on the other hand, more copies of messages will also be created, which may cause network congestion.

**Movement speed** determines how fast nodes move in the simulation. Higher movement speeds change network topology more rapidly, and nodes can meet more frequently. However, the contact times between nodes also decrease accordingly. The performances are affected by both features.

**msgTTL** is the TTL of messages created by the host. It directly affects delivery ratio. If the value is too small, it is possible that messages will be dropped before final delivery. If the value is too big, buffer may fill up quickly with arriving encodings, which makes the network congested.

## 4.4 Limitations of the ONE simulator

Although the ONE simulator is a useful tool for DTN research, it still has some limitations.

First, the ONE simulator is based on discrete-event agent. Node movements and message transfers are all processed within a single time step. Although this time step can be reduced arbitrarily small, small time steps tend to slow down simulations, even possibly below one simulated second per second. Also sometimes computing resources can pose bottlenecks, especially in scenarios involving large population of nodes and complex routing algorithms.

Second, the ONE simulator lacks support for lower layers, such as physical layer and MAC layer. When two nodes are in the range of each other, they transmit messages with a constant speed which is configured by users. This is not realistic since transmission speed is affected by the distance between devices as well as interference, and also by mobility of nodes.

Finally, the radio devices in the simulation are always turned on. However, for energy saving, some users may switch their devices to idle or suspending mode, and some will only probe other devices periodically. Therefore, the contact times between nodes in simulation may be too opportunistic.

Despite these drawbacks, the ONE simulator is still one of the best simulators so far for DTN research.

# Chapter 5: Simulation Results

In this section, simulation results are presented to evaluate the performances of nullspace stopping condition mechanism. Four routing protocols are considered for the study: (1) source coding with Bloom filter without rank check (Src-BF-Epsilon); (2) source coding with Bloom filter and rank check (Src-BF-RC); (3) network coding (NC); (4) network coding with nullspace stopping condition (NC-NSC). All are equipped with immunity mechanism described in section 2.4. The ONE version 1.4.1 [46] is used throughout all simulations. For each configuration, we generate ten simulation runs with different random seeds (detailed configuration can be found in ReadMe file in the source code [46]), and we evaluate the average numbers for each performance metric.

We consider two different event scenarios. In the first scenario, the source nodes generate multiple relatively small files throughout the simulation time. In the second scenario, one source node generates one large file at the beginning of the simulation. Also, all the simulation parameters for different scenarios are listed in Chapter A.

Figure 5.1: Istanbul Urban Map

## 5.1 Scenario #1: Urban Network

In this section, simulations are run on a urban map of Istanbul, which is shown in Figure 5.1. There are three stationary nodes and 29 mobile nodes moving on the map.

### 5.1.1 Multiple File Case

In the first case of simulation, 100 files of 25 Megabytes are generated throughout 1 million simulation seconds. Each file is chopped into 500 chunks of 50,000 Bytes. Figure 5.2 presents the number of delivered files and their average delivery latency with different routing protocols. It is shown that the performance of protocols with source coding outperforms that of protocols with network coding in terms of both the number of delivered files and delivery latency. Also the performance is comparable for two protocols with source coding, and so is that of two network coding-based protocols. Specifically as shown in Table 5.1, there is no clear

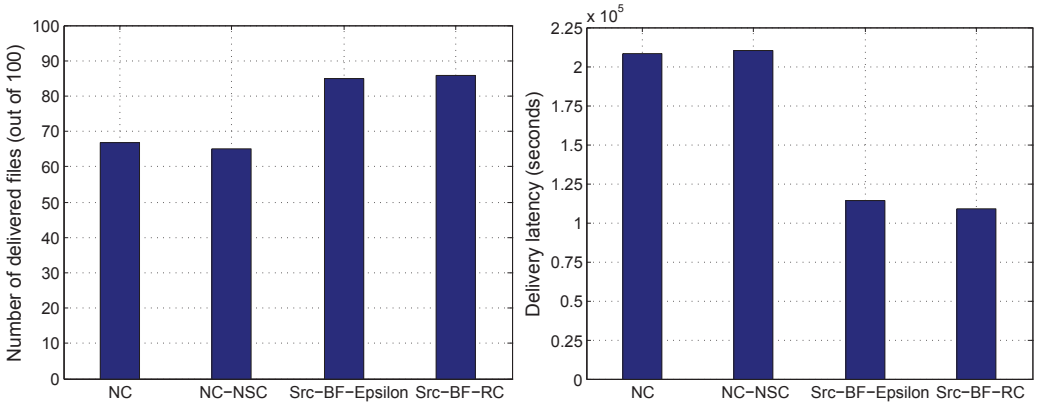



Figure 5.2: Number of Delivered Files (left) and Delivery Latency (right) in Istanbul Map with 100 Files

evidence that nullspace stopping condition helps to improve performance with network coding. Although the number of encodings created by network coding, the total number of transmissions and the number of redundant transmissions decrease slightly by 11.3%, 11.0% and 20.0%, respectively when equipped with nullspace stopping condition, the number of innovative encodings delivered to destinations reduces by 3.0% and delivery latency increases by about 10,000 s in NC-NSC.

