Large Scale Ad Hoc Information Centric Networks in Disrupted Environments

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Katherine (Raven) Russell
Master of Science
George Mason University, 2012
Bachelor of Science
University of Maryland University College, 2007
Associate of Arts
Anne Arundel Community College, 2004

Director: Dr. Robert Simon, Professor
Department of Computer Science

Fall Semester 2020
George Mason University
Fairfax, VA

Dedication

This dissertation is dedicated to my husband, Khristopher Vouvounas, who has been very patient for many years, and to my grandfather, Robert Bender, who I loved and greatly miss. Without their unwavering support during my life this would not have been possible.

Acknowledgments

I would also like to thank the following people who made this possible: My advisor Dr. Robert Simon who somehow always knew how to motivate me; my friend Kevin Andrea who kept me sane(r) by almost single-handedly providing me with a social life; Phil, Kerrie, Mark, Ian, and Margot Van Den Berghe who have been like family to me; my grandmother Patricia Reffit who has always had faith in me; my former advisors (Sean Luke and Gheorghe Tecuci), as well as my mentors and role models (esp. Pearl Wang, Don Seto, Mark Snyder, Jan Allbeck, Yutao Zhong, and my committee), who all gave me very good advice that (like Alice) I didn't always follow; my Japanese Language professor Miyoko Dickerson who gave me my first taste of teaching; and my other family, friends, colleagues, and coworkers over the years who made the time fly by (esp. Leon Cheng, Kevin Franklin, David Fleming).

I also want to thank the Motivation Moose who watched over this entire process.

Table of Contents

List of Tables

List of Figures

Abstract

LARGE SCALE AD HOC INFORMATION CENTRIC NETWORKS IN DISRUPTED ENVIRONMENTS

Katherine (Raven) Russell, PhD

George Mason University, 2020

Dissertation Director: Dr. Robert Simon

Information Centric Disruption Tolerant Networks (ICDTNs) have been forwarded as a new powerful approach to provide effective data and information sharing when existing communication infrastructures are degraded or destroyed. ICDTNSs can assist with communication infrastructure collapse and disaster response and aid is spreading vital information in a fast and reliable manner. This dissertation investigates the construction of very large (city-scale) ICDTNs without infrastructure support for such scenarios.

I first present a new, scalable architecture for ICDTN construction using a hybrid approach with Named Data Networking (NDN) and Delay/Disruption Tolerant Networking (DTN) which avoids the need to heavily modify existing NDN and DTN implementations. I analyze different design choices for routing and location awareness and show that a geographically aware architecture is particularly effective for handling mobility at scale in an ICDTN. This conclusion is based upon evaluating "geographical" and "node" aware NDN implementations under a number of mobility models, network sizes, DTN forwarding methods, and caching in disaster scenarios. The results demonstrate the scalability and effectiveness of this architecture with the appropriate selection of geographical information, name advertisement, and caching.

The effectiveness of an ICDTN in an infrastructureless environment is largely dependent upon efficient and scalable data advertising. Data advertising is the process by which hosts are informed about available data in the system and how to request this data. I address this issue by exploring the use of random linear network coding. My approach describes how to encode, decode, and use NDN advertising structures in an ICDTN environment, and I compare my solution to standard flooding techniques under a variety of network sizes, communication conditions, node densities, and mobility patterns. The results show that random linear network coding is highly scalable and can significantly decrease the time for advertisement message delivery.

I then develop a general-purpose probabilistic model for the data advertisement problem. Using this model, I analyze three protocols: a naive flooding mechanism, a network coding solution, and an improved flooding protocol inspired by the network coding approach. By measuring several key performance metrics and comparing the analysis results with empirical data for a variety of network topologies in disrupted environments, I show that network coding, and network coding inspired techniques, can be used to greatly reduce the minimum receive times and shorten the announcement period. My probabilistic model provides framework for analyzing future advertising protocols.

Finally, using Agent-Based Modeling (ABM) I evaluate the performance of my hybrid architecture and data advertisement protocols in a metro-level disaster response scenario. With over a million devices modeled, these results expose challenges not currently addressed in the literature that are unique to large-scale ICDTNs, including the interplay of NDN parameters, the effects of local caching, and the challenges with mass human movement. Experimentally, the hybrid architecture with the geographical-aware NDN setup can easily scale to deliver up-to-date information in an ad hoc manner in such a disaster scenario.

My work paves a way forward for future real-world implementations of city-scale infrastructureless ICDTNs by providing a ground-up approach to the problem. It addresses the initialization of the network during the data advertisement process, the architecture of the network needed to handle information distribution in disrupted environment, and an ABM approach to studying these networks at scale.

Chapter 1: Introduction

This work presents a ground-up solution to handle large scale telecommunication infrastructure collapse using Information-Centric Delay and Disruption Tolerant Networks (ICDTNs). Over the last decade, events have shown that it is not always possible to depend on existing infrastructure for communication. Large scale disasters, military conflicts, power outages, and even large urban gatherings, can bring down existing infrastructures [1–4]. Even a partial infrastructure collapse can cause major interference with emergency services in disaster scenarios since the majority of calls to 9-1-1 currently come through the cell phone infrastructure [5].

With the prevalence of small, network ready devices (phones, cars, IoT devices, and the like), large city-scale networks could be formed with these devices to continue communication in an ad hoc manner. Such ad hoc, small-device networks, sometimes referred to as Pocket-Switch Networks (PSNs), have a few basic properties in common: (1) they are often disconnected or semi-connected, (2) there is no, and should be no, centralized control, and (3) the devices are heterogeneous in nature. As a result, these networks have more in common with Opportunistic Networks (OPNets or OppNets) than with any other form of traditional mobile ad-hoc networks (MANETs).

Delay and Disruption Tolerant Networks (DTNs) and Information-Centric Networks (ICNs) have both been suggested as suitable for such situations. Delay-Tolerant Networking was originally designed to handle space communication but has been shown to have applications to many other disrupted environments [6]. Information Centric Networks (ICNs) are a newer class of networks that is expected to complement or ultimately replace many current IP-oriented systems [7]. By treating data content as the basic object for naming,

addressing, and routing, ICNs attempt to refocus the network management on connectionless data retrieval rather than host-to-host communication. Most ICNs naturally support mobility for data consumers and offer in-network caching of content.

Both these technologies have proposed applications for disaster relief, robotics, vehicular networks (VANETs), flying ad hoc networks (FANETs), pocket-switch networks (PSNs), Internet of Things (IoT), military drones, underwater networks, and many other fields where an existing communication infrastructure cannot be guaranteed and the nodes move in and out of communication with each other [8–15]. The benefits of combining these two field has also not been overlooked, leading to a new area research on Information-Centric Delay-Tolerant Networks (ICDTNs) [16, 17]. This new area is creating tremendous interest from both industry [18] and the research community [19]. However, this field is still in still in its infancy with many unsolved challenges. To quote one recent paper:

> Several research works attempt to directly extend ICN approaches to DTN. However, such migration is neither effortless nor seamless. For one thing, studies of ICN are still in its nascent stage and lots of problems remain unsolved. For another, it should be noticed that ICN and DTN have essential architectural differences. [20]

The work presented in this dissertation examines *exactly* these challenges, offering a holistic solution to combining ICNs and DTN in a natural, and above all simple, manner, while also addressing one of the fundamental challenges left open by the ad hoc ICN community: "data advertisement" in an infrastructureless environment (discussed shortly).

The first major contribution of this work addresses the problem of fundamentally different architectures. Traditionally, DTNs use a technique called "store-and-forward" or "store-carry-forward" which allows them to handle temporary network separations by simply waiting for an opportunity to send the message (store) and sending the message on when a path becomes available (forward) [21]. In networks where the nodes are mobile,

messages may also be physically carried around the world (carry). This technique is designed to address connectivity failure and network partitions. On the other hand, ICNs typically work using variations of distributed hash tables, reverse path routing, or rely on devices "anchored" to some existing infrastructure. These techniques are designed to handle mobility, but not substantial segmentation or disruption of the entire network.

There have been many proposals for combining these two technologies spanning the range of replacing core ICN components with DTN components [22, 23], to augmenting DTNs with ICN naming [24], to simply interfacing the two types of networks [16, 17]. This work takes a different approach, and uses layers to assign the responsibilities to the appropriate protocol. Information retrieval guidance and in-network caching is the province of the ICN layer, while a separate DTN layer takes charge of issues related to network disruption and fragmentation. This produces a "best of both worlds" architecture which gives up none of the benefits of the individual frameworks while also being simple to implement using the very small interfacing middle layer.

This middle layer approach also provides different ad hoc ICDTNs with tailored solutions to many common ad hoc scenarios. I show in Chapter 4 two different versions of this middle layer, one of which supports human mobility patterns (the "geo-region face layer") and one of which supports non-mobile disruption and data-mule scenarios (the "node face layer"). I also show in 4 that this simple architecture can also help scale the networks to thousands of devices.

While Chapter 4 shows the scalability of this solution, it does not specifically address the open challenge of data advertisement in an ad hoc ICDTN. Data advertisement is the process by which nodes become aware of data in the network that can be requested, and is the complement of network device discovery in host-centric networks. This is one of the foundational components left open by the ICN community. To add an additional challenge, there has been relatively little focus on analyzing and quantifying the performance of ICDTN protocols and implementations for a number of foundational components (including data advertisement). This work attempts to specifically address this gap by first defining the

problem of data advertisement and its relationship to more well-known networking problems in Chapter 2, proposing a concrete protocol using network coding in Chapter 5, and offering an analysis framework using the PUSH model for additional protocols in Chapter 6.

In addition to the contributions outlined above, I also examine *city-scale* ad hoc ICDTNs, which is a challenge not currently explored in the literature. Most ICDTNs proposed are simulated for a few hundred devices, while Chapter 4 of this work contains experiments with over a thousand devices, and Chapter 7 examines networks with over a million devices. By integrating with Agent-Based Models (ABMs), Chapter 7 also address the challenges of mass human movement in a disaster scenario, which in itself is adds another layer of complexity. Nevertheless, this work offers a simple, scalable solution to such large-scale, massively mobile, infrastructureless ICDTNs.

Providing communication support during and after natural and human-made disasters can provide a devastated population with support from aid workers and local community organizations when existing systems fail. Given the major contributions of this work, I hope that the exciting field of ICDTN research can begin to consider these city-scale implementations as a viable solution to infrastructure collapse. The next section offers more information on the structure of this work and succinctly states the contributions offered.

1.1 Abbreviations and Acronyms

For the reader's convenience, Table 1.1 gives a list of abbreviations and acronyms which are mentioned more than once in this dissertation.

Table 1.1: General Abbreviations and Acronyms

Term	Meaning
ABM	Agent-Based Model
CCN	Content Centric Networking
CS	Content Store
DTN	Delay and Disruption Tolerant Network(ing)
FIB	Forwarding Information Base
GIS	Geographical Information Systems
GMU	George Mason University
ICN	Information Centric Network(ing)
ICDTN	Information Centric Delay and Disruption Tolerant Network(ing)
IoT	Internet of Things
ISR	Information Satisfaction Ratio
IST	Information Satisfaction Time
LR	Long-Range (Network)
MGRS	Military Grid Reference System
NDN	Named Data Networking
PIT	Pending Interest Table
PSN	Pocket-Switched Network(ing)
SaW	Spray-and-Wait
SP	Static Connection Probability Network
SR	Short-Range (Network)
SWIM	Small World in Motion
TLV	Type-Length-Value (Format)
TSP	Two-State Connection Probability Network
VANET	Vehicular Ad Hoc Network(ing)
VGI	Volunteered Geographic Information
WoM	Word-of-Mouth

1.2 Road Map and Summary of Contributions

The major thrust of this dissertation is an architecture and accompanying protocols for implementing ad hoc ICDTNs for infrastructure collapse scenarios. Chapter 2 provides

necessary background, while Chapter 3 introduces basic architectural elements, communication models, motion models, and measurements which are *derivative* of work presented in Chapter 2, but are also used throughout this work.

- *Major Contribution of Chapter 3.* A set of models and measurements for quantifying performance of ad hoc ICDTNs, and a thorough exploration of the parameter space available for several key components.

Chapter 4 presents a basic layered architecture which provides the foundation for scaling to city-scale ad hoc ICDTNs. This architecture is unique in its flexibility compared to previous approaches discussed in Chapter 2 and demonstrates scaling for 1000+ nodes in a single network. Simplistic protocols are given in this chapter data advertisement as it is a preliminary step to data exchange.

- *Major Contribution of Chapter 4.* A new, simpler, and scalable architecture for ad hoc ICDTNs.

The next two chapter consider the problem of *efficient* data advertisement in ICDTNs. Chapter 5 discusses the practical problems of designing an advertisement protocol and presents network-coding based solution that operates in a fully disrupted environment. The proposed network coding based solution outperforms simple flooding mechanisms and is inspired by a diffusion mechanism presented in Chapter 4.

- *Major Contribution of Chapter 5.* A mechanism for efficient named-data advertisement using network coding. I also provide an analysis of the effect of partial decoding in multi-source network coding and appropriate accompanying heuristic.

Chapter 6 examines the data advertisement problem from an analytical standpoint, discussing how such protocols can be compared and contrasted, and then analyzes three solutions: the flooding approach used in Chapter 4, the network coding protocol presented in Chapter 5, and an additional network coding "inspired" solution introduced in Chapter 6.

- *Major Contribution of Chapter 6.* A framework for analyzing ICDTN protocols for data advertisement using system level abstractions for multi-source broadcasting in mobile ad hoc environments.

Chapter 7 ties the previous chapters together, presenting an ad hoc ICDTN capable of scaling to the size of city with 1.3 million people to disseminate information in a disaster scenario. It addresses issues related to this scale, such as disruption caused by mass movement and large-scale simulation concerns.

- *Major Contribution of Chapter 7.* An ICDTN implementation appropriate for city-scale networks and an approach to studying these networks using Agent Based (AB) human models for disaster scenarios.

Chapter 2: Background and Related Work

This chapter provides the background needed to understand the work in this dissertation. I present information on general ad hoc networking in disrupted environments, information centric networking, and the combination of the two (Sections 2.1-2.3). Additionally, I give an overview of the data advertisement problem in these networks (Section 2.4) and sufficient information on network coding (Sections 2.5) to understand the proposed solution to this problem presented in Chapters 5 and 6. Finally, I give a summary of the software utilized throughout (Section 2.6). Portions of this chapter have been published or presented as part of [25–27] and work currently under review.

2.1 Networking in Ad Hoc Disrupted Environments

As has been discussed in Chapter 1, it has been proposed that during infrastructure collapse, ad hoc networks could work to support communication. These networks are inherently disrupted, and communication might be temporarily or permanently prevented between any two devices in the network, while new connections may be established at any time. Slightly more formally, a disruption is a "break" in the source-to-destination path such that devices need to wait to send all, or part, of their message. Device motion, signal loss, and interference are all natural causes of disruption in these environments. Additionally, depending on the cause of the infrastructure collapse, there could also be intentional disruptions to the network by adversaries (e.g. in some military applications).

2.1.1 Modeling Disruption

There are several ways to model the causes of disruption, depending on the imagined scenario. These models include mobility models, network interference models, and probabilistic

connection models. Choosing an appropriate model is extremely important since the choice of model can heavily influence the performance of network protocols [28]. I will be using several existing models in this work to create disrupted networks: the random waypoint motion model [29], the Small World in Motion (SWIM) human motion model [30], and the Home-MEG probabilistic connection model [31].

The random waypoint motion model assumes that devices choose a location in the world, then travel directly to that location at some chosen speed. They then pause for some period of time before choosing a new location and new speed. This classic model is one of the foundational motion models developed for mobile computing but it has two well-known drawbacks: first, the distribution of nodes is not even throughout the environment and second, it does not match natural human motion patterns very well [32, 33].

Given that most pocket-switch network devices, such as cell phones, are held by humans that do not just walk randomly around the world, *many* human inspired motion models have been developed [28]. The SWIM model is human motion model which is easy to simulate at scale, and uses geographic information to inform human decisions. In short, this model works as follows: humans carrying devices choose a destination (such as a building), preferring to go to places near their home or to places that are popular. They travel to those locations using roads imported from a map at walking speed (or driving speed). Popularity is *observed* by the humans when they arrive, so they do not need global information about the world.

Probabilistic connection models are an abstraction of disruption where devices have a probability of forming a connection and also a probability of becoming disconnected at a given time. These models are often Markovian in nature, but are not required to be so. Two-state connection probability networks are a variation which can be used to mimic the unusual distribution of intercontact times in human mobility networks: exponential distributions [34] or power law + exponential tail distributions [31].

The Home-MEG model [31] is a two-state connection probability model similar to the Gilbert-Elliot model used to characterize burst-noise communication channels [35]. The

Home-MEG model allows every *pair* of nodes to be in one of two states (home or not home), and they can transition between states with probabilities p (to home) and q (to not home), and connect in each of those states with probabilities α (when at home) and γ (when away from home). This model can be fitted to several real-world traces, and it also provides a simple way to determine network properties, such as the probability that any two nodes are connected, for a number of different network types. Additionally, setting q to 0 in the Home-MEG model can be used to create a static connection probability networks (where there is simply a probability of being connection/disconnected at each time step).

To give a concrete example of a two-state probability model, imagine two devices: a employee's laptop taken back and forth from home to work, and his co-worker's cell phone. Imagining further that the two employees live in the same community, the two devices have a high probability of connecting at work, and a non-zero probability of connecting at home. With this example, p is the transition probabilities to work (the "higher connection" state), and q is the probability of transitioning to home (the "lower connection" state). Then α or γ is the probability of connecting when at work or home (respectively) in a given time slot. The states again, are per *pair* of nodes, so connections are not transitive: if the laptop and cell phone are in a high connection probability state, and a third device (maybe a desktop in the office) is in a high connection probability state with the laptop, that does *not* mean the desktop is also in a high connection state with the cell phone.

I use the Home-MEG model in Chapter 6 to study performance of a large number of network types and to validate the equations presenting in that chapter. Since this model can also be fit to data traces, it is also used to examine the theoretical performance for a stable network with a given connection probabilities and compared to the original data trace.

2.1.2 Communication in Disrupted Environments

Delay and Disruption Tolerant Networks (DTNs) were originally designed to handle disruptions in space communications [6], has been shown to have a large number of applications

outside of its original domain [36]. The DTN protocol uses 'store-and-forward" or "store-carry-forward" which handles network disruption by waiting (store), making use of opportunistic contacts (forward), and in mobile networks making use of data to be physically moved around the network (carry). The forwarding component is handled by a number of protocols which employ flooding, controlled flooding, or single-copy, depending on the constraints of the network [37].

Flooding (typified by "epidemic routing" [38]), has all nodes attempt to share information/messages with every node it comes in contact with. There are several advantages to this, flooding can provide redundancy in the system, message delivery does not need to know anything about the network topology, and the shortest path will eventually be traversed between the sender and the destination(s) in a connected network. That said, flooding is considered to be most appropriate for small networks since the overhead can be extreme in large networks (with potentially every node in the network receiving a copy of a message designed for only one node) and store-carry-forward may require a huge amount of "store" to keep all active messages [39].

Controlled flooding techniques attempt to solve the issues with flooding by simply limiting the number of copies of a message, a classic example of this is 'Spray-and-Wait" which sends a limited number of copies out (spray) and then attempts delivery by physically carrying the message copies around the network (wait) [39]. While controlled flooding greatly reduces the message overhead in a DTN, it also has several limitations. One of the main issues is determining how many copies of a message should be sent if the network size and node movement patterns are unknown, since a bad choice can result in a network incapable of delivering messages: at one extreme it may not be physically possible to carry a message from one side of the network to the other, and at the other extreme the network is completely congested due to flooding.

Sending a single copy of a message without any form of flooding would seem to be the most scalable approach, but it can be difficult to achieve high delivery rates. In a large

network without any infrastructure, messages can't just perform a random walk, more information is required. Depending on the network scenario in question, single-copy techniques range from physically carrying a message to its destination ("data mules" [40]) to more complicated heuristics like geographic routing (routing towards the physical destination, regardless of the network structure) [41–43].