| Performance metrics | NC | NC-NSC |
|---|---|---|
| Number of network coded encodings | 1683572 | 1492967 |
| Number of total transmission | 1705282 | 1517714 |
| Number of redundant transmission | 1123963 | 899151 |
| Number of innovative encodings to destination [1] | 17406 | 16893 |
| Delivery latency (seconds) | 396845 | 409081 |

Table 5.1: Performances of NC and NC-NSC in Istanbul with 100 Files

[1] An innovative encoding is that its encoding vector is linearly independent of all other encoding vectors received at destination.

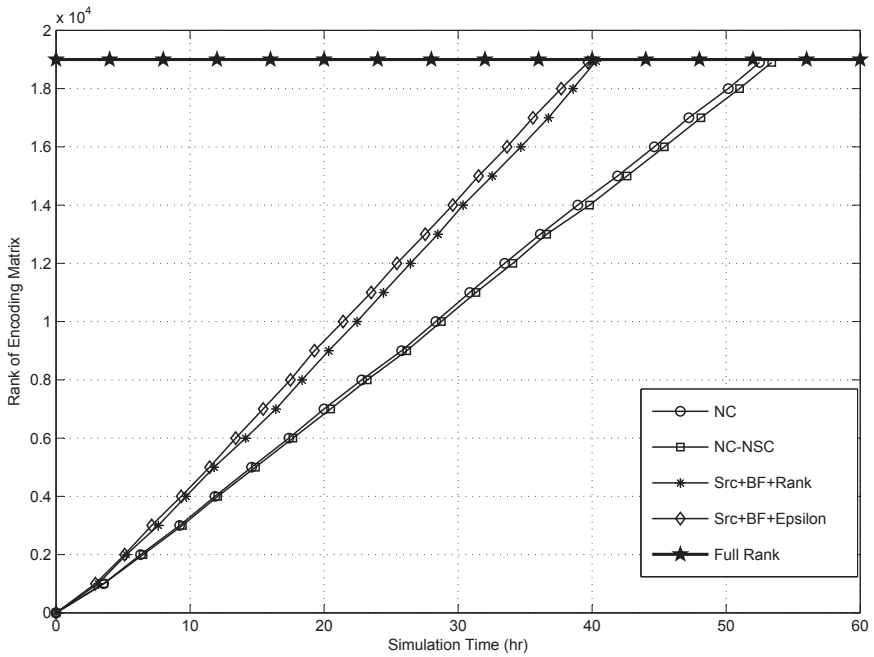



Figure 5.3: Rank of Encoding Matrix of the Single File in Istanbul

### 5.1.2 One Large File

In the second case of simulation, a single source node generates one large file of 1.9 Gigabytes in the beginning of simulation, which is chopped into 19,000 chunks of 0.1 Megabytes. With different routing protocols, this file is always delivered. The rank of encoding matrix at the destination node as a function of time is presented in Figure 5.3. It is shown that the rank increases faster using source coding compared to that with network coding, where the average delivery latency decreases by more than 10 hours. Furthermore, the performance of two protocols using source coding is comparable, and so is that of two network coding protocols.

Considering two protocols using network coding NC and NC-NSC, NC-NSC performs a little worse than NC. Table 5.2 compares several performance metrics of

| Performance metrics | NC | NC-NSC |
|---|---|---|
| Number of network coded encodings | 715301 | 720789 |
| Number of total transmission | 720125 | 725251 |
| Number of redundant transmission | 118683 | 123851 |
| Delivery latency (seconds) | 189253 | 192569 |

Table 5.2: Performances of NC and NC-NSC in Istanbul with One File

both NC and NC-NSC. It is shown that the number of network coded encodings, total transmissions, redundant transmissions and delivery latency all increase slightly with nullspace stopping condition.

### 5.1.3 Analysis

In both cases, although NC-NSC is expected to reduce redundant transmissions and delivery latency compared to NC, it is not shown that nullspace stopping condition helps to improve performances in the urban map. One possible reason is that the urban network is relatively highly connected. There exist multiple paths from the source to the destination over time. Nodes are very likely to receive innovative encodings in each contact, where nullspace stopping condition may not be effective. On the other hand, the occasional false positives, additional calculation to create NSBL, and extra overhead in control messages decrease the performance of NC-NSC.

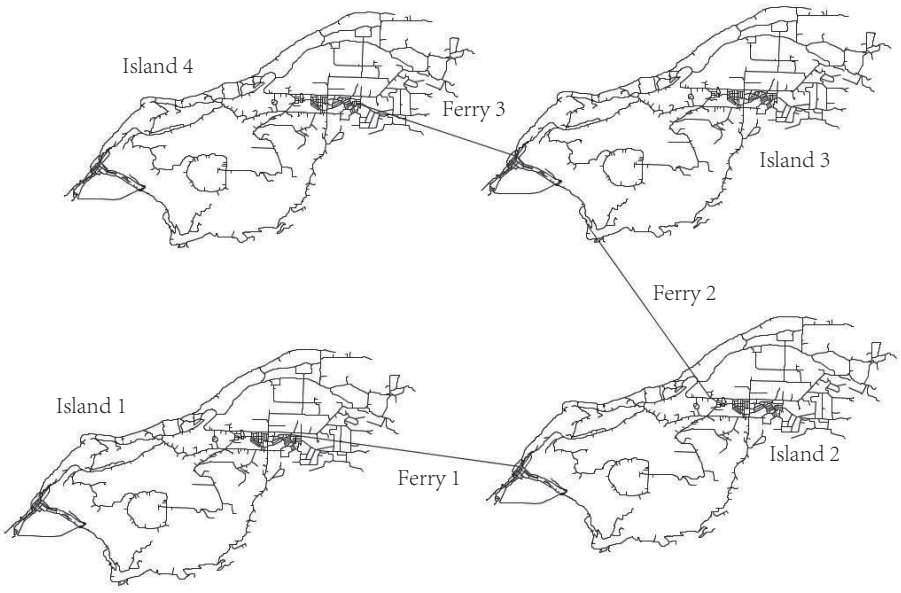



Figure 5.4: Island Hopping Scenario

## 5.2 Scenario #2: Island Hopping

In this section, the routing performances on a scenario called island hopping are presented. In this scenario, as depicted in Figure 5.4, nodes are separated into isolated islands, and the communication among islands is provided by ferry nodes moving occasionally between islands.

In the simulation, 40 nodes are placed in four islands, each with 10 nodes. These island nodes are configured with movement speed and pause time uniformly distributed from 10 m/s to 16 m/s, and 0 s to 120 s, respectively. Three ferry nodes move back and forth between islands, and stay at "harbor" at islands for 10 minutes, providing opportunities to communicate with island nodes. For notational simplicity, in Figure 5.4, the ferry nodes $i-1$ and $i$ are called upper ferry and lower ferry with respect to island $i$, respectively (island 1 does not have upper ferry and

island 4 does not have lower ferry). Similarly islands $i$ and $i+1$ are called upper and lower island with respect to ferry $i$. There is one stationary node on island 1 acting as source node, and there is another stationary node on island 4 which is a destination node, respectively. The difference between [32] and our setup is that we are interested in delivering the files to the destination node, not just to all nodes on the final island.

In island hopping, when a ferry node moves from island 1 to 2, the encodings on this ferry will be distributed to nodes on island 2. However, no matter how nodes on island 2 exchange and recombine these encodings, the space ultimately spanned by coefficient vectors in island 2 will be a subspace of that in island 1. The similar effect will also happen in the following islands. This phenomenon is analogous to "biological founder effect" that the colony breaking off from a larger population will have less genetic diversity.

### 5.2.1 Multiple Files

In the first case, source nodes create multiple small files throughout one million simulation seconds. We first investigate the situation where 40 files are generated with 30 Megabytes each, and each file is chopped into 600 chunks of 50,000 Bytes.

Figure 5.5 presents the number of delivered files and average delivery latency of different routing protocols. It is shown that Src-BF-Epsilon outperforms other protocols on both aspects, where approximately 32 out of 40 files are delivered, while the average delivery latency is less than 180,000 s. Also, although NC-NSC

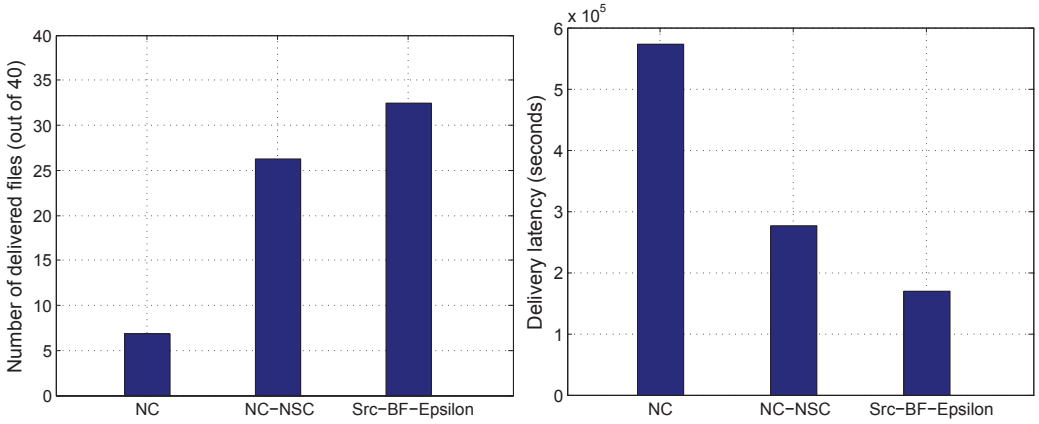



Figure 5.5: Number of Delivered Files (left) and Delivery Latency (right) in Island Hopping with 40 Files