In this work I use epidemic routing and spray-and-wait for our DTN layers, and I explore the use of geographic forwarding for the NDN layer. Geographic forwarding has long been studied as a good option for single-copy ad hoc network routing, but it requires information about the physical location of the destination. This information could be potentially known in advance (e.g. my refrigerator is always located in my house), but if it isn't, a location-lookup service must be provided by the network, though in small networks flooding location information is also a common practice [13]. In this work, the location of information being routed to is "naturally" made available to the network as part of the data advertisement process.

2.2 Information Centric Networking

Information-centric networks (ICNs) are a fairly recent area of research where, instead of IP addressable "host-centric" networks, information labels become the organizing focal point of the network [44]. This means that forwarding is based on the information being sought rather than knowledge of the host who provides that information. These networks are sometimes referred to as Content Delivery Networks (CDNs) and are thought to better reflect how modern networks are used [7]. Most ICN solutions allow substantial in-network caching and the ability for multiple devices to respond to a given data request.

The application area of these networks is proposed to be any network whose primary focus is information production and retrieval. By design, ICNs are also supposed to support consumer mobility since the networks are connection-less. There are various strategies proposed for these networks, but in general, messages sent based on distributed hash tables

or lookup tables, while messages returned via reverse-path or source routing.

2.2.1 Named Data Networking

Named Data Networking (NDN) is a fairly established ICN implementation with significant support [45]. Without loss of generality I focus on the NDN approach in this work, since it has features especially useful in a PSNs for disaster scenarios: (1) it allows human readable names, (2) it has built-in mobility support, (3) it provides more in-network caching, and (4) it does not require specific rendezvous points with existing infrastructure. Again, though this work chooses to focus on NDN, many of the techniques presented are also applicable to other ICN flavors.

It should be noted that data *consumers* are allowed to be mobile in NDN, but data *producers* are assumed to be non-mobile. Methods such as KITE [46] and MNDN [47] have been introduced to address the problem of mobile producers without affecting how the network operates. Because of this, I will assume all data producers are in fixed positions in the environment, but that this does not limit this work proposed to only non-mobile data producers in the long run.

In NDN, data is named with a unique identifier and a Forwarding Information Base (FIB) tells hosts how to forward information requests (called "interests") towards data for the given name. As interests flow through the network, they leave a kind of breadcrumb trail in the Pending Interest Tables (PITs) of each host. When the data reply comes back, it follows these PIT entries in a form of reverse-path routing. The remaining component is a content store (CS), which caches data on its way back to the sender in case of multiple requests for the same data within a short time window. Figure 2.1 shows a simplified example of these data structures for a network currently in use.

The FIB tracks which "face" (short for "interface") an interest should be forwarded on and the "cost" of that path. Cost can be based on any number of heuristics, such as response-delay or hop count. An entry in the FIB table thus consists of a name-prefix and multiple (face, cost) tuples indicating the "next hop" NDN should forward to for the

Name	Face	Cost
/data1	1	8
/data2	10	6
/data3/fred	6	2
/data3/fred	7	10

Simplified FIB

Name	Face
/data1	5
/data3/fred	2

Simplified
PIT In-Records

Name	Face
/data1	1
/data3/fred	6

Simplified
PIT Out-Records

Figure 2.1: Example NDN data structures.

given name-prefix. While there is a Routing Information Base (RIB) component that keeps information relevant to any routing protocols for constructing and maintaining FIB tables, the protocols themselves are left unspecified by the NDN community. Many ICNs, such as NDN, do not require that routing algorithms converge, instead, as nodes begin requesting the data, one or more *possible* routes are tried and, in a form of distributed reinforcement learning, paths that return the data quickest become preferred over the paths that are slower or form loops.

There are two types of entries in the PIT table for any given interest: in-records and out-records. Interests also keep a "nonce list" which remembers the nonces of seen interests[1]. In-records are a tuple of: the incoming face id (who requested the data), the last nonce seen, and the last renewal time. These are the "breadcrumbs" data follows on it's way back to the consumer. Out-records, on the other hand, are a tuple of: the outgoing face id (where the interest was forwarded to), the last nonce seen, and the last renewal time.

In terms of network support, my research relies upon an interest containing: a name, a boolean indicating whether the name is a prefix or an exact name, a boolean indicating whether the data must be fresh to satisfy the interest, a 4-octet byte-string nonce for loop detection, an interest lifetime indicating how long the consumer will continue to be interested in the data and/or a hop limit indicating how far the interest might be sent. Data, on the other hand, has the following items of interest: a name, a period of "freshness", the

[1] A nonce is a word or expression used only once, and in this case a number used only once.

data content, and a digital signature. There are additional fields which are not required by my techniques, such as forwarding hints for source routing, application parameters, and so forth.

2.3 Information Centric Delay and Disruption Tolerant Networking

Recently, many researchers have seen the natural fit of ICNs with DTNs (referred to as ICDTNs), since many of the application areas for ICNs are inherently disrupted [16,17,48]. Both network types offer advantages in these environments: ICNs offer a connection-less setup and have built-in mobility support and caching, while DTNs offer redundancy and are specifically designed to deal environments where disconnections are the norm rather than the exception.

While the fit seems natural, there are nevertheless substantial difficulties in combining ICNs and DTNs due to the fact that messages are shared in very different ways. The distributed hash tables and reverse path routing in ICNs do not perform well in disconnected environments, while DTN techniques like flooding, controlled flooding, and geo-routing are difficult to scale and not "organized" for information retrieval.

2.3.1 Architectures for ICDTNs

There have been many proposals to integrate ICNs and DTNs [16,17,20,22–24,49], but at a high level there are a limited number of combinations possible. I will refer to these ICDTN types here as "low" and "high" integration levels and propose a middle ground ("medium" integration) in Chapter 4.

A low level integration approach makes some nodes in the network into "ICN+DTN" nodes. These nodes act as gateways to a secondary DTN network and handle any disruption tolerant communications that need to happen. Essentially, the two networks work completely independent and use the techniques appropriate for each part of the network

without concern (or consideration) for the other half. An example of this technique using Content-Centric Networking (CCN) has been proposed in [49], where a "bundle layer" interfaces a CCN node to a DTN network.

Two ICN technologies have been singled out recently as especially compatible with DTNs [48]: PSIRP/PURSUIT [50] and CCN/NDN [45]. PURSUIT has become the basis of the RIFE project [16], an architecture which combines ICNs, DTNs, and IP-based networks using NAPs (Network Attachment Points) to act as border gateways to ICN/DTN networks from IP networks. NDN, which is an offshoot of CCN, has formed the basis of the UMOBILE project [17], an architecture which combines ICNs, DTNs, and IP-based networking by extending the NDN Forwarding Daemon (NFD) with the ability to send messages through a DTN or OppNet interfaces. Both of these projects use the "low level" of integration discussed above.

At the other end of the spectrum, the two technologies are essentially merged ("high level" of integration). This requires many interesting trade-offs between the two protocols, such as handling reverse-path routing through the DTN or handling expired PIT entries due to direct delivery. Much of the VANET community has embraced this approach, replacing major components of their ICN with flooding and modified forwarding strategies to make the ICN to work as a DTN [22, 23]. On DTN side, an expansion has been proposed to the DTN protocol to allow it to query information in a more information-centric manner [20, 24].

One of the core contributions of this work is a middle ground between these two integration levels. This is designed to be a "best of both worlds" solution which does not require significant modifications to the DTN or ICN (as the high integration level requires), while also allowing the two components to work together to complement and reinforce each other (which the low level integration doesn't allow).

2.4 Data Advertisement in ICNs

Information-centric networking relies on protocols that route towards information. Using NDN as an example, each piece of data must be uniquely named so that it can be identified in the system. A data advertisement is an "announcement" of the existence of a piece of data in the network by a "data producer". It is extremely important to understand that the data advertisements are *not* the data, but a description of a single piece of data *available* in the system, including attributes such as expiration times, creation times, and security information. The data itself may be quite large, a video for example, but the advertisements are comparatively small. The advertisement answers concerns like "how long is this information valid?", "who created this information?", "how can I trust the authenticity?", and so forth.

In NDN, data advertisement is part of the routing protocol that sets up the FIB tables and no forwarding could occur without this process. The original intention for ICNs was as that they act as a replacement for IP, so the majority of the research on data advertisement and routing protocols adopts well established techniques for networks with established infrastructure, such as NDNS (DNS for named data) for networks under the control of multiple entities [51] and NLSR (link state routing for named data) for networks under the control of a single entity [52].

2.4.1 Data Advertisement in ICDTNs

Unfortunately, there has been relatively little research done in the area of efficient and scalable data advertising in ad hoc ICDTNs. Often the networks resort to standard DTN techniques like flooding instead of traditional routing which bypasses many of the NDN benefits. In very small networks with a tens or a few hundred devices, interests and data can be sent via flooding so there is no need for routing protocols at all. Such networks can also exchange their forwarding tables (a variation of distance vector routing) fairly quickly [13] or utilize NDN's "self learning" protocol [53] when needed. Unfortunately, these methods become untenable when scaled up to a city's-worth of data.

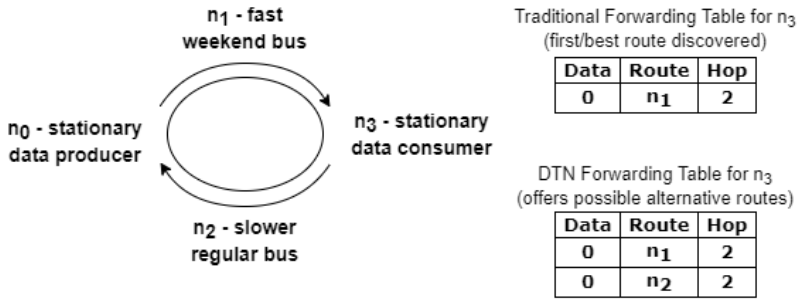


Figure 2.2: Example of alternative route formation in DTN environments.

It is important at this point to make it clear that nodes store *multiple possible paths* towards the data, not just the first path found. This is because the idea of an "optimal" path is time dependent. As a result, data advertisement in these environments is a type of routing problem where *multiple* paths towards the data need to be formed quickly and efficiently. Since there is no infrastructure in an ad hoc environment, nodes exchange information with their neighbors about what data they are aware of in the system and what paths they have to that data at the same time. This forms a type of hybrid problem of flooding the advertisements *while* determining possible routes to the data.

A simple example of the benefits of this is shown in Figure 2.2 where two nodes are stationary in different parts of a city, and two bus nodes travel between them. In this scenario, the buses are not equivalent: n_1 runs at double the speed of n_2 but only on the weekends. When n_0 sends a data advertisement (a_1) into the system, a_1 will be picked up by the two buses as they pass by. They both form entries in their routing table to n_0 for the data. Let's say n_1 passes the consumer first, the advertisement is passed along to n_3. The consumer creates a forwarding entry for a_1 through n_1. Later, when the n_2 bus passes, n_3 also creates an entry for another possible route. On Monday, if the consumer wants to request the data advertised, n_3 would need to wait five days to see n_1 again and only a short time to see n_2, even though n_1 delivered the announcement first, because the optimal path on a Monday is different than when the announcement was distributed over the weekend. If both buses were parked outside the location of n_3, then n_3 might choose to

send the request to *both*.

This general problem is the motivation for using NDN's "strategies", such as send-to-all-faces, to give NDN the redundancy of the DTN flooding mechanisms [54]. In a similar vein, a recent work on routing in ICDTNs has proposed a type of "patching" mechanism for disruptions along the shortest path [55]. This protocol has a series of fallback options for unavailable paths (multi-path routing, flooding, random walk, and store-and-wait). These techniques are a good option for a small amount of disruption, but in the case of a network in constant flux (such as an entire city of moving people), they do not offer a complete solution.

Therefore, the ultimate goal is that every node in the network is going to try to send each neighbor it meets, or some subset it deems reliable enough, an advertisement about every data item they can provide to setup the FIB tables. This needs to happen quickly, and reliably, in order for the network to function. Nevertheless, many ICDTN proposals have suggested that this process just happens "somehow". For example, in NDN's self-learning protocol, the consumer somehow knows about a piece of data that they want, but they don't know the path, so they flood to find a path. Replies to this flood creates a new path from the producer to the requester.

Table 2.1 outlines of some of the strategies proposed for data advertisement in the works discussed so far and their application to the ICDTN data advertisement problem in a city-size ICDTN. While none of these strategies are appropriate for this problem, there are fortunately a number of partial solutions from other areas of networking which can be combined into a larger solution. On the one hand, the problem is similar to a broadcasting problem (to spread the data advertisements) and on the other a traditional routing problem (for the path exchange), except in this case "a route" is just as useful as knowing the best route, since the ICN can optimize paths while forwarding.

Table 2.1: Data Advertisement Strategies

Strategies	**Example Paper(s)**	**Issues for ICDTN**
NDNS (DNS for Name Services)	[51]	Implies an infrastructure
NLSR (Named-data Link State Routing)	[52, 56]	Not appropriate for DTNs
Flooding	[22, 53]	Inefficiencies and overhead
Assumed Known or Not Specified	[49, 57]	Not practical

2.4.2 The Multi-Source Broadcasting Problem and the PUSH Model

Algorithmically the data advertisement problem can be described as a variant of the multi-source broadcasting problem, whereby one or more nodes in a system need to deliver their own message(s) to all other nodes. Distributed systems researchers have long studied versions of the single and multi-source broadcasting problem [58], where variants include gossiping and rumor spreading, where nodes produce messages and wish to ultimately deliver their messages to all of the other nodes.

The multi-source broadcasting problem has also been modeled as a variation of the "coupon collector problem". Put simply, this problem can be stated as: if a collector receives a random coupon in each round, how many rounds are needed to collect a set of n coupons?) [59, 60]. In the PUSH version of this problem time is divided into discrete units or rounds, and at each round a node can send the message to one neighbor, but only if the sender possess that message [61, 62]. The PUSH approach has been used by many researchers to study message distribution effectiveness in wireless, mobile, and ad hoc systems [60, 63, 64]. I use the PUSH model for my analysis in Chapter 6 since this model provides a tractable analytical framework which is very helpful for developing a generalizable performance model.

The coupon collector problem with the PUSH model implicitly assumes that all nodes have an equal chance of getting an advertisement (coupon) per round, which further implies that either all nodes can send to all other nodes in the network or that the distribution of information in the network is fairly uniform. These are both are common assumptions, but

not common in large-scale networks. This gives an estimate of something like the "average best case" (informally speaking) for the multi-source broadcasting problem, "average" being the number of rounds expected, and "best" in that it is an idealized version of the network.

In the data advertisement variation of this problem we have, nodes need a copy of the coupon from *each* neighbor, or at least to know that their neighbor has received a coupon. Receiving a single copy is only the minimum. Additionally, as will be explored in Chapter 6, nodes aren't usually assumed to talk to all other nodes, but rather a subset of nodes in the network.

2.5 Network Coding

Broadcasting is commonly addressed by flooding in DTNs and network coding has been shown to greatly reduce the number of transmissions needed for single-source unicast, multicast, and broadcast in such environments [59], multi-source unicast and multicast when mixing is not done between senders [65], and multi-source broadcast [66–68]. Network coding is as algebraic method where nodes in the network combine (encode) packets together, such that the combined packet is of a *similar* size to the original packets, and all packets can be separated (decoded) at the receiver. Intermediate nodes can also mix and remix encoded messages along the way as they see fit without first decoding.

Network coding is used both to improve throughput, as described above with broadcasting, and also to add redundancy for resilience to transmission loss. Additionally, in a wireless system that uses network coding, nodes can overhear transmissions between other nodes as a method to obtain additional encoded information, thereby reducing the amount of time required to decode new data. Previous studies have demonstrated that wireless overhearing can offer tremendous benefits for information dissemination [69].

For the coupon collector problem, which has been shown to require $\Theta(nlogn)$ rounds for the decentralized flooding, network coding has been shown to reduce this to and $\Theta(n)$ rounds. Network coding solutions can approach the min-cut for the network [70] and the

gain is from the more "even" spread of data throughout the network, since nodes have a higher change of sending something "innovative" (new information for the receiver) by sending a mix of their messages, than by selecting a single message at random [60]. One just needs to keep in mind that these assumptions are not particularly realistic for most application areas: nodes can't always receive an innovative coupon (or even any coupon) in any given time period, and the data distribution is often more uneven. This means that any practical application will be unlikely to match this theoretical improvement.

2.5.1 Random Linear Network Coding

One important network coding technique in ad hoc environment is Random Linear Network Coding (RLNC), which produces linear combinations of packets by multiplying them by randomly picked coefficients from a finite field. Messages can be decoded once a node receives a sufficient number of packets to solve a system of linear equations. This is a *distributed* version of network coding that does not require knowledge of the network topology. I would like to point the reader to [59] as a good primer on this subject, but I will also summarize the technique here as well.

In RLNC, messages are divided into what are referred to as "generations" (partitions of the message space). Messages from the same generation may be mixed together, messages from different generations are never mixed. The issue of *generation management* is the issue of determining which messages (in this case announcements) can and should be combined. For RLNC, all nodes need to have agreement about the "place" in a generation where a given announcement belongs in order to construct the appropriate packets and determine what is mixed in for an encoded packet. When packets are generated from a single source, the source determines which packets to mix, typically by encoding together "blocks" of a specific size.

For the sender who wants to transmit a set of messages, they will choose a random set of coefficients which determine which messages will be mixed, and in what way. For example, if the mixing is done XOR fashion, the coefficients are a bit-vector indicating which messages

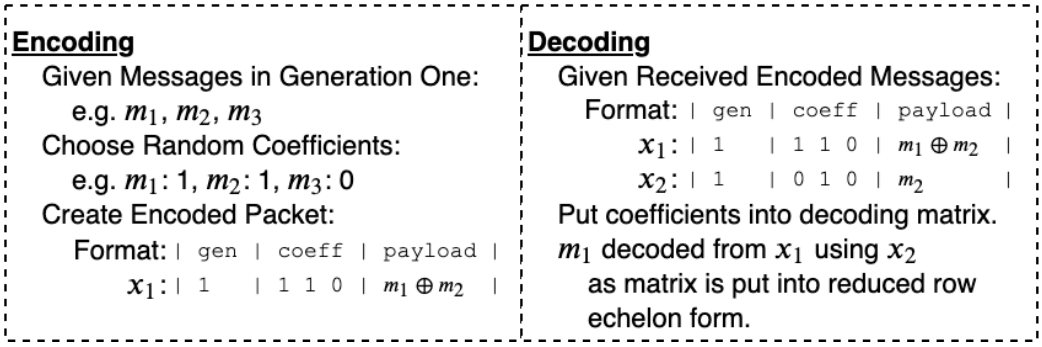


Figure 2.3: Simplified example of RLNC.

are XORed together. These coefficients are attached to the encoded message and referred to as the "encoding vector".

For the receiver, they take the series of encoded messages and coefficients from the encoding vector, and put them into a "decoding matrix". As they transform the matrix into reduced row echelon form, the messages are decoded. Figure 2.3 shows an example of this technique, showing a simplified example of RLNC encoding and decoding using the finite field GF(2) for coefficients (typically GF(8) is used). A coefficient of 0 indicates that message is not mixed in, regardless of the GF. Generations are chosen using generation management techniques discussed later.

At an intuitive level it would seem that RLNC is a a good choice for data advertisements in ad-hoc ICDTNs since data advertisement is a form of broadcasting, many applications for ad-hoc ICDTNs have a lot of multi-path, and throughput is vitally important when trying to utilize short transmission windows common to many DTN applications. However, there are many challenges to a direct application. The research challenges are discussed in Chapters 5 and 6, but there are some additional ones which have been addressed, at least partially, by others: multi-source network coding, generation management, and the parameter space for RLNC. These will be discussed in the following sections.