does not stand out, there is still significant improvement over NC in terms of above delivery metrics.

Specifically comparing performances of NC and NC-NSC in Table 5.3, it is clear how much nullspace stopping condition improves performance. The numbers of network coded encodings and total transmissions both reduce by around 13.3%, and that of redundant transmissions decreases by 20.5%. In terms of delivery metrics, nullspace stopping condition helps to increase the number of innovative encodings delivered to destinations by 66.1%, meanwhile the delivery latency decreases about by half.

| Performance metrics | NC | NC-NSC |
|---|---|---|
| Number of network coded encodings | 3541837 | 3063495 |
| Number of total transmission | 3574159 | 3100163 |
| Number of redundant transmission | 2557045 | 2032346 |
| Number of innovative encodings to destination | 11836 | 19663 |
| Delivery latency (seconds) | 573617 | 276202 |

Table 5.3: Performances of NC and NC-NSC in Island Hopping with 40 Files

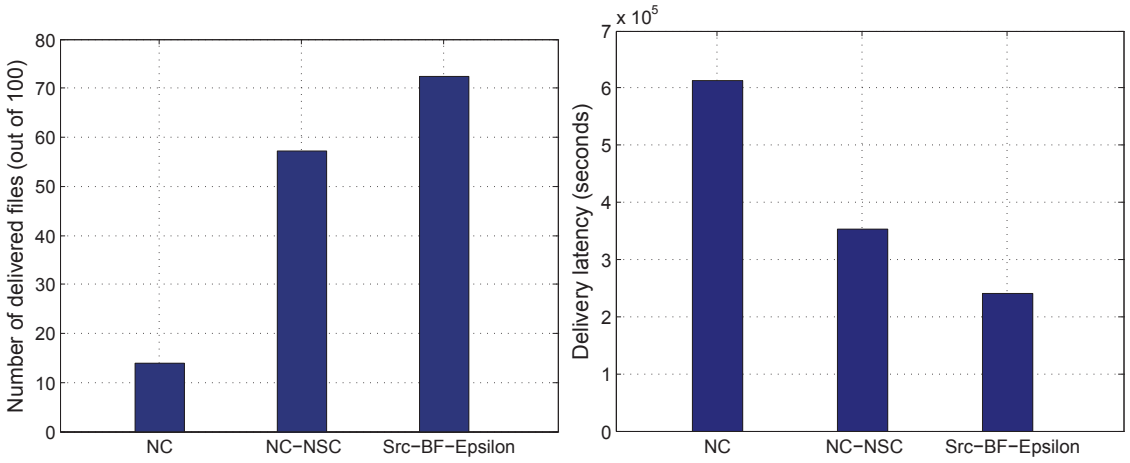



Figure 5.6: Number of Delivered Files (left) and Delivery Latency (right) in Island Hopping with 100 Files

In a more congested case, 100 files of 25 Megabytes are generated throughout the simulation time (also one million seconds), each of which is chopped into 500 chunks of 50,000 Bytes. Some performance metrics of different routing protocols and the comparison between NC and NC-NSC are presented in Figure 5.6 and Table 5.4, respectively.

| Performance metrics | NC | NC-NSC |
|---|---|---|
| Number of network coded encodings | 7177010 | 6474538 |
| Number of total transmission | 7288055 | 6588742 |
| Number of redundant transmission | 5228605 | 4369085 |
| Number of innovative encodings to destination | 23667 | 39071 |
| Delivery latency (seconds) | 613063 | 353549 |

Table 5.4: Performances of NC and NC-NSC in Island Hopping with 100 Files

Similar results as in the case of 40 files are found, where Src-BF-Epsilon outperforms the other two protocols, and nullspace stopping condition improves the performance of network coding. Nullspace stopping condition prevents unnecessary

network coding and transmissions, and facilitates the delivery of more innovative encodings (thus files).

Notice that in the above two cases, the size of each chunk is the same. We propose a parameter called relative performance gain $g$ to measure how much nullspace stopping condition improves performances. Relative performance gain is defined as

$$g = \frac{|value\ of\ \mathbf{NC} - value\ of\ \mathbf{NC\text{-}NSC}|}{total\ chunks \times transmission\ speed} \tag{5.1}$$

where the value (of NC and NC-NSC) means a certain performance metric number, and the total chunks represents the number of chunks of each file multiplied by the number of files.

Table 5.5 presents the relative performance gain in both 40-file and 100-file cases. It indicates that with an increasing number of total chunks, the relative performance gain decreases, i.e., diminishes the benefits brought on by nullspace stopping condition.

| Performance metrics × 200K | 40 Files | 100 Files |
|---|---|---|
| Number of network coded encodings | 19.93 | 7.02 |
| Number of total transmission | 19.75 | 7.00 |
| Number of redundant transmission | 21.86 | 8.60 |
| Number of innovative encodings to destination | 0.33 | 0.15 |
| Delivery latency (seconds) | 12.39 | 2.60 |

Table 5.5: Relative Performance Gain of NC and NC-NSC in Island Hopping with 100 Files

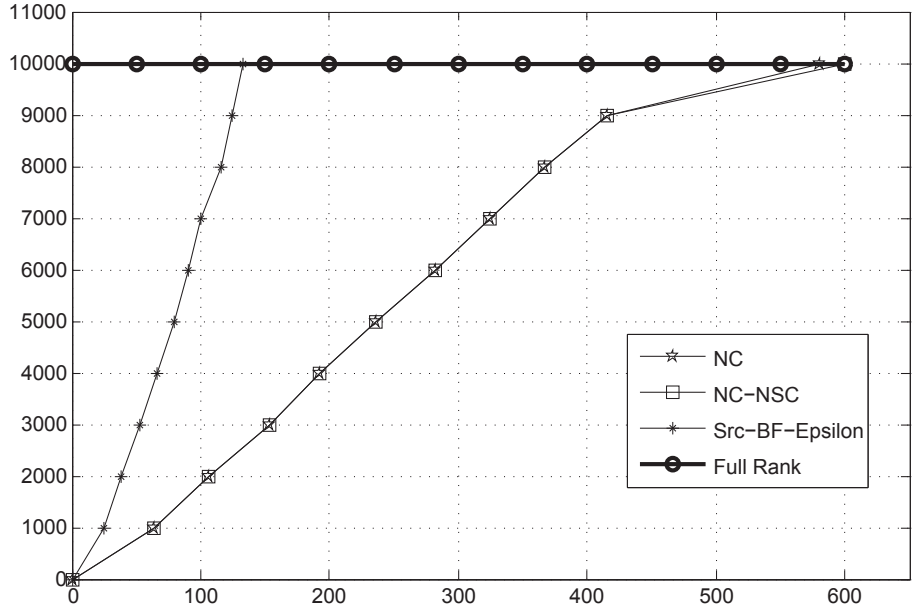



Figure 5.7: Rank of Encoding Matrix of the Single File in Island Hopping

### 5.2.2 One Large File

In the second case, source generating only one file is considered. One Gigabyte file is chopped into 10,000 chunks with 0.1 Megabytes.

The simulation results reveal that, this file is always delivered with different protocols. The rank of encoding matrix at the destination node as a function of time is presented in Figure 5.7. The delivery latencies for NC, NC-NSC and Src-BF-Epsilon are 2,090,946 s, 2,159,155 s and 476,854 s, respectively. Therefore, Src-BF-Epsilon performs best in this situation, and the performances of NC and NC-NSC are almost indistinguishable. Furthermore, it is shown in Figure 5.8 that the performances of NC and NC-NSC are comparable with respect to other evaluation metrics. In terms of the number of network coded encodings, the number of total and redundant transmissions, NC-NSC only reduces them by around 6%, yet the