2.5.2 Multi-Source Network Coding

Typically, network coding is utilized to mix multiple packets from a *single*-source, but data advertisements are not necessarily limited to originating at a single source. In fact, many applications allow all hosts in the network to be potential sources of named data. Another traditional use of network coding is to divide large files up and transmit the pieces coded together, offering redundancy if packets are lost. However, unlike these traditional applications, data advertisements are not large, and not all hosts will have multiple data announcements they want to burst out into the network at once. So if we apply traditional single-source methods of network coding, we will rarely utilize it. Instead we want to mix small messages from many sources together and use *multi*-source network coding with *mixed* sources.

The traditional challenge to mixed sources is that in multi-source unicast and multicast a well-known problem arises: nodes may have to wait to receive information they don't care about to decode the information they do care about [65]. This is a form of "pollution" which is harmful to the information dissemination process. However, for multi-source *broadcast*, it is not possible to pollute the network in this way, since everyone cares about all data advertisements.

Multi-source network coding has unfortunately received little attention for ad hoc mobile scenarios, even without the component of data advertising. For all-to-all broadcasting with network coding, [66] discusses using one-to-all broadcasting applied for each sender (without mixing messages), and [67] discusses mixing messages, presenting a deterministic algorithm for "growing" generations, but their focus is on very small, dense networks (10-100 nodes). A much more thorough investigation is done by [68], including a very extensive analysis of deterministic broadcasting mixed with network coding for mobile ad-hoc networks and an algorithm for generation growth with mixed messages, the networks being looked at were again very small (less than 150 nodes) and very dense (with neighborhood sizes of around 15-30 nodes) compared with many application areas for ICDTNs.

When any node can contribute to a generation, there are only two possibilities for generation management: via an out-of-band communication mechanism or via some deterministic algorithm that all nodes are aware of. An example deterministic algorithm can be found in [67].

Chapter 5 provided a small but significant contributions to this research area, providing a "when to send" heuristic for packet transmission in ad hoc mobile networks and a replacement for "slow start" in such networks.

2.6 Software and Agent Based Models

Due to the scale of the networks being examined, large scale network simulators were needed to complete this work. In Chapters 4 and 5, the DTN simulator ONE is used for experiments [71]. It provides a wide range of existing DTN protocols, such as epidemic routing and spray-and-wait, as well as many primitive movement models. There are some minor limitations for simulating fully connected networks (limited throughput for very small messages), but these are such that they underestimate network performance rather than overestimate it.

I implemented a minimalized version the NDN layer for ONE following the [72], and a variation of SWIM called SWIM-Prox as well [30]. Both of these are discussed in more detail in Chapter 3. In Chapter 5, a RLNC library was needed, so one of the more robust RLNC libraries available (Kodo [73]) was integrated with ONE for this purpose.

In Chapter 7, the Multi-Agent Simulator Of Neighborhoods/Networks (MASON) is used [74]. This is a discrete-event simulator that supports massive multi-agent scenarios. It is written in Java and provides extensive 2D visualizations. The GeoMason extension provides geospatial support [75] and was also used as part of the simulation in that chapter. Agent-Based Models (ABMs) are widely recognized as one of the best approaches to study large-scale systems of autonomous or semi-autonomous entities [76] and are particularly well suited to simulation of disaster scenarios [77].

2.7 Conclusion

Ad hoc ICDTNs are at the intersection of many overlapping research areas: ad hoc networking, mobile networking, delay/disruption tolerant networking, and information centric networking. Additionally, utilizing such networks for disaster relief is well motivated. However, there are many challenges which need to be addressed before this becomes a reality. This work attempts to bridge these gaps and present a holistic approach to the field.

First, a solid, adaptable architecture needs to be decided upon. My work proposes and evaluates such an architecture in Chapter 4. Second, the foundational step of data advertisement needs to be carefully examined and not "left for future research". My work addresses this challenge in Chapters 5 and 6. And third, both the architecture and advertisement protocols need to scalable to not only hundreds of devices, but potentially hundreds of thousands or millions of devices. My work examines these challenges throughout, but special attention is paid to this in Chapter 7.

Chapter 3: Architectural Models and Performance Analysis

In order to conduct a rigorous scientific study of ICDTNs, it is necessary to lay out some basic architectural elements, communication models, motion models, and measurements, which makes it possible to quantify performance. This chapter introduces these basic elements, outlines assumptions being made, and covers implementation details that demonstrate the practical utility of my approach.

I begin with a discussion of the network devices in Section 3.1 and the implementation details for NDN in Section 3.2. This is followed by a discussion of the RLNC parameter space ICDTNs with special attention to the impact of these parameters on the data advertisement problem in Section 3.3. Next, Section 3.4 discusses the causes of disruption and the models used to "create" this disruption. And finally I present a brief overview and discussion of the performance evaluation criteria used throughout this work in Section 3.5. Portions of this chapter have been published or presented as part of [25–27] and work currently under review.

3.1 Network Devices

Devices (nodes) throughout this work are assumed to be relatively homogeneous in storage capacity, have consistent transmission/receiving power, and otherwise function as relatively similar units. Specific parameters may be changed between experiments and chapters, but within any one experiment the devices are identical unless otherwise stated. I also assume infinite energy devices, or at least energy sufficient for the device to function during the simulation period.

Nodes are also assumed to have some shared "concept of the world", such that they agree about things like generation contents, the concept of a certain geographic regions,

their membership in any group, and so forth. In other words, they are not working at cross purposes intentionally or unintentionally. As a concrete example, when devices are using geographical regions for routing, they are assumed to understand where they are in the world, or at least commit to some "belief" that they belong to a given geographical region that makes sense relative to other nodes in the network. For example, it would be ok for a node to believe itself inside Building A when in fact it was just outside the entrance, as long as everyone else believes it too. On the other hand, it would not be ok for a device to believe it was on the other side of the world since this would be inconsistent with other devices which see it within their communication range.

In the same vein, I do not assume that nodes are adversarial or interested in harming the network intentionally. This is a hefty assumption, but there is currently significant work in trust building for ad hoc networks which may fill this gap [78]. For the authentication of data, I assume that secure key distribution has been used, either by building keys into the hardware of the devices or or by traditional key distribution mechanisms used prior to the disaster. This allows devices to authenticate data advertisements as coming from some official source. For message mixing, I assume either intermediate nodes can be trusted or homomorphic signatures are used [79, 80]. See Chapter 8 for more discussion of these security assumptions and future work related to them.

3.2 NDN Implementation

This work uses NDN as its target ICN flavor, as discussed in Chapter 2. This section outlines the specifics of the NDN system which were altered or changed, and the components I designed to enable this deployment in an ICDTN environment. Specifically, I address: (1) changes to the NDN architecture, (2) the concept of a "virtual face" used throughout this work, and (3) the data advertisement structure I created.

3.2.1 Changes to the NDN Architecture

I implemented the NDN layer used following the NDN Developer's Guide [72]. However, the following changes were made for reasons of simplicity:

1. The dead nonce list is storing unique ids of messages instead of (name, nonce) tuples. This allows for a simpler integration with ONE and MASON.

2. The following features were not implemented as they were not required for this work: nacks, the "wants new nonce" flag, longest prefix matching, or any security-related items. All new NDN messages get a new nonce by default. Pre-publishing packets was not required, so all unsolicited data is dropped.

Additionally, the following changes were needed for the DTN environment:

1. The PIT is not using the straggler timer. This timer is set experimentally in NDN and can have no easily determined setting for a DTN.

2. The CS must be checked for all arriving interests due to a race condition between sending out interest messages and receiving new interests (see Chapter 4).

3. The PIT out records keep a lifetime counter. This allows nodes which generate an interest to have out-record with no corresponding in-record and keep the PIT entry alive for face statistics.

3.2.2 Virtual Faces and NDN

As discussed in Chapter 2, interfaces in NDN (usually called "faces") originally represented traditional network interfaces that a message could be routed on. However, in adapting NDN to a DTN environment it has been suggested that "virtual interfaces" could be used to adapt NDN for different scenarios [22].

Generalizing this idea, I propose in this work that a virtual face layer can be used to create *any* arbitrary subset of the network and that NDN can use this to represent it's "goal" in a hybrid NDN/DTN network. As another way of looking at this, the face layer

under NDN allows it to filter incoming or outgoing messages in some way to hide from NDN the details of the underlying network and simplify the routing tables. This will be discussed in more detail in Chapter 4, but I want to address here the fundamental question of 'What does an NDN face represent?" and how this choice impacts the resulting network. Below are some example face types for ad-hoc networks which NDN can use:

Faces represent a specific node. Every node has a unique id and is represented as its own face. This is very granular and good for delivering messages to specific nodes which regularly come in contact with the sender, as would be the case with the data mules. One drawback is that this can produce extremely large PIT/FIB tables (worst case O(NodeSpace*NameSpace)) and changes in node movements can invalidate PIT entries.

Faces represent a geographical direction ("geo-faces" [22]). In this setup, messages are forwarded in a direction (e.g. north, north-west, etc.). This ensures a fixed number of faces at any node, producing worst case PIT/FIB table size of O(NameSpace). The underlying face is able to make use of opportunistic contacts, but node movement may require regular updating of the FIB and movement can invalidate the return-path routing via the PIT.

Faces represent a geographical region. With faces of this type, messages are forwarded to a specific geographical regions (geo-region) rather than a direction, for example, a grid square in Military Grid Reference System (MGRS). Compared to representing directions, this type of face ensures that node movement will not invalidate FIB entries. On the other hand, data producer movement could require regular FIB updates and the worst case PIT/FIB size is O(WorldSize*NameSpace) without careful setup. My work in Chapter 4 provides an implementation that requires only O(NameSpace).

Faces represent all nodes in a DTN. This is the choice for the "low" integration discussed in Chapter 2. Messages are sent from NDN to a single DTN face which transmits the message into a separate DTN network.

Other categorizations of nodes are certainly possible, for example group/non-group, trusted/untrusted, as well as combinations of the above face choices (e.g. there is an "all nodes" face, a "north" face, and a "trusted group" face). However, the above face types

represent the virtual faces most applicable to a mobile DTN environment. In this work I focus primarily on geo-region faces, but I also use node faces in several sections to simplify examples, highlight certain challenges, or show performance for different network types.

3.2.3 NDN Advertising Structure for ICDTNs

The data advertisement problem is addressed in Chapters 5 and 6 but it is important from an implementation standpoint to understand the exact structure being sent during this process. I have created a Type-Length-Value (TLV) NDN packet for this purpose which takes the advertisement header from NDN's "self-learning" interest packets [53] combined with NDN's standard conventions for naming and signing data. This format is designed to fit in with NDN's other data structures and is assumed throughout this work for all unencoded data advertisements.

Figure 3.1 shows the packet structure. T(#=X) is an NDN TLV type where # is the value (using 1, 3, or 5 octets) and X is the human-readable type, LEN is a TLV length (using 1, 3, 5, or 9 octets), NNI(X) is a non-negative integer defining X (using 1, 2, 4, or 8 octets), OCTET is a byte sequence in network-byte order. Also '*' means "any number of", square braces are used for optional components, and '(A / B)' means "A or B".

The announcement requires a type (848 is the chosen code from [53]), a length, and some data (the advertisement itself). The name and signature are in standard NDN message format, so both consist of a number of other component parts. The meta-data might be removed, but having a content-type tag in the meta-data is consistent with other NDN packets. The expiration period indicates when the announcement expires and should no longer be transmitted, while the period when the announcement is valid indicates when requests should start (and stop) being forwarded by the FIB. This would mean that an advertisement, in its smallest format, would be about 168 octets (assuming a very short prefix name, key name, and a 128-octet signature) or around 300 octets with a 256-octet signature. This announcement, or a mix of several announcements in the case of network coding, is then be packed into a DTN bundle or other appropriate lower layer structure.

announcement ::=	T(848=Prefix-Ann.) LEN data
data ::=	name meta-info content signature
name ::=	T(7=Name) LEN *name-comp
name-comp ::=	T(8=Gen-Name-Comp) LEN *OCTET
meta-info ::=	T(20=Meta-Info) LEN meta-data
meta-data ::=	T(24=Content-Type) LEN NNI(5=Announce)
content ::=	T(21=Content) LEN exp-period [valid-period]
exp-period ::=	T(Exp) LEN NNI(ms-until-exp)
valid-period ::=	[not-before] [not-after]
not-before ::=	T(Valid-Start) LEN NNI(unix-ts)
not-after ::=	T(Valid-End) LEN NNI(unix-ts)
signature ::=	sig-info sig-value
sig-info ::=	T(22=Sig-Info) LEN sig-type [key-loc]
sig-type ::=	T(27=Sig-Type) LEN NNI(sig-type)
key-loc ::=	T(28=Key-Loc) LEN (name / digest)
digest ::=	T(29=Key-Digest) LEN *OCTET
sig-value ::=	T(23=Sig-Value) LEN *OCTET

Figure 3.1: NDN announcement format for ad hoc ICDTNs.

3.3 RLNC Parameters in Ad Hoc ICDTNs

The two major components of RLNC for ICDTN advertisements need to be discussed: generation management and the effects of RLNC parameters on message delivery and decoding. The following discussions are necessary to for understanding the implementation choices used in this work. A formal packet structure for encoded announcements is presented in Chapter 5 as there is a larger discussion needed to understand this.

Note on Network Coding Integration. For coding announcements sent by NDN, I integrated Kodo with the ONE simulator. ONE is written in Java and Kodo in C++, so I wrote a Java Native Interface (JNI) module to allow hosts in ONE to have Kodo encoders and decoders. Since ONE messages cannot be serialized, I created a "secret" 800-byte array of randomly generated values for each message (the size of a large announcement). The Kodo encoders and decoders used these byte arrays as their symbols and only when valid secret

values were decoded in Kodo where messages in ONE considered decoded.

3.3.1 RLNC Generation Management

The issue of *generation management* is the issue of determining which messages (in this case announcements) can and should be mixed together. For RLNC, all nodes need to have agreement about the "place" in a generation where a given announcement belongs in order to construct the appropriate packets and determine what is mixed in for an encoded packet. When packets are generated from a single source, the source determines which packets to mix, typically by encoding together "blocks" of a specific size.

When any node can contribute to a generation, there are only two possibilities for sharing such information: via an out-of-band communication mechanism or via some deterministic algorithm that all nodes are aware of. As noted in Section 2.5, an example algorithm can be found in [67]. Generations might be formed by time epochs, by specific producers, or by the data being advertised, but these categories can be mixed as well. The important thing is that the grouping *must* be understood by all nodes and cannot be misinterpreted by any encoder or the network will become polluted (encoded messages are never decoded because of improper mixing).

I now outline some categories of generation management strategies and document their trade-offs and use cases for data advertisements. Table 3.1 shows this general breakdown, but these categories can be mixed. Table 3.2 gives a breakdown of how the types of data producers affects the generation management choices. In this work I use single large generations and generations divided by time epochs.

The multi-source network coding works referenced in Section 2.5 choose to use very small generations based on [81] which showed that this amount of encoding provided a substantial benefit in their experiments (and larger generations did not increase the benefit). However, since [81] did not seek to prove that this was universally true, simply an experimental result for their scenarios, I explore much larger generations and show that a heuristic for partial decoding of messages provides similar benefits (see Section 5.3).

Table 3.1: General Generation Dividers for Announcements

Generation Divider	**Use Restrictions**	**Example Use Case**
None	Global coordination possible.	A few messages valid for the life of the network.
Epoch	Clocks with similar time.	Advertisements occur (or change) over time.
Producers	Identifiable groupings of producers.	Many data producers. See Table 3.2.
Data	Identifiable data similarities which can be used for coordination.	Many advertisements with the same prefix, expiration time, valid start time, etc.

Table 3.2: Producer Generation Dividers for Announcements

Prod. in Gen.	**Use Restrictions**	**Example Application**
Single Producer	Data producers are advertising multiple items with separate advertisements.	An emergency center advertising food, water, and other supply availability as separate data.
Coordinated Groups	Each group is able to coordinate through a back-channel or offline.	Firefighters, rescue workers, and police. Everyone in Building A and everyone in Building B.
Uncoordinated Groups	A deterministic algorithm is needed for coordinating within a generation.	People in need of rescue (more generally, any unknown number of data producers who have no mechanism for offline or back-channel coordination).

3.3.2 RLNC Parameter Space for Data Advertisements

To understand the challenges faced when applying RLNC to ICDTNs for data advertisements, it helps to focus on the parameter space available, understand the relationship between each of these configurable components, and see how they relate to the sending of announcements. I will discuss here two groups of tune-able parameters: local and non-local. Local parameters can be chosen by each node independent of all other nodes. Non-local

parameters, on the other hand, are outside the control of individual nodes, but may or may not be outside the control of the network. I will not discuss parameters outside the network's control, such as the likelihood of two nodes coming into contact, while such parameters are very important, I am concerned here with what we have the power to change.

I begin with a discussion of the controllable parameters at the network level: generation contents and generation size.

The Effect of Changing the Generation Contents. Generation management was touched on in the previous section, but the primary effect is that choosing different generation dividers impacts the generation size (for example, all announcements created in the last hour will be a smaller generation than all announcements created in the last day). On the other hand, the more generations "in the air" at any one time affects, the less likely any one message being sent if from any one specific generation.

The Effect of Changing the Generation Size. Increasing the generation size decreases the chance of decoding a message at any one time, but increases the chance that a given announcement is in a given generation (and in turn this increases the chance that a given announcement is sent in any given encoded message). This is simple to see since generations are typically partitions of the announcement space, and encoded messages only send announcements from the same generation, the fewer partitions there are, the more likely any one partition is chosen.

Decreasing the generation size, on the other hand, has the opposite effect, increasing the chance of decoding some announcement from the encoded packet, but decreasing the chance any one announcement is sent. Taken to an extreme, if a generation were picked such that there was one announcement per generation, we would be back at flooding (generations/announcements are decoded at the same time they are sent, but there are many more generations/announcements to choose from for any given send).

Figure 3.2 shows what generation size looks like from the perspective of a single node. It must receive one packet per "mixed in" announcement in the generation before any one item from the generation can be decoded. However, as soon as all enough encoded packets have

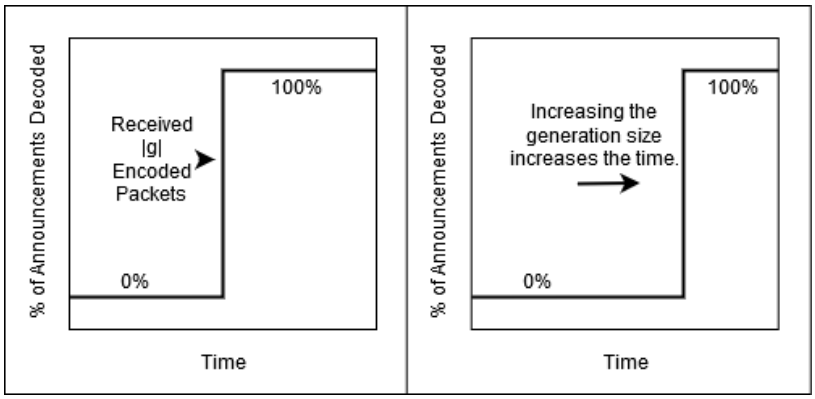


Figure 3.2: Relationship between generation size and the time to decode a generation.

arrived, the node can decode all the mixed in announcements. Increasing the generation size "pushes" the time until enough packets are received to the right, so the chance of decoding at any given packet send is smaller.

Node-Local Parameters. The parameters an individual node can play with are the coding density and the coded announcement contents. Both of these are functions on the coding coefficients for encoded announcements, setting various coefficients to 0s to remove them from the mix.

The Effect of Changing the Coding Density. The "coding density" (also called the "degree of encoding") can be understood as controlling the number of symbols (messages) a node chooses to mix into an encoded message, though the actual effect on re-encoding partially decoded messages is actually significantly more complex. We can see this from the following example: if a node has received a single encoded message in the past which was a mix of all symbols in the generation, when choosing to send, even if the density was set to some infinitely small number, the node must in fact send a mix of all symbols in the generation. This is because the node, at this time, has no mechanism for dividing the encoded symbols into their respective parts; if it could, this would imply nodes could decode all the symbols upon receiving a single message. Therefore, the density attempts to reduce the number of symbols mixed together, but it does not guarantee that nodes will always

achieve their desired density.

The second local parameter is the choice of which specific announcements to mix into a given encoded message (or optionally to mix in random packets). To utilize choosing specific packets properly requires information exchange between nodes or some mechanism to predict what the node might have, but this is often seen in static networks.

The Effect of Choosing Specific Announcement. Choosing which messages to send can have an immediate benefit for two nodes sending in a static network. To decode any one message from a packet mixed using network coding (such as our announcements), one needs only be "off by one" between the received message and what already exists in the node's decoding matrix. For example, in a simple XOR mixing scheme, if a node receives a coded announcement (x_k) and already has another coded announcement (x_m), if $x_k \bigoplus a = x_m$, where a is a specific announcement that is mixed in, then one can decode a using those two packets. Additionally, one can replace x_m (the previously received message) with any combination of encoded messages which together form x_k (i.e. the node can reconstruct x_k from its decoding matrix).

An example of utilizing this "off by one" scenario is commonly used to "slow start" sending packets in static networks. Let's say that a node (n_i) has all of a given generation (g_1) and wants to send it to another node (n_j). To "slow start", n_i sends the first encoded packet (x_1) with only the first announcement from g_1 (let's call it a_1), then the second encoded packet (x_2) with only the first two announcements from g_1 (a_1 and a_2), and so on. This uses network coding for helping with resilience to packet loss, and at the same time allows decoding most announcements as they arrive. An alternative to this for multi-source ad hoc networks is presented in Chapter 5.

The Effect of Announcement Expiration Time. While it may seem like an obvious choice to simply increase the announcement expiration time in order to increase the time to utilize the announcements, this parameter is generally controlled by the data producer. It is considered bad practice to base the data availability on the underlying network, but it may be useful for data producers in some scenarios (such as disaster workers in an emergency

situation) to be aware of its impact.

3.4 Disruption Models and Network Communication

As discussed in Chapter 2, there are many possible causes of disruption, such as movement and transmission interference. This work will use five types of models for causing disruption: the random waypoint motion model, a variation of the SWIM human motion model, a "broken" WiFi interface model, a static connection probability model, and a two-state connection probability model. In all models the following assumptions are made: (1) messages arrive or don't arrive, they are never jumbled, (2) 'stack time' does not exist, i.e. it does not take time to process a message when it arrives, and (3) nodes are assumed to know about only a limited portion of the network.

For (3), this is different from the common coupon collector problem used to study broadcasting. In that model every node can potentially send to any other node, whereas I restrict nodes to communication based on (1) physical location and range communication range, if location is part of the model, or (2) a pre-set as a list of "potential" neighbors, for the connection probability networks.

3.4.1 Disruption Due to Movement

In order for motion to cause disruption, nodes are only allowed to communicate when within a certain range. Transmissions can fail if nodes move out of range while transmitting, but both nodes are assumed to be aware of this failure. The most basic model used for this is random waypoint, and in my implementation of this nodes "walk" at speeds of 0.5-1 m/s (a slow meander), and pause of 0-30 seconds at each location. Their transmission ranges are set per-experiment.

I developed and implemented a variation of the SWIM human motion model [30] designed to work for environments which do not have a concept of "home" (such as a college campus). My variant also allows agents to move around an area once they arrive, which

is not part of the original SWIM model. In summary, this works as follows: people are in buildings, performing a random walk inside, or walking between buildings using roads. They remain in a building for 1-5 minutes and then chose a new building to go to, weighting their choices by the building's popularity and proximity. This means that "home" from the SWIM model is replaced with "current location", and I call this SWIM-Prox(imity) to distinguish it from the original model. The SWIM parameters used are 0.25 for alpha, so that people care more about the popularity of a building than the distance, and 0.05 k, which is used to scale distances. The initial popularity of a building is seeded based on the *size* (area) of the building so that, with no other knowledge, people will assume large buildings are popular places.

The particular implementation in Chapters 4 and 5 uses the Fairfax Campus at George Mason University (GMU), with the paths, roads, and buildings imported from OpenStreetMaps [82]. There are two circular buses that travel around the map, stopping at random locations along the outer ring, and people will ride these if it would reduce their walking distance. Due to how the campus is built, buses are mostly only used to get from the northern most buildings to the southern most buildings, which matches the real campus. People move at walking speed (0.5-1 meters/sec) and buses move at slow car speed (7-10 meters/sec). The buses can also communicate with the human nodes, the assumption being that the bus driver would also carry a phone or other small device.

3.4.2 Node Placement with Probabilistic Network Disruption

When experimenting with non-mobile DTNs, nodes are placed in random locations, with the same initialization as the random waypoint model, and I created a "broken" WiFi interface which has only a 10% chance to connect each decisecond it is in range of a node, and a 90% chance each decisecond it was not transmitting to disconnect again. This creates randomly fluctuating connectivity in the network and causes a noticeable delay in message delivery. Note that in this model nodes are still limited to connections with *range*, so they can only ever potentially connect to a subset of the network due to the non-mobile nature. This is

a form of two-state probability connection network (discussed in Chapter 2) but with "real world" restrictions on location and potential neighbors.

3.4.3 Connection Probability Models

For more analytical sections of this work (specifically Chapter 6), the PUSH model is used along with discretized rounds for time. Like with most setups for the coupon collector problem, with this model a single node is able to send to one neighbor at any given time slot. This means that the model ignores interference and nodes can receive from more than one neighbor at the same time. An entire message is sent as a unit during this time, so there is no distinction between larger or smaller messages, thus sending one unencoded or coded announcement is considered to be one transmission.

These probabilistic modeling rely in part upon a characterization of the underlying contact and inter-contact distributions. These distributions are generally specified by a rate parameter λ. Previous empirical DTN studies of human mobility patterns show that these distributions are well approximated by exponential [34] or power law + exponential tail distributions [31], and I base my analysis on this distribution class.

3.5 Performance Evaluation and Measurements

In order to evaluate the network performance, I use a number of different metrics, some of which are common to other areas of networking, and some which are not. I summarize them here for reference.

DTN performance is typically measured by packet delivery ratio (PDR) and overhead, either in number of packets or in number of transmissions sent. These measurements are preferable to measurements related to time, since the natural delays in the DTN environment dominate any network transmission delays. Overhead is important since, as discussed in Section 2.1.2, flooding and multi-copy techniques are quite common.

On the other hand, ICNs tend to measure the number of interests satisfied out of the number of interests generated (interest satisfaction ratio or ISR) and the time to satisfy the

interests (interest satisfaction time or IST). Again, this is due to the network goal being information dispersal and not host-to-host communication.

The coupon collector and broadcasting problems typically study the time the number of "rounds" needed to converge or solve the problem. In systems which do not have discretized rounds, the equivalent is just "time to completion".

In Chapter 4, I focus on ISR, IST, and overhead; and I also note here that PDR is encompassed in ISR. I measure overhead "end to end" looking at how many transmissions were generated by a given NDN "kickoff" event, such as a new interest generation from an application or a data reply at a data producer. This means that all transmissions along all hops (and leaps, see Chapter 4), are counted together as being part of the overhead for the given initial interest or data reply.

Chapter 5, focuses on the time for all nodes to have some percentage of the data announcements ("time to completion"). The time for 100% of the network to receive 100% of the announcements defines what I call the "announcement period" (discussed more in Chapter 6). When studying data advertisement with motion-based disruption, I focus on the time for "Y% of the network to have X% of the advertisements". This is because nodes could theoretically spend the entire simulation disconnected from the network and because moving in the world can temporarily reset the FIB tables, such that they have the announcement but no routes for it.

The remaining chapters have application-specific measurements. Chapter 6 examines the component equations that determine the announcement period and receive times. Chapter 7 examines the a specific, city-scale disaster scenario.

3.6 Conclusion

In this chapter I have given the basic architectural, movement, and communication models needed to measure city-scale ICDTN performance. As part of this, I have offered concrete implementation details for NDN, including an analysis of the design choices for virtual faces in NDN and the data advertisement structure, explored the RLNC parameters for the data

advertisement problem, and detailed the models used for disruption. This provides the basis for a rigorous scientific study of my proposed architecture and announcement protocols for ICDTNs in the remaining chapters.

Chapter 4: Geo-region Based Architecture for ICDTNs

In this chapter I examine an architecture to create ICDTNs such that the ICN and DTN components *compliment* each other. This is done by adding a simple middle layer between two existing ICN and DTN frameworks to make the ICN resilient to disruptions but also to direct the underlying DTN to support the ICN (Section 4.1). Using NDN and two DTN protocols (epidemic routing and spray-and-wait) I evaluate two proposed middle layers, study the effect of different motion models and network parameters on the results, and motivate this work with a disaster scenario using emergency centers (Section 4.2). I demonstrate through simulation the performance in a variety of network types and sizes and, for the disaster scenario, I show how a unique combination of NDN and DTN is able to scale to large networks of 1000+ mobile nodes. This chapter has been published in [25] and presented in [26].

4.1 A Middle Layer Approach for ICDTNs

In Section 2.3, I discussed the current ICDTN implementations and described them as either a "low" level of integration, where the ICDTN is composed of two separate networks with gateway nodes, or "high" level of integration where the ICN and DTN protocols are heavily modified to merge the components of one technology into the other. This section outlines a third architecture option, a "medium" level of integration.

This hybrid approach is aimed to retaining the best of both techniques. In the medium integration level, the ICN works to "direct" the DTN, while at the same time the DTN component performs the disruption handling needed by the ICN. A useful analogy is that the ICN provides the "next leap" and DTN provides the "next hop". To make clear the differences in this solution, in the low level integration there would be no in-network caching

within the DTN section of the network and there would be no useful strategy to gather statistics about how to forward *within* the DTN (only *to* the DTN). On the other hand, while a high and medium level of integration both serve the same purpose (all nodes have the ability to store, carry, and forward), in the medium integration the ICN and DTN protocols are left completely intact and can be replaced with different protocols as needed, but in the high integration level the ICN and/or DTN are modified to "fit" into the other network type.

The benefit of this layering to NDN is obvious (disruption tolerance), but the benefits to the DTN are also interesting: (1) NDN can provide in-network caching without the need for dedicated cache servers, (2) NDN can guide replies back, even if the network has completely reconfigured, (3) NDN can gather "face statistics" which can be used to better direct the DTN packets and choose good parameters for the DTN protocol, and (4) NDN can help scale the network since the DTN only has to reach the next leap and not across the entire network.

Figure 4.1 shows a comparison of the three architectural approaches discussed. At the top of this figure is a pair of networks interfaced together using the "low" level of integration. In the middle are modified protocols for NDN to work in a DTN environment using a "high" level of integration. The lower portion shows the layered approach where each node makes contextual decisions about which protocol to use ("medium" level of integration). In this last integration option, the ICN placed at the top of the protocol stack, an interface layer between, and the DTN as the "first stop" for incoming messages.

4.1.1 Design Considerations

As discussed in Section 3.2.2, there are a number of different types of "virtual faces" that NDN can use to abstract away some of the details of the network. It needs to be noted at this point that different integration levels use faces differently. In most high level integrations, faces are used at the *sending* node to filter messages as they leave the host (basically forming multiple message queues, one to each face). In a medium integration, the face layer is used

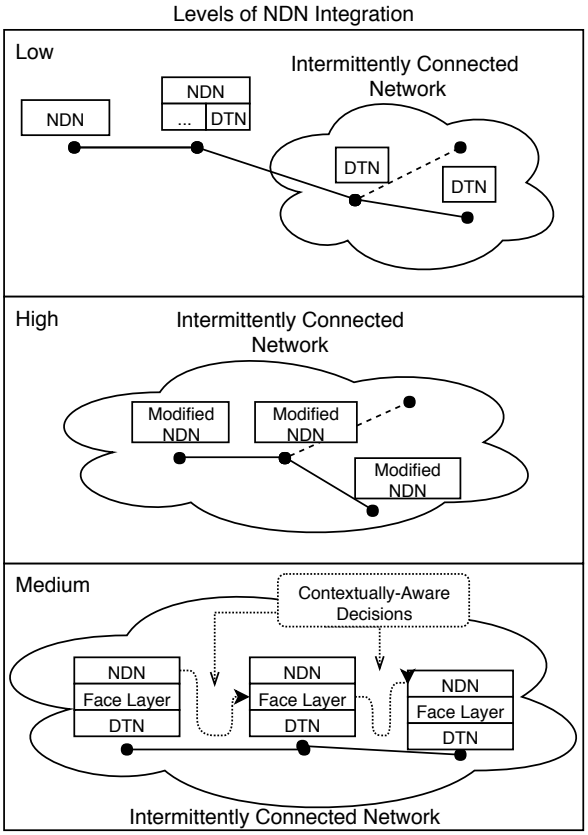


Figure 4.1: Integration levels for NDN+DTN integrations.

on the *receiving* node to determine when to deliver messages to the ICN and when to leave it at the DTN level. This difference is important as it is what allows the DTN to act as a completely separate layer with its own routing/forwarding protocols, rather than merely performing store-and-carry for the ICN.

To give a concrete example of this, let's say a face represents a specific node in the network. In this case, if a message from n_0 is destined for n_1 and faces are used to group *outgoing* messages, then when the two nodes were in contact, n_0 could send on its interface for n_1. n_1 will accept any incoming messages and tag them as coming from the appropriate incoming interface. This is a simple store-and-carry implementation and the DTN does not act as a separate layer with its own routing/forwarding protocols (instead it is integrated into the face layer and NDN).

On the other hand, if *outgoing* messages are tagged and faces are used to group *incoming* messages, a node can decide if the incoming message needs to be handled by NDN, or if instead it can be continued along using DTN protocols. This means that nodes don't have to wait to encounter a node to send to it. This is the hallmark of the medium-level integration: NDN isn't modified to do store-and-carry, the middle layer determines if NDN is "interested" in a message based on the *destination* face (the next "leap").

I have chosen to use geographical regions (geo-regions) and nodes as faces/leaps for the majority of this work. The node face type, as discussed in Section 3.2.2, is appropriate for networks where nodes regularly meet a small number of "neighbor nodes" on a regular basis, such as with data mules. This face type is easy to understand for those familiar with more traditional DTN networks, so it is used to (1) demonstrate issues with simple examples, and (2) illustrate the generality of my layered approach.

However, one of the main contributions of this chapter is the geo-region face type for scaling to city-size networks. As will be shown in the next section, this type of face borrows many of the techniques of geo-routing at the "leap" level instead of the hop level. The next section gives an example of a geo-region face type, then discusses design choices specific to raising a message to NDN on the receiving end.

4.1.2 Geo-regions for a Middle Layer ICDTN

I will now describe how to use geo-regions as a face choice through an example. For this example, there are three geo-regions and six hosts as shown in Figure 4.2. Host A has some named data (/dataA) and it can advertise this data such that the FIB for hosts B, C, and D will contain /dataA $\rightarrow$ 1 (indicating that the best place to forward requests for /dataA is Region 1 which each has mapped to face 1). Hosts E and F, however, can have entries /dataA $\rightarrow$ 2, providing the "next leap" only to neighboring regions. This allows the FIB (and PIT) to have O(NameSpace) table sizes rather than O(WorldSize*NameSpace). Figure 4.2(a) shows the setup for the scenario and FIB tables, (b) shows how an interest message transfers from Region 3 to Region 2 for Node F looking for data from Node A,

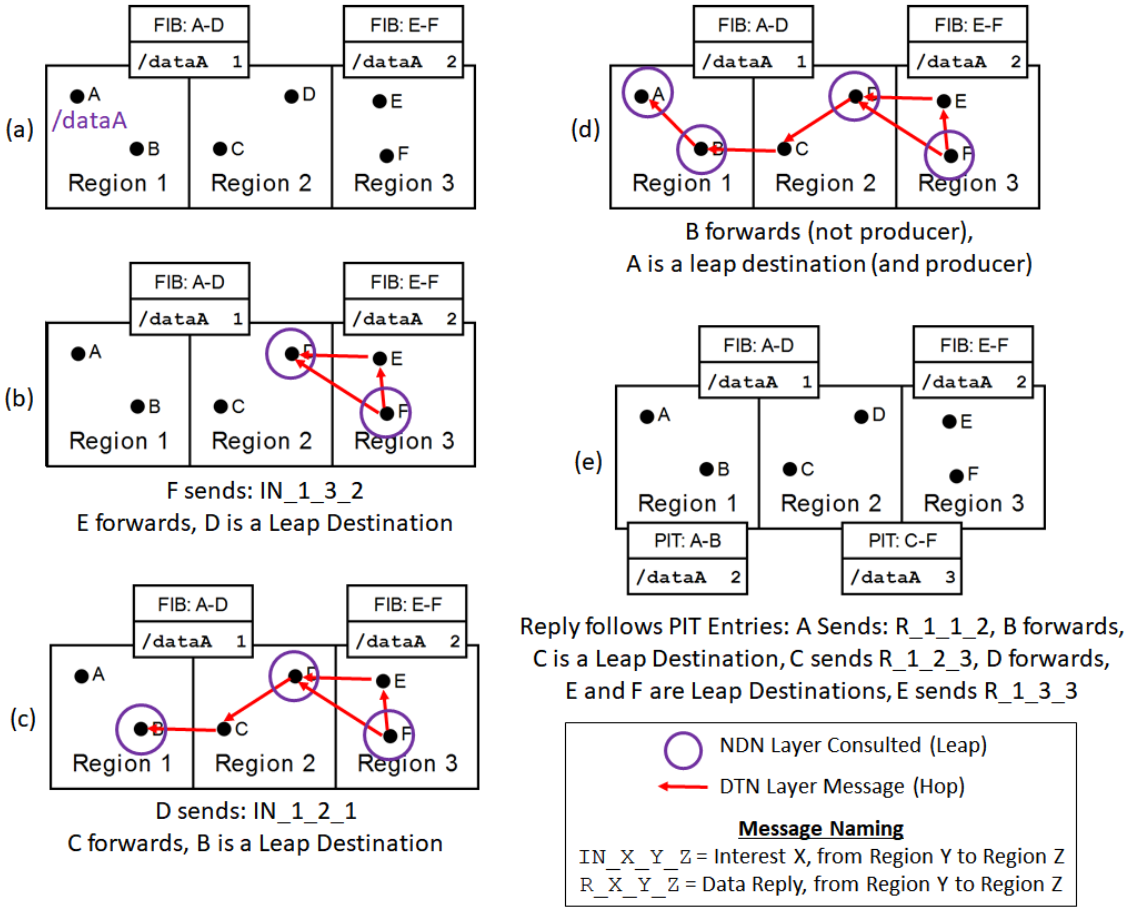


Figure 4.2: Geo-region architecture example.

(c) shows how the interest travels to Region 1 from Region 2, (d) shows how the interest continues within the region to Node A, and (e) shows the data reply following the PIT entries created.

If Host F wants /dataA, it will generate an interest message following the typical NDN process at the NDN level and send it down to the face layer indicating that it wants to send to Region 2. The face layer will tag the message with the current node's NDN region (in this case 3) and the destination region (2), this message is then passed down to the DTN level with a message ID of "IN_1_3_2" (interest number one, from Region 3 to Region 2). This message naming scheme is just showing information attached to the message, it is

useful when discussing messages.

If Host E is within communication range of F and receives the message at the DTN level, it asks the face layer if NDN has any interest in the message. The face layer looks at the destination for this message (Region 2) and declines. Host E will then forward the message on to anyone else it meets (in this case D). When Host D receives the message, its face layer will pass it up to NDN for processing in the "incoming interest pipeline". The new message created by NDN will be forwarded to Region 1 and passed back down to the face layer to be tagged and passed down to the DTN (as IN_1_2_1) and NDN will follow the "outgoing interest pipeline". The remaining leaps and hops follow the same pattern, Host D sends a DTN level message (which goes to C and E), C does not send the message up to NDN since it is not the destination region and continues the message to B.

At this point, B realizes that it is a leap destination and that it is in the current region for the data. B forwards the message within the region, and all nodes within the region which receive the message will check to determine if they can satisfy the message. In this way, Host A receives IN_1_2_1 and can provide the data, so it creates a data message reply for Interest 1 which came from Region 2 (R_1_1_2), according to the PIT entries created by NDN as the interest message traveled). This eventually delivers R_1_2_3 to F and satisfies the interest.