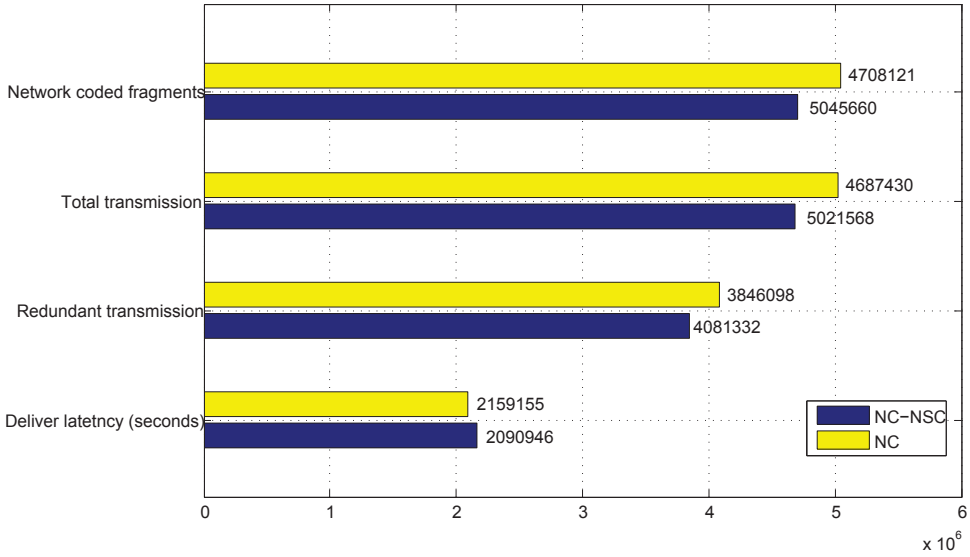



Figure 5.8: Performances of NC and NC-NSC in Island Hopping with One File

delivery latency increases about 3%. In other words, there is no major improvement when equipped with nullspace stopping condition.

### 5.2.3 Analysis

It is illustrated by simulations that nullspace stopping condition is effective in island hopping scenario with multiple small files. The number of network coded encodings and redundant transmissions in NC-NSC decreases significantly compared to that of NC. Different from urban network, island hopping has "bottlenecks" in the network, i.e., the ferry nodes. On each island, the encodings brought by the upper ferry may have spread out in the island before this ferry fetches new encodings with the same UUID. This is because file transmission follows scheduling rules (in our simulation, round robin). Therefore, if some encodings with certain UUID in a ferry node are transmitted to lower island, it needs to wait for next turn to transmit encodings of the same UUID, and also it may take relatively long time for this ferry

to receive innovative encodings of the same UUID from its upper island. Before new encodings come, although the exchange of encodings on the island can not create innovative encodings no matter how they are combined, NC still allows transmission of encodings among this island nodes even when they are not innovative to others. It is likely to happen on the island during contact that the space spanned by one node's (A) encoding matrix is a subspace of the other node's (B), or identical. Therefore, transmission of encodings from node A to node B is often unnecessary. Nullspace stopping condition exactly prevents such wasteful network coding procedures and exchange of encodings.

It is also discovered that the advantages of NC-NSC over NC decrease as the number of files rises from 40 to 100. One possible reason is that, although the probability of false positive is the same in both cases, if more files, therefore more encodings, are involved in transmission, more false positives may happen, which spoils to some extent the benefits brought by nullspace stopping condition.

In the one large file case, nullspace stopping condition does not appear helpful to improve performances. There may be two reasons for this observation. On one hand, since there is only one file needs to be transmitted, the ferry might have brought innovative encodings to its lower island before the existing encodings on that island have been fully exchanged. Therefore, almost every node is innovative to others, and nullspace stopping condition does not take effect. On the other hand, since nullspace stopping condition occasionally gives false positives, a slight degradation of performance may have occurred, as indicated of delivery latency in Figure 5.8.

# Chapter 6: Conclusion

In this thesis, nullspace stopping condition incorporating network coding in DTNs is introduced, where an improved algorithm calculating nullspace matrix has been proposed, and its performances in different scenarios with various simulation events are extensively evaluated. Simulations show that in the urban map, nullspace stopping condition neither prevents redundant transmission nor facilitates file delivery efficiently in both cases of generating multiple small files and one large file. In the island hopping scenario, nullspace stopping condition takes effect in multiple small file case significantly, whereas it is not helpful in the one large file case; it is also found that when the file number increases in multiple file case, the benefits brought by nullspace stopping condition decrease. Finally, it is shown that Src-BF-Epsilon outperforms other routing protocols in both urban map and island hopping scenarios.

# Chapter A: Parameters for Simulation

In this chapter, all the simulation parameters for different scenarios are listed in tables.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Transmission Speed | 120kBps | Transmission Range | 150m |
| Bloom Filter Size | 500000bits | Number of Hash Functions | 7 |
| Source Coding Weight | 21 | Network Coding Weight | 15 |
| Buffer Size | 2GB | Epsilon($\epsilon$) | 30 |
| Projection Vectors | 10 | Update Interval | 0.01s |
| Movement Speed | (10m/s, 16m/s) | Node Wait Time | (0s,120s) |

Table A.1: Simulation Parameters for Istanbul Map with 100 Files

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Transmission Speed | 200kBps | Transmission Range | 150m |
| Bloom Filter Size | 190000bits | Number of Hash Functions | 7 |
| Source Coding Weight | 25 | Network Coding Weight | 15 |
| Buffer Size | 2GB | Epsilon($\epsilon$) | 30 |
| Projection Vectors | 3 | Update Interval | 0.01s |
| Movement Speed | (10m/s, 16m/s) | Node Wait Time | (0s,120s) |