It should be noted that, depending on the communication range of the hosts, multiple hosts within a region may generate the same NDN message. For example, if F can communicate directly with E, D, and C, and sends IN_1_3_2 to all of them, then IN_1_2_1 might be generated at both nodes C and D. This is fine as long as two interest messages with the same id are treated the same at each host. NDN will not forward on interests in /dataA if it already has a pending request for that data and the DTN nodes can treat messages with the same ID as duplicates.

How is movement handled? As discussed in Section 2.4, constructing and maintaining FIB tables for forwarding is one of many challenges left unspecified by the NDN community. Maintaining the FIB for nodes which can change faces, such as the geo-region

faces we are using, is an important step since, when nodes move from one face to another, their FIB tables may be incorrect. To handle this in geo-region faces nodes moving from one region to another simply dump their FIB tables and any routing messages they are carrying. They then take the FIB table (and PIT in-records) from the next same-region node they come in contact with. All NDN-forwarding is put on pause until the FIB is rebuilt. Any nodes within the same region also merge their FIB tables when they are in contact. This in-region merging is not a necessary step, but it cuts down on duplicate messages. Nodes with pending interests resend them.

To ensure that nodes with pending interests receive their data when they move, once the FIB tables are recreated, any interest messages generated at the host which have not been satisfied are resent (as messages from their new region). If the node has moved further from the data source, this ensures there are PIT entries pointing to their new region. If the node has moved towards the data source, nodes which already have PIT out entries will ignore this new message.

In Chapter 5, I address other mechanisms for FIB table construction. In Chapter 7, I also explore and address some of the issues with regions that become completely "abandoned" (empty).

When and how should the face layer raise a message to NDN? As described in this chapter, NDN will be providing the "next leap" while the DTN provides the "next hop". This implies a node needs some basic information and that the answers will depend on the face choice made for the given network: (1) what is the "leap destination" of this message? and (2) am I part of the "leap destination"?. In order to correctly make entries in the PIT and gather face statistics for the FIB, some additional information is also needed: (1) what "leap destination" was the sender in and/or (2) what was the "leap destination" of the previous NDN node?.

If faces represent single nodes, simple "to" and "from" fields are all that is needed. If a face represents a geographic direction, one needs information about the last NDN node's position, the current node's position, and the destination position. For geographic regions,

information about the region of the last NDN node, the current node, and the destination are needed, but an additional field (the sending node's region) is also useful.

In order to get a sense of how the layers actually work, Figures 4.3-4.5 give the details of the interconnections between the layers. As shown in Figure 4.3, for incoming interests the face layer checks the leap destination to determine if NDN is needed. If it is not needed, it is sent back down to the DTN layer. On the other hand, if NDN is needed, the message is passed to NDN and put into the Incoming Interest Pipeline (see [72]). If the node does not have the data, but has a path to the data, a new message is sent back down to the DTN layer and the Face Layer tags these with any updates to the leap destination and other fields listed above. The DTN layer handles all DTN layer protocols on messages before they go out (such as decrementing the number of copies for SaW). When messages are sent at the DTN layer, there is a second check at the face layer to determine if NDN's Outgoing Interest Pipeline needs to be invoked.

Figure 4.4 shows the process for incoming data. This is similar to incoming interests, but here the Incoming Data Pipeline is split. This needs to happen since the PIT shouldn't be marked as satisfied until the message has actually been sent out into the network. This allows any duplicate incoming interests (e.g. from other nodes in the same region) to be satisfied by data sitting in the outgoing queue instead of generating new interests.

Figure 4.5 shows the extremely simple implementation needed for a face layer. In later sections I will show how easily these can be swapped to create completely different types of networks using this generic structure for face layers. The only tasks a face layer needs to perform is (1) identification of the node's face type in the network (e.g. a node's ID, a geo-region the node is in, etc.) and (2) applying tags for a given face.

What coordination is needed for the NDN pipeline? Examining the NDN pipelines for interests and data you can see that a delay between the incoming pipeline and the outgoing pipelines can cause undesired behaviors at the NDN level. These pipelines do not behave as atomic actions since there is often a delay between when a message comes in and when a message can go out due to the DTN environment. Figure 4.6 shows what

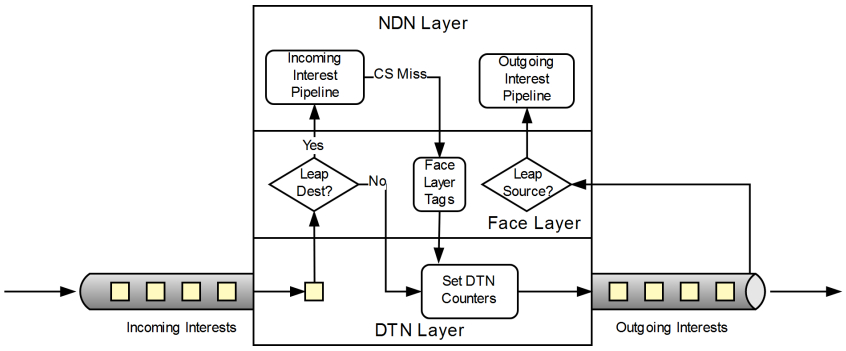


Figure 4.3: Layer interactions for incoming interests.

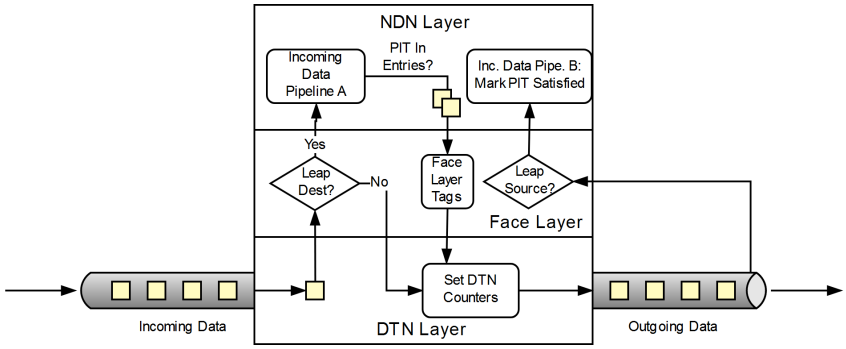


Figure 4.4: Layer interactions incoming data.

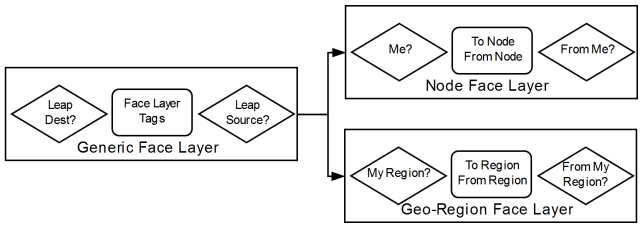


Figure 4.5: Face layer decisions for node faces and geo-region faces.

happens if the interest pipeline is processed in full, before a data pipeline processes. In this figure, six node network receiving data from one producer (P). Nodes with grey stripes have an announcement but don't any pending interests, nodes in yellow have a pending interest but no data, nodes in blue have data. Arrows show messages passed at each step.

At step (c) an interest arrives at the same time as data response for a prior interest.

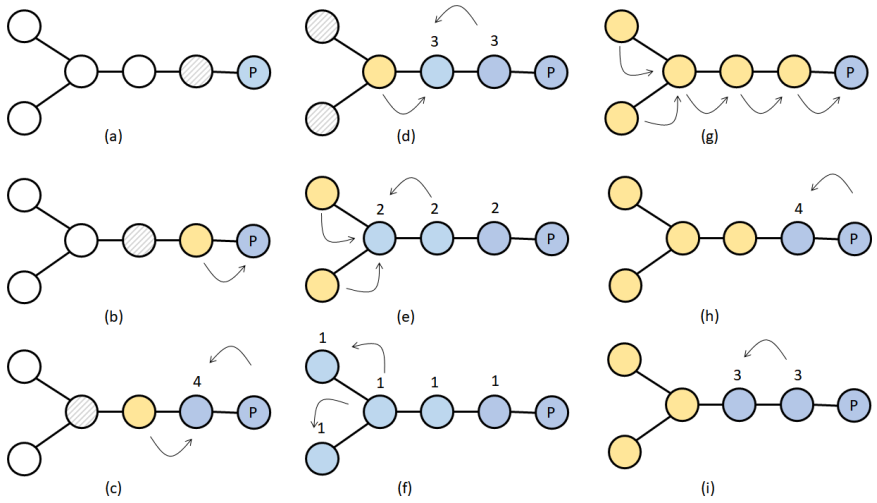


Figure 4.6: Race condition where interest pipeline finishes before data pipeline starts.

In this case, when an interest arrives, it adds to the PIT in-records but does not trigger additional messages to be sent out. This is the best-case scenario.

Figure 4.7 gives the same network when the content store is off and an interest pipeline triggers after the data pipeline. Here the data pipeline clears the PIT out-records and does not continue the message towards the new interest request. This causes the interest pipeline mistakenly think the interest needs to be be continued towards the data producer for a new message, even if the data has not actually been sent out yet due to DTN level delays. This scenario is the reason in the previous discussion that the data pipeline is split within a node.

Figure 4.8 shows what can happen when the content store is off and the two pipelines are occurring simultaneously (e.g. if different nodes in the area are each processing one pipeline unbeknownst to each other). The data *incoming* pipeline can clear out PIT out-records, then the interest *incoming* pipeline adds new PIT in-records, then the data pipeline continues the replies, and then a new request is forwarded towards the producer. This, interestingly, creates a continuous cycle of updates if which keeps data on the edge of the network fresh longer (at the obvious cost of additional overhead).

The last design choice is related to the above discussion. The two pipelines can be made

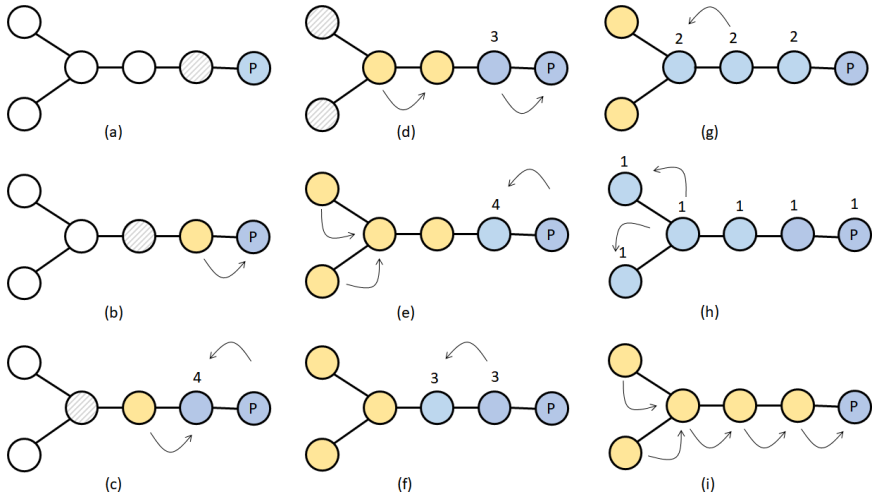


Figure 4.7: Race condition where data pipeline finishes before interest pipeline starts.

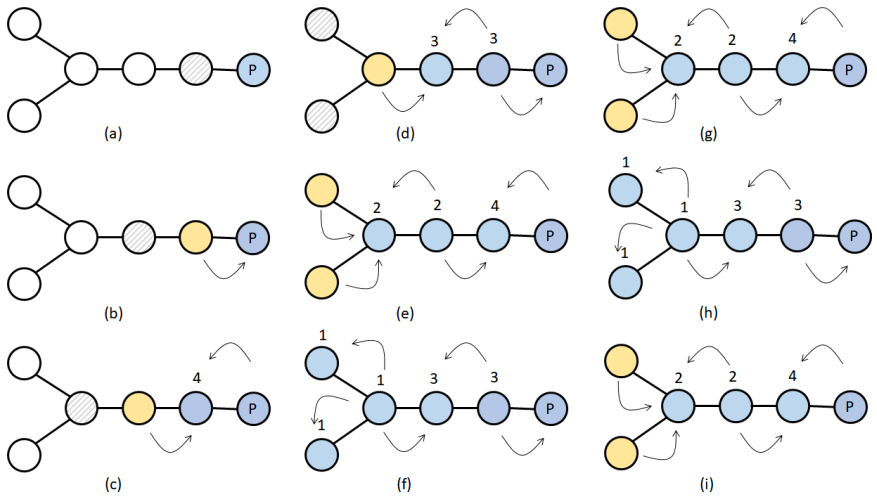


Figure 4.8: Race condition where the data and interest pipelines are intertwined.

to coordinate *within* a node with modifications to NDN to absolutely prevent undesired behavior, but if the incoming and outgoing pipelines are running on different nodes in a geographical region, then the only way to prevent the situation in Figure 4.8 is for nodes to share their FIB and PIT entries and keep in tight coordination. Therefore, a second inter-region communication protocol may be needed if this behavior is a big concern. That

said, as I will show in the next the experimental section of this chapter, simply allowing DTN messages to flood within a region helps address this concern, and in Chapter 7 I will show that having caching within the network can also addresses this issue even if all nodes in the region are unaware of each other.

In this chapter, I have mostly chosen to have faces represent geographical regions which work together. However, the generic structure of medium level integration with different face design choices can follow the same techniques described here. My choice of using geo-regions is based on the suitability of this implementation for the scenarios of interest.

4.2 Experimental Setup and Evaluation

In this section I evaluate two types of face layers (geo-region and node faces) in a number of different networks to highlight the strengths of this adaptive approach for a disaster scenario. I used three DTN protocols for the "next hop": Epidemic (Ep), Epidemic with a hop-count limit of 4 (Ep-4), and Binary Spray and Wait with 8 copies (SaW-8).

Emergency Center Scenario. Given one of the primary motivations is disaster relief, I created a scenario where emergency centers were placed at intervals into the world and advertised data about their center when they come online (e.g. "/data1" would be the name for the data about Emergency Center 1). All other nodes in the network (assumed to be people) were interested in getting regular updates about these centers to their phones, but emergency center information was only fresh for a period of time, so nodes needed to request fresh data when it expired (e.g. if Person 1 was aware of Emergency Center 1, but did not have fresh information for that center, they would generate an NDN interest message in "/data1"). Expired data/interests were requested with a Poisson distribution (mean of 60 seconds) after expiration.

Measuring Performance. As defined in Section 3.5, I will focus on Interest Satisfaction Ratio (ISR) and Interest Satisfaction Time (IST), as well as overhead here. I measure overhead "end to end" looking at how many transmissions were generated by an NDN

event (interest generation, or data generation) which includes all transmissions along all leaps (e.g. transmissions for IN_1_1_1 and IN_1_2_1 are counted together as being for IN_1). The ONE DTN simulator [71] is used for all experiments in this chapter.

I simulated 20-25 minutes of real time depending on the scenario, the emergency centers came online every two minutes (i.e. Center 1 came on at 2 minutes into the simulation, Center 2 at 4 minutes, etc.), data was fresh for 2.5 minutes after it was generated and interests had a lifetime of 5 minutes. Nodes generated interests until the 15 minute mark (to allow time for the interests to be satisfied before the simulation ended). The simulations were run 10 times each, with different node positions and emergency center positions. Nodes were given a Content Store (CS) large enough to cache the data from all centers, but data was removed from the CS when it was no longer fresh.

4.2.1 Geo-region Face Experiments

Baseline Experiments. To demonstrate the geo-face functioning in a non-DTN environment with working infrastructure, I created a network of 125 nodes randomly positioned in a 400m x 400m space, divided the world into a grid of 16 areas (4x4) and chose five random nodes as emergency centers. Named data was advertised via a flooded "announcement" messages when an emergency center came online. Upon receiving an announcement, nodes updated their FIBs and re-transmitted the message with updated information such that FIB entries were only for neighboring regions as discussed earlier. I created two baseline experiments. First, all nodes remained stationary and were given a communication range of 80m to ensure the network was connected. Next, each non-data node performed a random walk through the world (they choose a direction, wandered between 0 and 50m, and choose a new direction). Data producing nodes remained stationary.

DTN Experiment. To simulate the infrastructure going down and the network reforming as a DTN, I cut the communication range in half (40m instead of 80m) to create simple DTN environment. The data freshness and interest lifetimes were double the baseline experiments (5 and 10 minutes respectively) to account for the DTN environment. I allowed the same

period for interest generation (15 minutes) so the total simulation time was increased to 25 minutes to accommodate the cooldown period. People nodes performed random walk and emergency centers remained stationary.

DTN with Diffusion Experiment. The announcement messages used to setup the FIB has limitations in a DTN which can greatly affect the network's performance. If the TTL for the announcement is set too short or if a portion of the network remains disconnected for longer than the TTL of the message, some nodes will never discover certain emergency centers. This can be improved by using a diffusion mechanism which shares between regions rather than within regions. To examine this, I switched the network to a diffusion mechanism which exchanged tables whenever nodes came into contact. Nodes meeting within a region swapped table entries and nodes meeting at the border between two regions update their tables to direct messages to the neighboring region. More advanced data advertisement protocols will be discussed in Chapters 5 and 6.

Cache Experiment. NDN provides in-network caching which the medium integration is making use of. To get a sense of how the network is being benefitted, I re-ran the above diffusion experiments with a CS size of 0.

Increased Network Size Experiment. An emergency center may need to provide information to hundreds or thousands of users who may not be near the center, so scaling to a larger network is a major concern. To explore this I ran the DTN random movement experiments (40m range) w/ diffusion and the cache on in a 800m x 800m world with 64 areas (8x8) and 500 nodes. Much larger networks with over a million nodes will be discussed in Chapter 7.

Increased Data Producers Experiment. Depending on the scenario, five emergency centers might be insufficient, so I explored how the network fared as the number of emergency centers increased. I first "turned on" all emergency centers at the beginning of the simulation and then increased the number of emergency centers from 5 to 10, 15, and 20. Larger numbers of announcements are discussed in Chapters 5 and 6.

Human Motion Model Experiment. Humans do not just randomly walk around the world, as noted in Chapter 2, so SWIM-Prox was used for these experiments (see Section

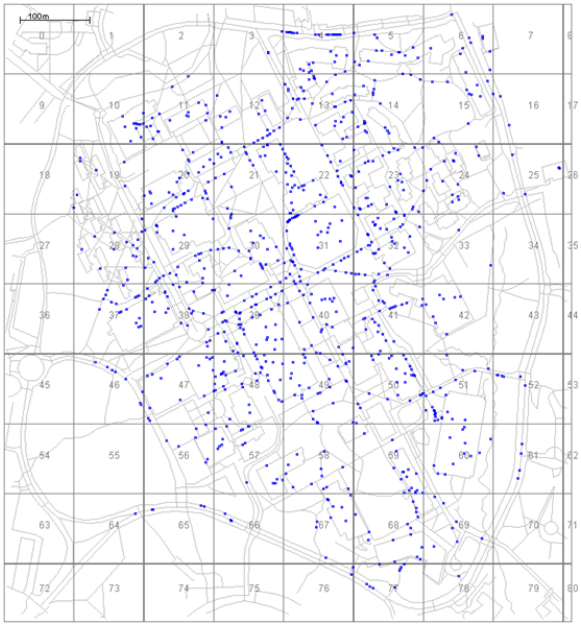


Figure 4.9: GMU campus with grid overlay in ONE.

3.4). I simulated 1000 people in addition to the two busses. The communication range for people was reduced to be closer to a PSN (25m). The grid-size remains the same from the other experiments (100m x 100m), making 80 distinct faces (8x10)[1]. Five emergency centers were randomly placed in buildings on campus (different for each run). A screenshot of the GMU campus in ONE is shown in Figure 4.9.

4.2.2 Geo-Region Results and Discussion

Baseline vs. DTN Environments. As can be seen in Figure 4.10(a)-(c), all three DTN strategies performed well in the baseline and DTN experiments, and the change from connected network to a DTN had the expected outcome (lower ISR, higher IST, and fewer interests generated). In the DTN the average time for nodes before finding their first emergency center was approximately 2.2 minutes after the first emergency center goes online,

[1]There is no reason to use a square grid-shape for geo-regions, but the grid is easy to keep consistent between the baseline and GMU experiments. The network performed similarly when geo-regions were a grid for outdoor spaces and each building on campus was its own geo-region.

and the average number of centers nodes became aware of was 3.2 centers. In the DTN experiments, the difference in ISR between Ep and Ep-4 was not statistically significant, but the difference between Ep/Ep-4 and SaW was. This slight difference occurs because the "spray" component may happen to send the message in the wrong direction. The slight difference between satisfaction times were not statistically significant.

Benefits of Using Diffusion for Named Data Advertisement. Figure 4.10 also shows that using diffusion increased the average number of emergency centers known about by the end of the simulation to 4.5 but also increases the average time until the first route is discovered (approximately 6.1 minutes after the first center goes online) because tables are only exchanged when a new connection is established, so connected sections of the network do not spread their routing tables as quickly. However, diffusion increases the ISR of all three DTN algorithms above a 95% and significantly reduces the IST.

In addition to the information in Figure 4.10 which summarizes the ISR, IST and overhead raw values, I show in Figure 4.11 the results of dividing the nodes into quintiles by their individual ISTs and averaging across all runs. This gives a slightly better view of what is happening at the node level. Figure 4.11 shows ratios for (a) IST and (b) IST variance, and compares (1) announcements : diffusion, (2) cache off : cache on, (3) 16 regions and 125 nodes : 64 regions and 500 nodes, and (4) slow startup emergency centers : fast startup. Results are shown per IST quintile. Higher values indicate second value produces less time/variance. Note that time is reported in seconds.

I focus here on Ep-4, but Ep and SaW showed similar trends. Looking at the first column in Figure 4.10 we can see that diffusion helped nodes in the top 20% of the network the most (in both the IST and IST variance) indicating that nodes near the data centers were benefited more.

The Benefits of NDN Caching. My experiments demonstrate some interesting benefits of the NDN cache, and examining the second column in Figure 4.11 (a)-(b) (showing the effects of turning *on* the cache) we see that the cache improves the IST for all quintiles and greatly reduces the variance in IST for the upper 40% of the network. We also see that

Interest Satisfaction Ratio	Ep	Ep-4	SaW-8
Baseline and DTN (125 Nodes)			
Stationary	99.7%	99.5%	99.6%
Random Walk	99.9%	99.9%	99.6%
DTN Random Walk	93.2%	91.9%	81.0%
Diffusion Variations			
Diffusion	99.0%	96.8%	94.6%
Diff. Cache Off	97.4%	96.2%	94.9%
Diff. 500 Nodes	n/a	96.8%	92.8%
Diff. 5 Starting Centers	99.2%	98.2%	95.5%
Diff. 20 Starting Centers	89.1%	96.8%	96.9%
SWIM-Prox Movement (1000 Nodes)			
Announcements	n/a	90.6%	96.6%
Diffusion	n/a	82.4%	99.1%

(a)

Interest Satisfaction Time (Sec)	Ep	Ep-4	SaW-8
Baseline and DTN (125 Nodes)			
Stationary	1.2	0.8	2.2
Random Walk	4.3	2.1	4.2
DTN Random Walk	80.2	84.2	109.1
Diffusion Variations			
Diffusion	54.8	50.8	44.1
Diff. Cache Off	84.0	76.9	93.1
Diff. 500 Nodes	n/a	62.6	84.9
Diff. 5 Starting Centers	65.3	68.2	77.0
Diff. 20 Starting Centers	119.9	90.6	79.0
SWIM-Prox Movement (1000 Nodes)			
Announcements	n/a	107.0	61.4
Diffusion	n/a	112.8	54.8

(b)

Transmissions Per Interest/Data	Ep	Ep-4	SaW-8
Baseline and DTN (125 Nodes)			
Stationary	183.5	107.1	24.9
Random Walk	212.0	158.8	44.1
DTN Random Walk	222.6	82.8	31.8
Diffusion Variations			
Diffusion	259.0	94.8	38.1
Diff. Cache Off	315.9	130.3	55.6
Diff. 500 Nodes	n/a	120.1	41.5
Diff. 5 Starting Centers	240.1	91.4	36.3
Diff. 20 Starting Centers	168.3	87.1	36.1
SWIM-Prox Movement (1000 Nodes)			
Announcements	n/a	639.0	281.2
Diffusion	n/a	577.1	313.3

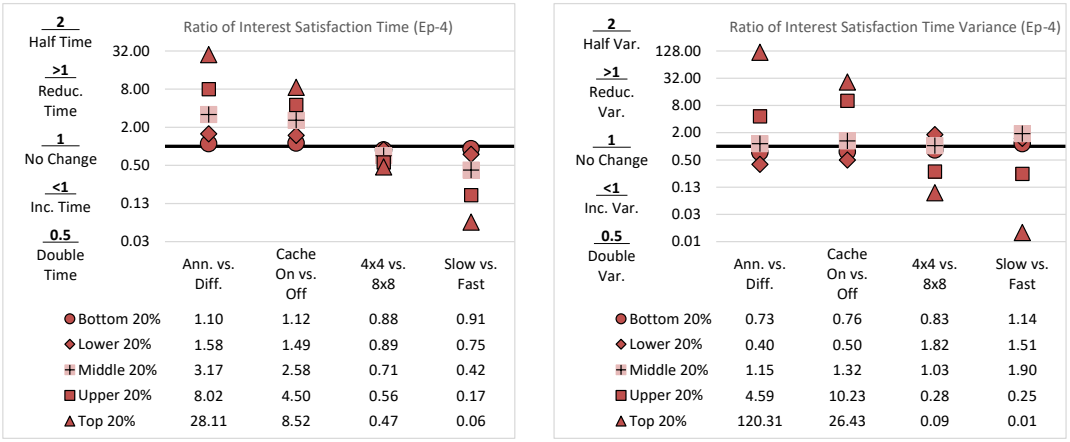
(c)

Figure 4.10: Geo-region architecture expirimental results.

cache increases the variance for the lower 40%, which makes sense since nodes with the lowest IST in any experiment were likely the furthest from the emergency centers and the cache would give them occasional fast responses.

Minimal Effects of Increased World Size. As can be seen by the results, the geo-region faces scale well regardless of the number of nodes or size of the network. Examining the third column in Figure 4.11 (a)-(b) we see that quadrupling the world size and number of nodes did not quadruple or even double the IST for the lower 80% of the network (only the top 20% were double).

Impact of Increased Number of Data Producers. At a glance the raw results in Figure 4.10 for "fast startup" of all emergency centers looks bad for the network but looking into the data closer, it turns out that number of interest messages and overhead in Ep is approaching a point where the network is slightly congested which is why Ep shows a sudden decrease in

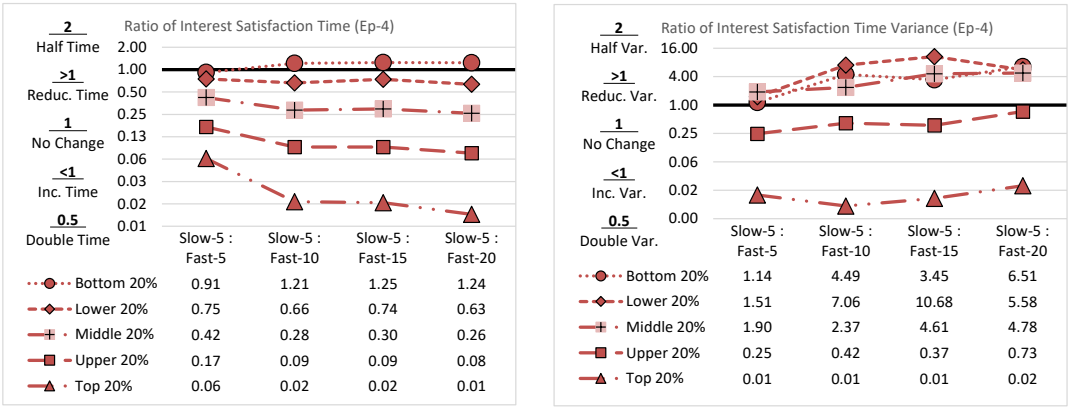


(a) (b)

Figure 4.11: Ratios for IST and IST variance for a number of different scenarios.

ISR and Ep-4 and SaW do not. Examining the fourth column of Figure 4.11 (a)-(b) we see that the top performers (nodes which happened to be near the first center to open) were most impacted as they needed to contact further centers earlier in the simulation. Happily for the network, the initial increase does not continue after the first 10 emergency centers are introduced, Figure 4.12 show the effect of switching from a slow startup of 5 centers to a fast startup of 10, 15, and 20 centers. Data in this figure is shown by IST quintile and flat lines indicate changes which have an equal impact. The nearly horizontal line indicates that after the reduction in IST from 5 to 10 centers, thereafter there is almost no change in adding more centers. Additionally, the IST variance is substantially reduced for the bottom 60% of the network the more emergency centers we add.

Positive Impact of Human Motion Models. There are two interesting differences in the results for this environment: the number of transmissions per data/interest message is higher and SaW much performs better. The number of overall transmissions is explained by several factors. First, the environment is no longer square, so some messages may need to travel further naturally. Second, adding more nodes to a given area increases the number of transmissions within the area (as noted earlier, the hop-limits and SaW copies do not affect in-region transmissions as these are considered "direct deliveries"). And third, the



(a) (b)

Figure 4.12: Comparison of "slow startup" of emergency centers and "fast startup".

communication range being reduced means more transmissions may be used to travel the same distance. The improvements in SaW are easy to explain when watching the simulation run: in the random walk model it is just as likely for a node to move toward any of the eight adjacent regions, but in the human model there are often fewer adjacent regions humans will travel in; so with the communication range reduced, the human movement is dictating both the best direction to forward messages and limiting the directions the "spray" is likely to take, creating better performance.

Interestingly, the dense clustering of nodes along paths and in buildings means that the time before the average node gets its first route has been reduced to approximately 1.9 minutes with announcements and 2.3 minutes with diffusion (meaning most nodes are aware of the first emergency center before the second goes online). By the end of the simulation most nodes were aware of around 4.8 emergency centers without diffusion and 4.9 with diffusion (much higher than the random walk).

4.2.3 Comparison to High Integration

One of the benefits of the "medium" integration does *not* just simply supplement NDN with the ability to "store-and-forward/wait", as is often done in high integration levels discussed

Interest Satisfaction Ratio	High	Medium
Stationary		
Baseline	100.0%	100.0%
Connection Failure	100.0%	100.0%
Random Walk		
Baseline (80m)	41.2%	77.9%
DTN (40m)	20.9%	43.4%
Diffusion, Cache	48.5%	69.9%
Diffusion, No Cache	32.9%	56.7%

(a)

Interest Satisfaction Time (Sec)	High	Medium
Stationary		
Baseline	0.2	0.4
Connection Failure	1.4	0.9
Random Walk		
Baseline (80m)	24.7	16.3
DTN (40m)	56.0	44.8
Diffusion, Cache	34.6	23.0
Diffusion, No Cache	47.3	46.9

(b)

Transmissions Per Interest/Data	High	Medium
Stationary		
Baseline	4.7	11.6
Connection Failure	5.0	13.7
Random Walk		
Baseline (80m)	8.9	13.9
DTN (40m)	6.4	12.2
Diffusion, Cache	1.1	7.2
Diffusion, No Cache	1.5	12.2

(c)

Figure 4.13: Node-face architecture experimental results.

in Section 2.3. To highlight this difference, I swapped the geo-region layer for a node face layer (FIB entries in the form of /data0 → 1 now mean "forward to node 1" rather than to "region 1"). I used the *identical* SaW 8-copy DTN layer and the *identical* NDN layer from my earlier experiments. This is one of the key features of this architecture: a single layer swap can produce completely different ICDTNs.

For a high integration level, I swapped the DTN layer and face layer with a single "filtering face layer" which only acts to control where outgoing messages are sent. This gives NDN a simple store-and-carry interface, but without the DTN layer, forwarding /data0 to 1 means to send to node 1 directly and wait if it isn't available. Now, as with all high integration levels, every hop must perform an NDN decision.

Non-Mobile DTN Experiment. As noted earlier, the random walk motion model is not a good fit for a node face layer as the tables quickly become out of sync without some secondary routing protocol. Since this face is a good design choice when you can expect to see the same set of nodes over and over, I designed a more appropriate scenario for this section with stationary nodes. As discussed in Section 3.4, I created a "broken" WiFi

interface which has only a 10% chance to connect each decisecond it was in range of a node and a 90% chance each decisecond it was not transmitting to disconnect again. This created randomly fluctuating connectivity in the network and caused a noticeable delay in message delivery while each node was still limited to connections with only a portion of the network.

Mobile DTN Experiment. To show just how important designing for a given motion model can be, I used the random motion from earlier with restricted range (but a working WiFi interface).

The Benefits of Medium Integration. Examining Figure 4.13 we see that the baseline experiments with both the high and medium integration perform almost perfectly when the network is connected and functioning. Adding in connection failures, however, shows off the advantages of the separate DTN layer as it can work around the failures and find another path, speeding up the interest satisfaction time (with statistical significance). Additionally, when we get significantly lower performance with a poor match between the face and motion model choice (much lower than the 80-100% satisfaction we saw with the geo-region faces), the medium level of integration is able to continue functioning as the DTN layer "reaches out" into the network after nodes have moved out of range.

4.3 Conclusion

In this chapter I have shown the viability of a medium level of integration between an ICN and a DTN using NDN and two standard DTN algorithms using a virtual face layer. I have explored the design choices which must go into the creation of this middle layer which provides the "next leap" to a DTN and I have shown a particular configuration amenable to the emergency center scenario. From an architectural standpoint, the ability to swap the virtual face layer and DTN layers independently of each other and independently of NDN, enables an amazingly diverse and powerful set of tools not available to other types of ICDTNs.

I have shown that, compared to high-integration proposals, this approach avoids the

need to heavily modify either the ICN or DTN protocols. Additionally, I have shown substantial benefit of this approach over simply providing store-and-forward capabilities to NDN. In comparison with the low-integration approaches, I have demonstrated that this approach provides the benefit of in-network caching throughout the network and that NDN can provide the contextual awareness needed to route messages efficiently in a disaster scenario.

Chapter 5: A Data Advertisement Protocol Using Network Coding

Efficiently announcing the availability of new data (data advertisement) is a foundational requirement for the successful deployment of the new generation of information centric networks (ICNs). As discussed in Section 2.4, this problem is solved in ICNs using techniques similar to traditional IP networks, but the new interest in infrastructureless ICNs in DTN environments means that these traditional techniques no longer apply.

In the previous chapter I showed that large scale, ad hoc ICDTNs can operate in challenged environments, however one of the interesting discoveries was the effect of the data advertisement scheme on the network. Table sharing ("diffusion") performed better than flooding, but this isn't scalable when the tables are large, which they are inclined to be if there are many "faces" or if there is a large set of named data in the network.

This chapter, and the following chapter, consider the problem of efficient data advertisement in ICDTNs. This chapter discusses the practical problems of designing an advertisement protocol and presents a network coding based solution that operates in a fully disrupted environment and outperforms simple flooding mechanisms. It is inspired by the diffusion mechanism from Chapter 4 and introduces network coding as a potential solution. Chapter 6 examines the problem from a more theoretical standpoint, discussing how such protocols can be compared and contrasted, and then analyzes the flooding approach, this chapter's network coding protocol, and an additional network coding "inspired" solution.

In this chapter I utilize *network coding* to improve the process of data advertising, analyzing specifically how to use random linear network coding (RLNC) in this context (Section 5.1), and evaluating the resulting performance improvements under a number of disrupted environments (Section 5.2). My results show that, under a number of scenarios,

my network coding based solution substantially improves the performance of ICDTN data advertisement. These sections have been published in [27] and presented in [26]. Finally, a deep-dive is given on the density parameters for RLNC and a heuristic setting is examined (Section 5.3).

5.1 A RLNC Protocol for ICDTN Data Advertisements

The procedure for network coding flooded messages from a single source is well established (see Section 2.5), but it's application to ICDTN data advertisements is not quite as straight forward. Network coding from multiple sources, rather than single source, as discussed, is an area not well studied. Additionally, there are practical implementation details with NDN and the architecture from Chapter 4. Finally, RLNC has several parameters and configuration settings which need to be examined in more detail than the preliminary discussion in Section 3.3.

In order to get a concrete solution in place, this section addresses the practical implementation elements for the RLNC solution, and addresses some basic multi-source network coding issues, including (1) what is the exact structure of a data advertisement? (2) how and when can advertisements be mixed together to form an encoded message? and (3) what are the effects of the DTN environment on the network coding protocols and the effects of delayed decoding on the ICN protocols?

The data advertisement structure for this work has been discussed in Section 3.2. The RLNC parameters appropriate to solve the "when to encode" problem are both complex and interesting enough to deserve a separate discussion in Section 5.3 with its own proposed heuristic and experimental evaluation. This heuristic is a contribution to the general field of multi-source network coding. Therefore, the next section details how to mix (encode) data announcements and the effects of the DTN environment.

5.1.1 Augmentations to NDN for Announcement Arrival

When data announcements arrive, the FIB tables are constructed using several pieces of information for each name and/or prefix advertised: (1) the name of the data – required and provided by the announcement itself –, (2) the "face" where interests should be sent – at least one provided by the face layer as the message arrives –, (3) any known statistics regarding the efficiency of the face for the given name – used to choose between faces when forwarding and provided by the hop count, leap count, or arrival time – , and (4) timeouts for removal of the FIB entry – if needed and provided by the announcement itself.

When using *encoded* announcements, these FIB entries must be constructed only after decoding, so we must store a set of tuples containing each observed incoming face and associated cost for each symbol in the decoding matrix. Additionally, encoded messages need to include vectors for *any* meta-data that needs to be updated by intermediate nodes. For example, the cost (leap count) must be an unencoded vector so that intermediate nodes update these fields freely without decoding first.

This means that the standard decoding matrix for RLNC needs to be augmented so that the incoming face information is retained for later use in the FIB. Traditionally RLNC doesn't care where the data arrived from, it only cares about decoding the payload. Figure 5.1 shows the required flow, and changes to the decoding matrix, needed to ensure that incoming encoded messages can create FIB entries once decoded. Two coded messages x_1 and x_2 arrive for g_1 and are placed into the decoding matrix, but the interface they arrived on, and the additional unencoded cost vectors are saved in an augmented portion of the decoding matrix. When messages can be decoded, the payload from the decoding matrix and the augmenting fields are used to construct the FIB entries.

As discussed in Section 2.4, it is not vital that paths be "optimized" to represent the shortest path at the time of announcement delivery, because (1) the path during the announcement period may not represent the optimal path at some later time, (2) multiple paths might be tried at once, and (3) NDN can use PIT timeouts to provide "feedback" on potential paths for cost updates. That said, at a minimum, at least one (face, cost) tuple

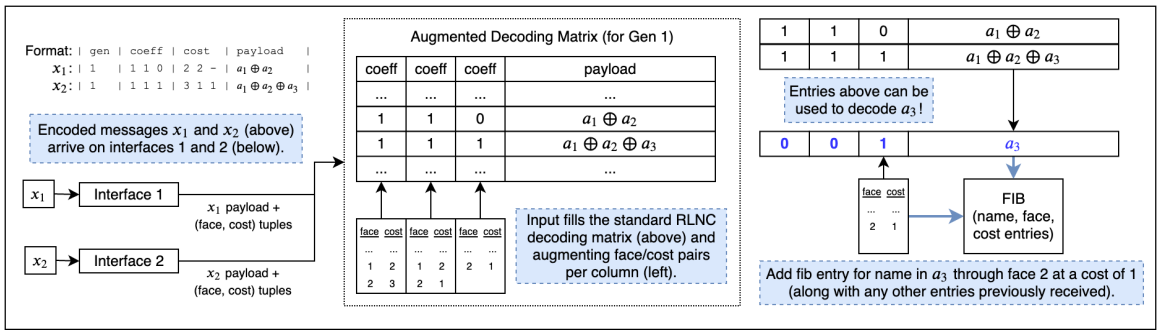


Figure 5.1: The flow for incoming RLNC advertisements upon arrival.

should be stored for each symbol (advertisement) indicating the lowest cost discovered or the first time the symbol was seen. The first time, in this case, being a heuristic guess for the shortest path. Space allowing, it is preferable to track all faces and face statistics so that all paths can be utilized later, this will be discussed further in Chapter 6. For a fair comparison with message flooding and to give a sense of the trade-offs when implementing a network-coded announcement system, I will be assuming that we want to track *all* incoming faces.