Table A.2: Simulation Parameters for Istanbul Map with 1 Large File

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Transmission Speed | 200kBps | Transmission Range | 200m |
| Bloom Filter Size | 240000bits | Number of Hash Functions | 7 |
| Source Coding Weight | 21 | Network Coding Weight | 15 |
| Buffer Size | 2GB | Epsilon($\epsilon$) | 30 |
| Projection Vectors | 10 | Update Interval | 0.01s |
| Island Nodes Parameters | | | |
| Movement Speed | (10m/s, 16m/s) | Node Wait Time | (0s,120s) |
| Ferry Nodes Parameters | | | |
| Movement Speed | (10m/s, 16m/s) | Node Wait Time | (508s,602s) |

Table A.3: Simulation Parameters for Island Hopping with 40 Files

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Transmission Speed | 400kBps | Transmission Range | 200m |
| Bloom Filter Size | 500000bits | Number of Hash Functions | 7 |
| Source Coding Weight | 21 | Network Coding Weight | 15 |
| Buffer Size | 2GB | Epsilon($\epsilon$) | 30 |
| Projection Vectors | 3 | Update Interval | 0.01s |
| Island Nodes Parameters | | | |
| Movement Speed | (10m/s, 16m/s) | Node Wait Time | (0s,120s) |
| Ferry Nodes Parameters | | | |
| Movement Speed | (10m/s, 16m/s) | Node Wait Time | (508s,602s) |

Table A.4: Simulation Parameters for Island Hopping with 100 Files

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Transmission Speed | 400kBps | Transmission Range | 200m |
| Bloom Filter Size | 600000bits | Number of Hash Functions | 7 |
| Source Coding Weight | 25 | Network Coding Weight | 15 |
| Buffer Size | 2GB | Epsilon($\epsilon$) | 30 |
| Projection Vectors | 15 | Update Interval | 0.01s |
| Island Nodes Parameters | | | |
| Movement Speed | (10m/s, 16m/s) | Node Wait Time | (0s,120s) |
| Ferry Nodes Parameters | | | |
| Movement Speed | (10m/s, 16m/s) | Node Wait Time | (508s,602s) |

Table A.5: Simulation Parameters for Island Hopping with 1 Large File

# Bibliography


[1] K. Fall, "A delay-tolerant network architecture for challenged internets", ACM SIGCOMM, August 2003, Karlsruhe, Germany

[2] V. Cerf et al., "delay-tolerant network architecture," IETF RFC 4838, informational, April 2007.

[3] K. Scott and S. Burleigh. Bundle Protocol Specification. RFC 5050 (Experimental), Nov. 2007

[4] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing", In Proc. of IEEE workshop on mobile computing systems and applications, New Orleans, LA, USA, Feb. 1999

[5] IEEE Computer Society. Internet protocol, rfc 791, Sept. 1981

[6] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant networks", ACM SIGCOMM, Portland, Oregon, USA, Aug. 2004

[7] D. Henriksson, T. F. Abdelzaher, and R. K. Ganti, "A Caching-based approach to routing in delay-tolerant networks", IEEE WCSP, Nanjing, China, Nov. 2009

[8] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Single-copy routing in intermittently connected mobile networks", IEEE SECON, Oct. 2004

[9] S. C. Nelson, M. Bakht and R. Kravets, "Encounter-based routing in DTNs", In Proc. of IEEE Infocom, 2009

[10] A. Vahdat, and D. Becker, "Epidemic routing for partially-connected ad hoc networks", Duke University Tech. Report, 2000

[11] B. Bloom, "Space/time trade-offs in hash coding with allowable errors", Communications of ACM, 13(7):422-426, Jul. 1990

[12] B. Dawkins, "Siobhan's problem: the coupon collector revisited", The American Statistician 45 (1): 7682, JSTOR 2685247, 1991

[13] M. Luby. M. Mitzenmacher, A. Shokmllahi, and D. Spielman, "Efficient erasure correcting codes", IEEE Trans. on Info. Theory, 47(2):669-584, Feb. 2001.

[14] M. Luby. "LT codes", In Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 271-282. 2002.

[15] A. Shokrollahi, S. Lassen, and M. Luby, "Multi-stage code generator and decoder for communication system", U.S. Patent. Application 20030058958.

[16] D. J. C. MacKay, "Fountain codes", IEE Proc. of communications, volum 152, issue 6, Dec. 2005

[17] M. Mitzeumacher, "Digital fountains: a survey and look forward", IEEE information theory workshop, 2004, Oct. 2004

[18] Y. Dai, P. Yang, G. Chen and J. Wu, "CFP: integration of fountain codes and optimal probabilistic forwarding in DTNs", IEEE GLOBECOM, 2010

[19] E. Altman, F. D. Pellegrini, "Forward correction and fountain codes in delay-tolerant networks", IEEE/ACM trans. on networking, vol. 19, No. 1, Feb. 2011

[20] F. L. P. Albini, A. Munaretto, and M. Fonseca, "Delay tolerant transport protocol - DTTP", IEEE Global information infrastructure symposium (GIIS), 2011

[21] R. Ahlswede, N. Cai, S. R. Li and R. W. Yeung, "Network information flow", IEEE Trans. on info. theory, vol.46, no. 4, jul. 2000

[22] N. Cai and R. Yeung. "Secure network coding", In Proceedings of the 2002 IEEE International Symposium on Information Theory, 2002.