While the structure above is sufficient for implementation, an additional optimization is possible: senders can add their own "best" (face, cost) tuple for each announcement as an additional vector. This is only possible if (1) the face represents the same network subset for every node (e.g. a geo-region of the world), and (2) the receiving NDN face might also be able to send to that face (e.g. three adjacent regions). This allows NDN to add another potential (face, cost) tuple to the augmented matrix *in case* it might be able to send to that face in the future. Node faces for data mules are also a good use case for this. This augmentation allows the protocol to adopt a level of "diffusion" from the previous chapter where the table sharing is limited to the generation being transmitted at any moment rather than the entire table.

A note must be made here that generation management plays a role in the number of decoding matrices required. Determining which messages (in this case announcements)

can and should be mixed together, as well as forming agreement about the "place" in a generation where a given message belongs, as discussed in Section 3.3, is what allows nodes to construct the appropriate encoded packets with the given generation tag and correct coefficient vectors. One decoding matrix is then needed for each generation "in play" at any given time. In my experiments I will examine forming generations by time epochs, but other divisions are possible as well. This requires one to two decoding matrices per node at most times.

5.1.2 The Structure of Encoded Advertisements

Putting together the discussion from the previous section, the size of encoded packets is roughly: *senderFieldSize* +*genIdFieldSize* + (*numSymbols*metaDataSize*) + *payloadSize*. This is important to quantify as there is a trade-off between the benefits of network coding and sending slightly larger packets.

Encoded packets require a TLV field to identify them as such, sender information tagged by the face layer, and a generation id. The exact size of the sender field will depend on the type of face id tags needed. For example: a node-face might use something like a MAC address for the WiFi card (6-8 bytes); a world-wide geo-region-based face might use the MGRS (which would require 8 bytes for four characters and two 2-byte numbers); and a tiny geo-region network within a single building could use a room number (1 byte). Additionally, as discussed in Section 4.1, some face types benefit from (or require) additional sender information. The size of the generation id depends on the max number of generations assumed to be active at the same time.

The number of symbols is the number of announcements in the generation of x_i and the ordering of the symbols depends on the generation division and generation order agreements (see Sections 2.5 and 3.3). The size of one coefficient in the coefficient vector is the size of the finite field used for encoding (traditionally $\mathbb{F}_8$, so one byte for each).

The hop-count in the cost vector must be sized to store the maximum expected hop count, bearing in mind that these are NDN "leap counts" in this architecture and fairly

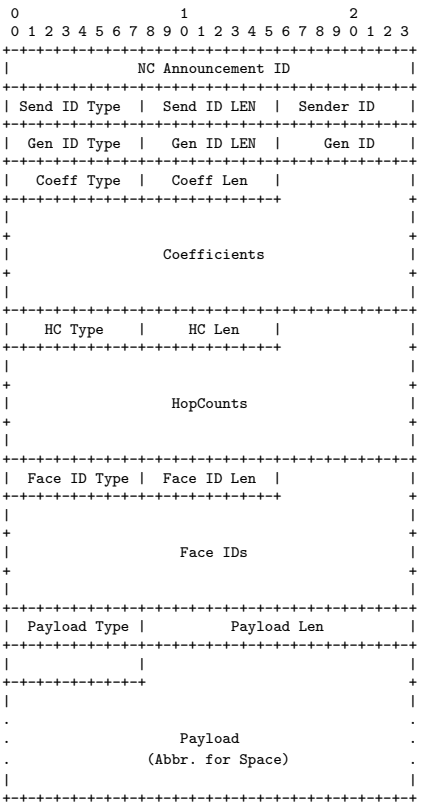


Figure 5.2: Example ad hoc ICDTN announcement packet.

small compared to the DTN level hop counts. The face ids shared from the sender's FIB tables, which allow the diffusion-esque features discussed in the previous section, are sized the same as the sender field. To ensure only a single vector is needed (and not a vector of vectors), only the shortest/best (cost, face) pair discovered so far is transmitted for these two fields.

The size of the payload must be the maximum size of any announcement encoded (assuming 0 padding for shorter messages). Figure 5.2 shows an example encoded packet for /data1 which expires in 10 minutes from receipt and whose public key is in a data packet named /key1, assuming that the generations are of size 10 and coefficients, hop-counts, face ids, and generation ids all require one byte. The remainder of this chapter assumes this structure for encoded packets.

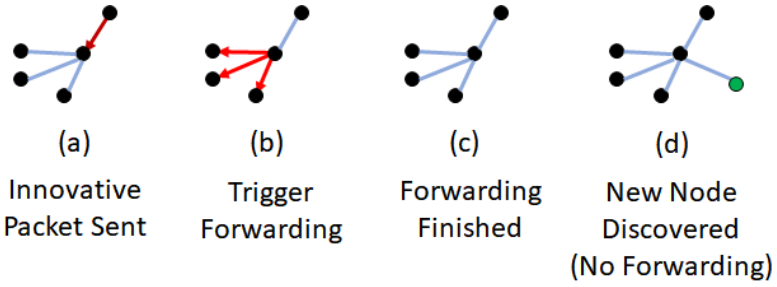


Figure 5.3: DTN scenario for send-on-innovative-packet-received heuristic.

5.1.3 Packet Transmission for Network Coded Packets

Network coding in DTNs, especially mobile DTNs, is slightly different than in stable environments. Importantly, the common "when to send" heuristic — send on receiving an innovative packet — is not a good choice when the network is heavily disrupted. Innovative packets (ones that increases the rank of the decoding matrix) are good indicators of *receiving* new information, but not good of *having* innovative information compared to a newly encountered neighbor. Figure 5.3 shows what happens in a DTN when innovative packets are used as a sending heuristic. (a) Shows an innovative packet being sent to the central node which triggers sending in (b) by the central node to distribute the information (send-on-innovative-packet-received). (c) Shows that no additional packets are being sent, then a new node joins in (d) and still no additional packets are sent.

Some implementations assume that decoding matrix *rank* is sufficient for determining if two nodes have innovative data to share (which is appropriate in many networks), but that is not the case when nodes are receiving information from different sources at different ends of the network (e.g. a rank of 1 does not indicate which symbols have been received, only that a single value has been received). Figure 5.4 shows a simple example where this is not true. Nodes 1 and 2 have data that can be shared to fully decode their matrices, but share a rank of 2.

So a more appropriate action is a handshake that shares information about the current status of a node's symbols. Combining this idea with the desire to pass additional "diffusion"

Node 1 Decoding Matrix

Enc. Vect.			Messages
1	0	1	Mix of m_1 and m_3
0	1	0	Decoded m_2

Node 2 Decoding Matrix

Enc. Vect.			Messages
1	1	0	Mix of m_1 and m_2
0	0	1	Decoded m_3

Figure 5.4: Two nodes with the same rank decoding matrix.

table information, as part of the handshake when connecting, nodes transmit a small bit array indicating which symbols they have for a given generation and which symbols they have decoded. If a node discovers one of its neighbors does not have the same symbols or does not have the same decoded symbols, it tries to send.

Since there is no diffusion equivalent for flooding, nodes always need to send their neighbors any announcements they have not previously sent to the same neighbor, so the number of transmissions is noticeably higher for flooding. Specifically, in a non-mobile network, the number of transmissions for each node is the number of announcements times the expected number of neighbors. Using the symbols from Chapter 6 (see page 90), that makes the total number of transmissions:

$$TotalMsgsFlooding = |N| * |A| * E[|\Upsilon|] \tag{5.1}$$

For network coding, the number of transmissions is approximately $|N| * |A|$. This is not exact because it is possible to send non-innovative packets, and all nodes need to send at least one message to each neighbor for table creation, and so the final calculation is closer to:

$$TotalMsgsNC = |N| * (\max(|A|, E[|\Upsilon|]) + \epsilon) \tag{5.2}$$

For flooding, nodes need to keep one packet per announcement in the buffer while it is valid. If a better route for some piece of data is received, the appropriate announcement is either changed in the buffer (to give it a new in-face id and hop count) or that the old

announcement is dropped from the buffer and a newly constructed one is added. For the network coding, only one encoded packet is needed for each generation, which in turn needs to be updated or reconstructed whenever new information about the generation is received.

5.2 Experimental Setup and Evaluation

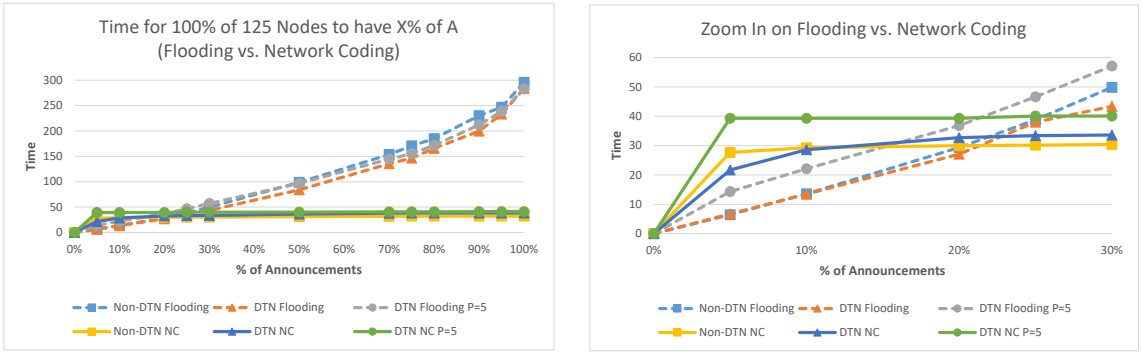
To verify the discussion in the previous sections and demonstrate the performance of the network coding solution to data advertisement, I focus in this section on the following two items: (1) a comparison of network coding with flooding announcements, and (2) the effects of scaling for *mobile* ad-hoc ICDTN networks.

I chose to use simulation since the focus is on very large, potentially mobile networks using ONE and Kodo. Ten runs were performed for all experiments and averages are reported. The number of nodes, data producers, announcements, and transmission ranges vary by experiment.

5.2.1 Comparison with Flooding

I start this comparison by using a 400m x 400m area with 125 static (non-moving) nodes. Nodes are randomly placed in the environment, and I use 80m transmission range to ensure all areas of the network are reachable without motion. To get a baseline, I look at both a "stable" connection, within a given transmission range, and a "flickering" connection, where hosts have a low chance of initiating a connection and a high chance of disconnecting again even when in range. These setting produce a non-DTN and a DTN environment respectively and are discussed in more detail in Chapter 3.

I first examine a network where each node has one piece of data to advertise at the beginning of the simulation. Next, since many of the motivating examples have a limited number of data producers, I also look at the same environment where 5 data producers each have 25 of the 125 announcements (P=5). Figure 5.5 shows the results of these experiments and the enormous benefits of using network coding. Figure 5.5(a) gives the time for 100%



(a) (b)

Figure 5.5: Comparison of flooding and network coding for announcements.

of the network to have heard of X% of the data advertisements in various static (non-mobile) networks, while (b) is a zoom in for under 30% of the data advertisements. The number of transmissions was as predicted by Equations 5.1 and 5.2, and it should be noted that the average decode time for messages (approx. 10 seconds) and the maximum decode time observed during any run (41.5 seconds) were essentially "worth the wait" network-wide when trying to share more than 20-25% of the announcements, but at an individual node level or for networks where nodes only need to acquire some small percentage of the announcements to be functional, the flooding mechanism could be preferable.

Assuming that the networks require data advertisement to happen over a period of time, I examined different production rates. Figure 5.6 shows the time for 100% of the network to receive X% of the advertisements for the same environment along with the min and max time all data might be available in the network. Figure 5.6(a) looks at the environment with a "flickering" connections for 25 data producers generating 5 announcements each, one new announcement every 7-10s. Figure 5.6(b) looks at 5 data producers generating 25 announcements, at the same rate. The results show that the network coding solution is able to keep up with the data production much better than the flooding, on average ending the data announcement period shortly after all data becomes available.

As discussed, we do not actually want to put all announcements in a single generation,

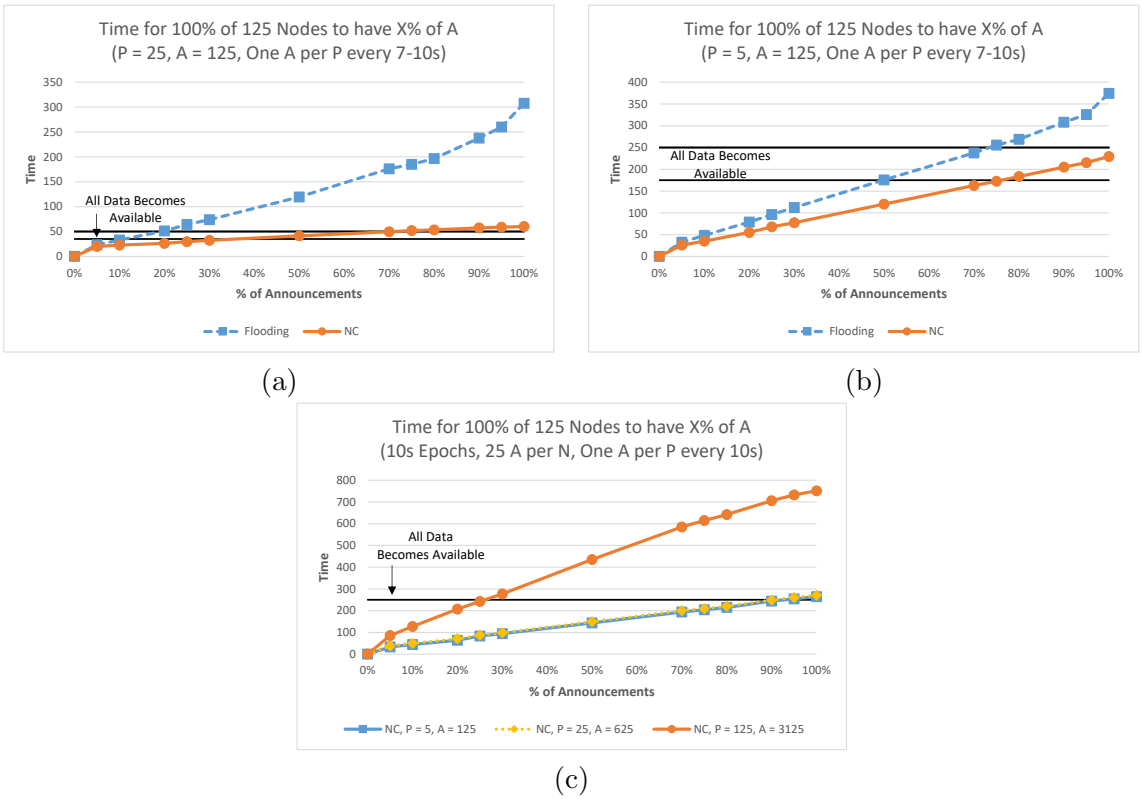


Figure 5.6: Experimental results for epoch-based generation management.

or the size of the encoded announcements will become unreasonable and we will lose much of the benefits of using network coding (the size of encoded packets should to be less than the unencoded advertisement size times $E[|\Upsilon|]$ if we want to transmit less data overall). Additionally, we would like to ensure that this is able to scale for a large number of announcements if necessary. Figure 5.6(c) shows the results of 5, 25, and 125 data producers releasing announcements at 10s intervals and putting them into one of several 10s epochs (such that each epoch has 5, 25, or 125 announcements respectively). As can be seen, 125-625 announcements scales very well, but 3000+ messages in around 4 minutes is noticeably slower as would be expected.

Table 5.1: Expirimental Worlds

Name	Size (#Regions)	Nodes	Trans.	Movement
Small	400 x 400m (16)	125	40m	Rand. Waypoint
Dense	400 x 400m (16)	250	40m	Rand. Waypoint
Big	800 x 800m (64)	500	40m	Rand. Waypoint
GMU	811 x 881m (81)	1007	25m	SWIMProx

5.2.2 Movement and Scaling

As has often been observed, motion models greatly affect the performance of routing protocols, so it is vital to study the performance with motion. For this I created four different sets of experimental "worlds" which are detailed in Table 5.1.

As discussed in Sections 2.1 and 3.4, for random waypoint nodes travel to a randomly chosen location and wait for a short period of time before choosing a new location, and SWIM-Prox is a human inspired motion model where nodes choose a building in the world, travel to it using roads, wander around the building for a period of time, then choose a new building. Five stationary data producers are used in all experiments.

In order to facilitate motion, I use the geo-region face discussed in Chapter 4 where nodes route to neighboring regions instead of neighboring nodes. This means that each node will create only 3, 5, or 8 tuples for each announcement depending on if the region is a corner, edge, or central region. The number of transmissions for flooding is still $|A|*|N|*E[|\Upsilon|]$, but $E[|\Upsilon|]$ is no longer the average number of neighbors but the average number of neighboring *regions* (between 3 and 5.25 since the diagonal is much less likely to be crossed). Testing on a non-mobile network with the geo-region faces shows that the number of transmissions is consistent with this. The $E[|\Upsilon|]$ in the previous experiments was around 12.8, so one would expect a lot fewer transmissions for flooding and therefore a smaller, but still perceivable, benefit from network coding (this geo-routing inspired technique is essentially "smarter" than pure flooding).

Since with geo-region routing, FIB tables need to be updated whenever nodes cross

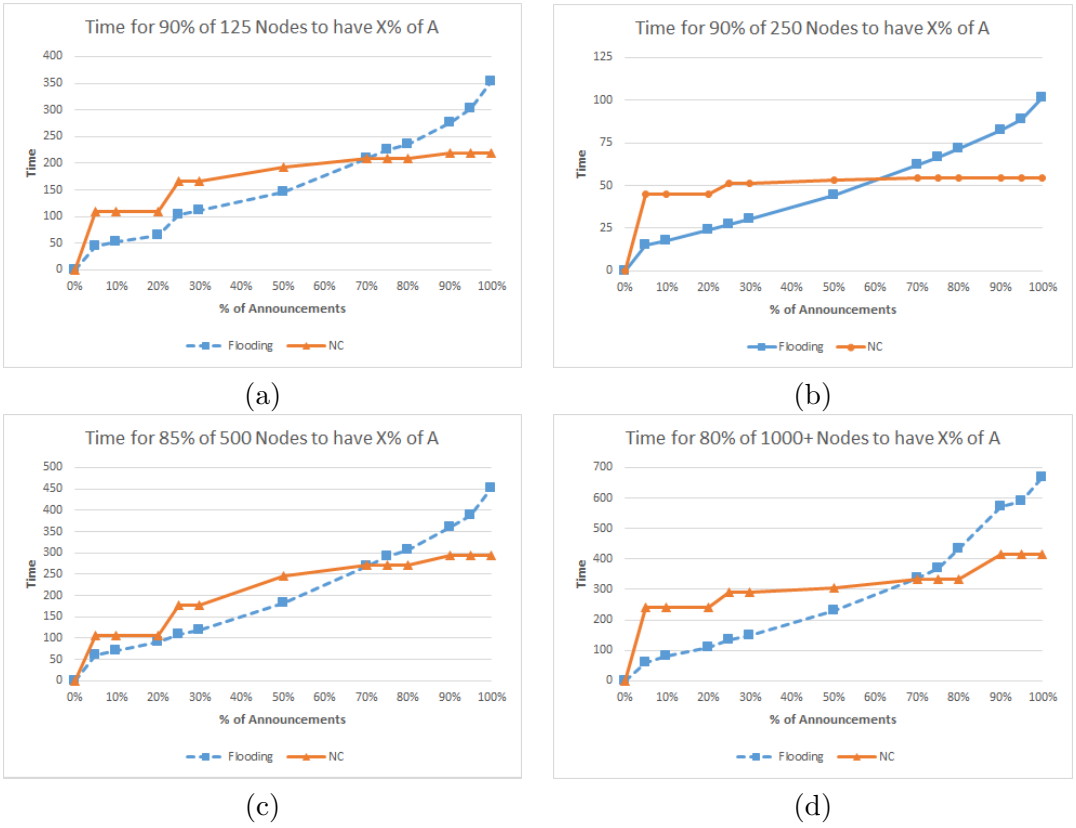


Figure 5.7: Expirmental results for geo-region architecture with movement.

the region boundary, nodes traveling between regions simply drop their FIB tables and, in the case of network coding, drop their decoding matrix for each generation and all associated information. Nodes then have to reconstruct these tables using the chosen method (announcement flooding, network coded announcements, etc.).