[23] C. Gkantsidis and P. R. Rodriguez, "Cooperative security for network coding file distribution", IEEE Infocom, 2006.

[24] X. Zhang, G. Neglia, J. Kurose and D. Towsley, "On the benefits of random linear coding for unicast applications in disruption tolerant networks", IEEE

Symposium on modeling and optimization in mobile, Ad hoc and wireless networks, 2006

[25] Y. Lin, B. Li, and B. Liang, "Efficient network coded data transmissions in disruptive tolerant networks", IEEE Proc. of INFOCOM, 2008

[26] B. Walker, C. Ardi, A. Petz, J. Ryu and C. Julien, "Experiments on the spatial distribution of network code diversity in segmented DTNs", ACM CHANTS, Las Vegas, Sept. 2011

[27] A. Petz, A. Hennessy, B. Walker, C. Fok, and C. Julien, "An architecture for context-aware adaptation of routing in delay-tolerant networks", ACM ExtremeCom, Mar. 2012, Zurich, Switzserland

[28] P. Mundur, M. Seligman, and G. lee, "Epidemic routing with immunity in delay tolerant networks", IEEE MILCOM, San Diego, California, USA, Nov. 2008

[29] T. Matsudda, and T. Takine, "(p-q)-epidemic routing for sparsely populated mobile ad hoc networks", IEEE JSAC, Issue 5, Vol. 26, 2008

[30] Z. Hass, and T. Small, "A new networking model for biological applications of ad hoc sensor networks", IEEE Trans. on Networking, vol. 14, No. 1, Feb. 2006

[31] X. Zhang, G. Neglia,, J. Kurose, and D. Towsley, "Performance modeling of epidemic routing", *Computer Networks*, Elsevier, vol. 51, no. 10, pp. 2867 - 2891, Jul. 2007

[32] A. Hennessy, A. Gladd, and B. Walker, "Nullspace-based stopping conditions for network-coded transmissions in DTNs", ACM CHANTS, Istanbul, Turkey, Aug. 2012

[33] B. Haeupler, "Analyzing network coding gossip made easy", In Proc. of the 43rd Symp. on Theory of Computing (STOC), 2011

[34] http://en.wikipedia.org/wiki/Kernel_linear_algebra

[35] The Network Simulator NS-3. http://www.nsnam.org/.

[36] A. Varga. The OMNET++ discrete event simulation system. In Proceedings of the European Simulation Multiconference, pages 319-324, Prague, Czech Republic, June 2001. SCS - European Publishing House.

[37] D. Gorgen, H. F., and Hiedele, C.JANE, "The java Ad hoc network development environment", in Proc. of the 40th annual simulation symposium (ANSS), 2007

[38] S. Jain, K. Fall, R. Patra, "Routing in a delay tolerant network", in Proc. of ACMSIGCOMM 2004

[39] DTNSim2 simulator. http://watwire.uwaterloo.ca/DTN/sim/

[40] The CRAWDAD project. http://crawdad.cs.dartmouth.edu/

[41] N. Eagle and A. S. Pentland, "Reality mining: sensing complex social systems", Personal Ubiquitous Computing, 10(4):255-268, 2006.

[42] Jean-Yves Le Boudec and Milan Vojnovic, "Perfect simulation and stationarity of a class of mobility models", In Proc. of IEEE Infocom, 2005

[43] D. R. Choffnes, F. E. Bustamante, "An integrated mobility and traffic model for vehicular wireless networks", In Proc. of the 2nd ACM International workshop on vehicular Ad-hoc networks, 2005

[44] C. Bettstetter, "Smooth is better than sharp: A random mobility model for simulation of wireless networks", In Proc. of ACM MSWiM, Jul. 2001

[45] S. Kurkowski, T. Camp, N. Mushell, and M. Colagrosso. "A visualization and analysis tool for ns-2 wireless simulations: inspect", In MASCOTS '05: Proc. of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 503-506, Washington, DC, USA, 2005. IEEE Computer Society.

[46] http://www.netlab.tkk.fi/tutkimus/dtn/theone/

[47] Ari Keränen and Jörg Ott and Teemu Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation", SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Rome, Italy, 2009

[48] Ari Keränen, phd thesis, 2008