As discussed in Section 3.4, adding motion means that it is no longer practical to discuss the time for 100% of the network to acquire some percentage of the data, since (1) nodes could theoretically spend the entire simulation disconnected from the network, and (2) nodes will no longer have their routes if sampled just after a region change. Instead I examine the time it takes "Y% of the network to have X% of the data" in Figure 5.7 for nodes are moving in (a) the small world, (b) the dense world, (c) the big world, and (d) the GMU world. Y is chosen based on observed number of nodes resetting their tables at any

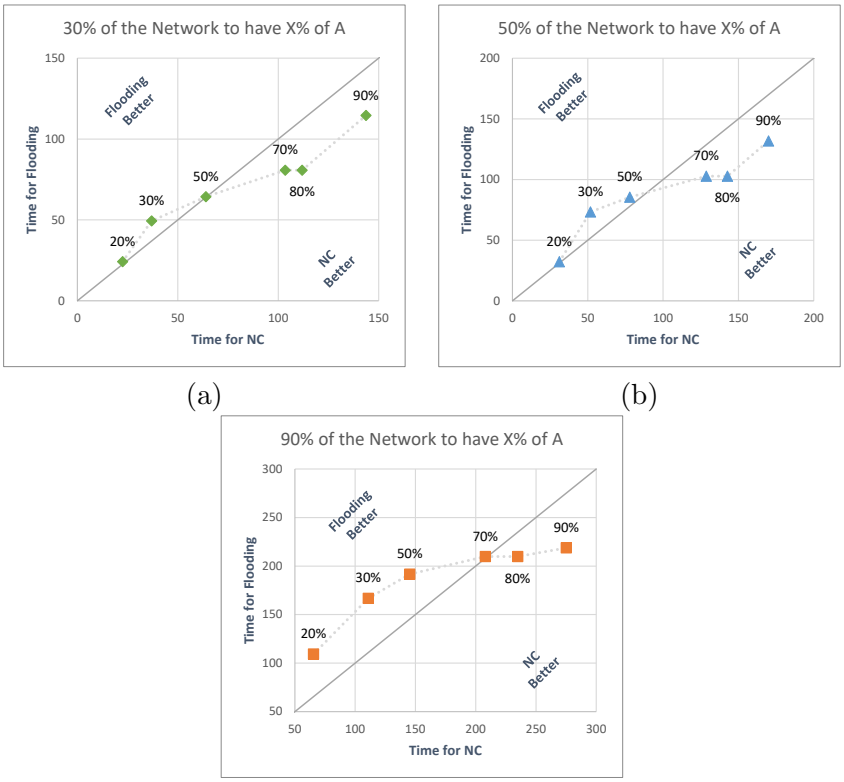


Figure 5.8: Average time for 30/50/90% of the small network to recieve X% of the data.

given time. The results are fairly clear: the crossover still happens where network coding is preferable to flooding, but the particular point of the crossover is later for this network due to the change in face type discussed earlier. It should be noted that all four networks using this face type perform very similarly, which is exactly as expected with the analysis. The max decode time is much larger when motion is involved, which makes sense since the network itself is prone to regular partitions and individual nodes may receive an encoded packet then wander away for some substantial amount of time.

We can quantify when network coding benefits the network in relation to the particular percentage of the network we desire to alert. Figure 5.8 captures these trade offs, showing the time comparison for 30%, 50%, and 90% of the small network receiving some percentage of the announcements. Figure 5.8(c) shows that for 90% of the network to receive the

announcements, flooding is out performing network coding until trying to collect over 70% of the advertisements. However, as seen in 5.8(a), for only 30% of the network the crossover is happening at around 50% of the advertisements and for less than 50% flooding is not out performing network coding as much.

5.3 RLNC Parameters for Data Advertisements

In the previous section I examined a vanilla implementation of RLNC for data announcements, and it can be observed in the experiments, especially Figure 5.5(b), that network coding produces a "delayed" effect, especially when the number of data producers is small, indicating that the producers are encoding many messages together which can't be decoded until enough messages are received. Chapter 6 analyses the time to decode messages more formally for this vanilla implementation, but to get a sense of how this can be improved with only a little local knowledge, in this section I explore setting the encoding "density" (see Section 3.3) and determine a heuristic for this particular parameter.

As discussed in Section 3.3, the two controllable node-local parameters for network coding are choosing the coded announcement contents, which requires a lot of information to utilize properly, and adjusting the density parameter, which can be understood as controlling the number of symbols a node chooses to mix into a message. In static networks, manually choosing the mix of messages is used in "slow start" where nodes send their first packet unencoded and each subsequent messages mixes in a single additional packet. This adds redundancy to the system while also speeding up decoding by increasing the chance of "partial decoding". Partial decoding happens when one symbol out of a mix can be decoded alone, such as if one sends a mix of messages a_1, a_2, a_3, then sends a mix of only a_1 and a_2. The receiving node can decode a_3 from the mix but not a_1 and a_2.

However, for ad hoc networks it is much less easy to plan the appropriate mixes for optimal delivery. The issues with slow starting, for example, are twofold. First, since nodes have no idea what other nodes are sending, and have a high chance of sending information that is redundant instead of sending new information each time. For example, imagine that

n_k is also sending to n_j and also using a slow start; both n_k and n_i will be sending the exact same messages to n_j. Second, if a node is "tuning" what it sends to specifically address the issues of one neighbor, the messages may be less useful to other nodes which "overhear" the message (see Section 2.5).

Choosing a density rather than choosing specific packets means that nodes are still making use of the "random" component of RLNC, and are not falling into the trap of presupposing what the rest of the network is doing. Setting the density lower allows nodes to choose to send fewer items which increases the "off by one" chance. Setting the density higher increases the chance that individual nodes experience the situation shown in Figure 3.2. That said, a better way of looking at this is that setting the density low increases the chance of decoding something *now*, but deceases the chance that any given announcement is sent (just like reducing the generation size), while increasing the density improves the chance of decoding *later*, because it increases the number of previous encoded messages containing each announcement. This is why later, in Chapter 6, the density parameter gets "canceled out" from the decode time in Equation 6.13: the two factors act as mirrors if no partial decoding occurs.

5.3.1 Partial Decoding and the Decode Time

In this section I will discuss how we can use the density parameter in somewhat the same way as slow starting to increase the change of partial decoding. To start with, it is important to understand that there are two ways to be off-by-one, first is for the receiving node to have all the symbols of the sending node unencoded, except 1, and second is for the receiver to have (or be able to create) a mix encoded messages that contains exactly one symbol less than any new messages sent. I will examine here the probability of sending a message that has exactly one more symbol than the receiver is aware of, and assume that the receiver either has these symbols decoded or can assemble the appropriate mix of symbols (this is a significant assumption, but it makes the math more tractable).

The probability of sending exactly one additional symbol than the receiver has is simply:

$$P(oneNewSymbol) = \frac{\binom{c}{e-1}\binom{x}{1}}{\binom{c+x}{e}}, \text{if } c \geq e-1, \text{or 0 otherwise} \tag{5.3}$$

Where c is the number of symbols both nodes have in common, x is the number of symbols that needs to be transferred, and e is the number of symbols the node decides to encode (*density*, or D, times *genSize*. Simplifying some you get the following when e is greater than 1:

$$P(oneNewSymbol) = \frac{x \cdot e \cdot c! \cdot (c+x-e)!}{(c-e+1)! \cdot (c+x)!} = \frac{x \cdot e \cdot \prod_{i=c-(e-2)}^{c} i}{\prod_{i=c+x-(e-1)}^{c+x} i} \tag{5.4}$$

In short, e plays a direct role in increasing the probability, since it appears in the numerator and the last $(e-1)$ terms of the $(c+x)!$ are kept in the numerator as well, but on the other hand, e plays a role in decreasing the probability since the last $(e-2)$ terms of the $c!$ are kept in the denominator. This makes for a quite complicated relationship between e and the probability of partial decoding.

Luckily, there are some simple things we can discover with this formulation. First, if nodes have very little in common (a low c) setting e above $c+1$ completely prevents any chance of partial decoding. In this case, encoding fewer symbols (keeping a low density) is preferable to permit partial decoding. Second, when there is sufficient c in relation to e, the summation in the denominator will *always* outweigh the summation in the numerator so increasing x *or* e will be a balancing act between the increased x or e terms in the numerator and the increased pull of the $(e-1)$ terms in the denominator. That implies that keeping e low when x is high is a good heuristic since this will keep the minimum terms in the denominator by raising $c+x-e+1$ and increase the number of terms in the numerator by decreasing $c-e+2$. Figure 5.9 visually captures this trade off showing the

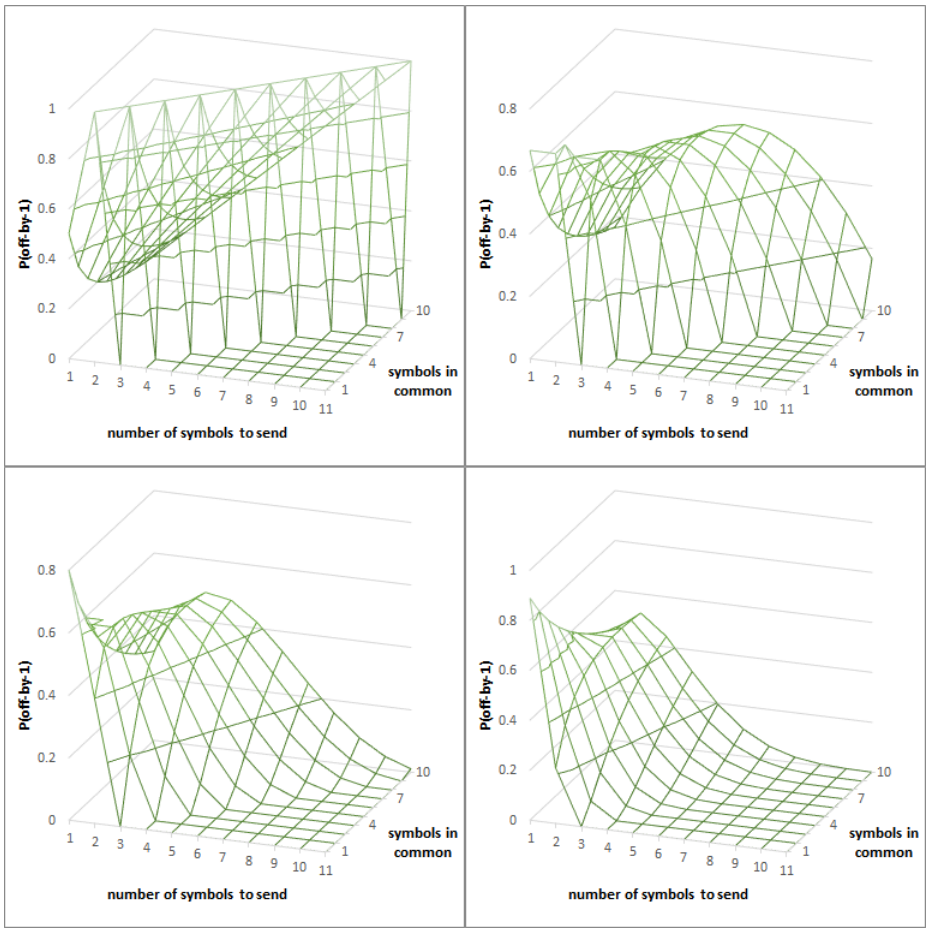


Figure 5.9: Probability an immediate partial decoding for 1, 2 (top), 4, 8 (bottom) symbols.

probability an immediate partial decoding when trying to transfer 1, 2, 4, or 8 symbols (x) in relation to the number of symbols in common (c) and the number of symbols to send (e). The "dip" when encoding very few symbols with a lot of shared information can be intuitively understood as occurring because there is an increased chance of sending only information shared between the two nodes (and not actually sending any new symbols).

The ideal value for e is also a changing game as two nodes communicate, since each message could mean the nodes have more in common than they did previously. If we have nodes n_1 and n_2 and n_1 wants to send 10 messages to node n_2 who has 0 messages in common. The "ideal" parameter if the only goal was to increase the probability of "off by

1" would be to set e to the following values as the number of symbols in common increases: 1, 1, 1, 1, 1, 2, 2, 3, 5, 10. This, however, would be strategically a very bad idea as the chance of sending duplicate information is 50% when e is 1 and 5/10 symbols are shared.

What does all of this mean for the decode time? It means that, if the parameters are chosen well for network coding, then compared to a density of 1, the decode time could be as small as *half* the decode time if every message was "off by 1". On the other hand, it could degenerate into sending random packets if set improperly.

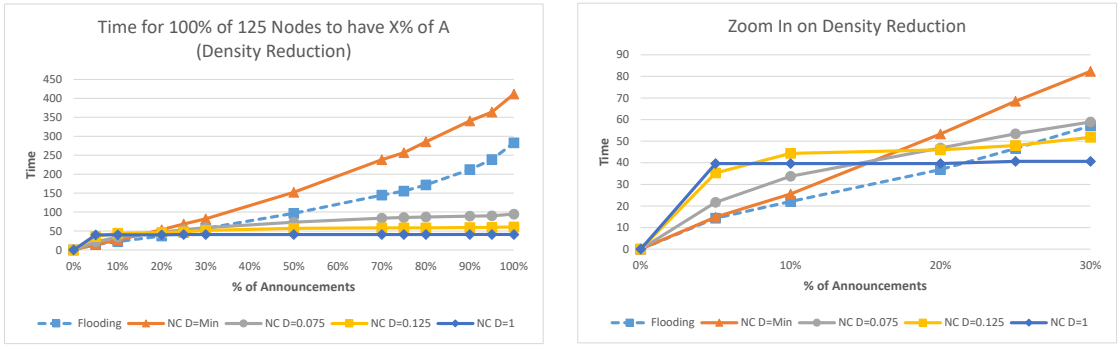
5.3.2 Experimental Setup and Evaluation

In this section I examine using the same experimental setup from Section 5.2, but varying the density parameter. In Kodo's RLNC library, the density parameter is based on the generation size, but it is more to our purpose to choose a density based on the node's current decoding matrix rank, so the actual parameter sent to Kodo is scaled such that:

$kodoParam = ((density * rank)/genSize) * maxDensity$

For Kodo, $maxDensity$ is $1 - (1/q)$ where q is the size of the finite field. The minimum density is set such that at least one symbol will be sent. Note that the default Kodo configuration (assuming one does not write their own encoder), has a target number of non-zero coefficients which is chosen with a binomial distribution around the chosen $kodoParam$ value, so even attempting to choose a specific density does not always result in exactly that density being used.

Figures 5.10(a) show the results of setting the density parameters to specific values below 1, and (b) is a zoom in for under 30% of the data advertisements. As we can see, setting the density lower produces a better "startup" time for the network, but setting it too low produces even worse results than the flooding. As discussed, this is because the symbols being sent are still chosen at random, so reducing the density too much means that nodes are sending random messages to each other in the hope of sending something useful. Since none of these density settings perform better than the full network coded version, I also looked at a local heuristic which sets the density parameter inversely proportional to



(a) (b)

Figure 5.10: Flooding vs. network coding with a variety of density (D) values.

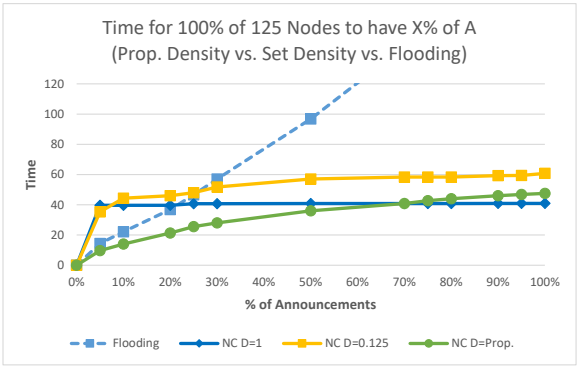


Figure 5.11: Density parameter heuristic evaluation in the small world.

the number of symbols missing at the receiving end: $density = 1/missing$. The result of using this heuristic is shown in Figure 5.11 as D=Prop. A partial line for flooding is also shown for comparison.

Looking at movement in Figure 5.12, we see that this heuristic holds for different (and much larger) networks. Again, the heuristic starts off much better than network coding with full density, but finishes off much better than flooding.

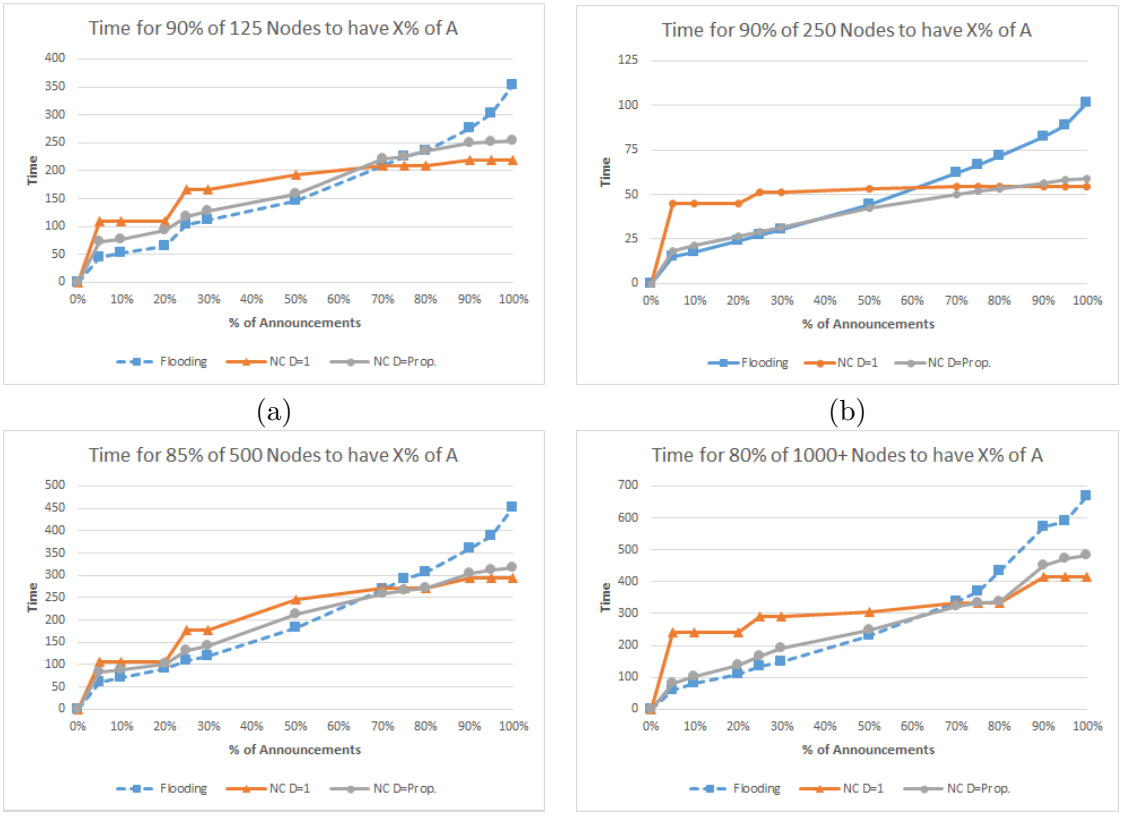


Figure 5.12: Density parameter heuristic evaluation in the GMU world.

5.4 Conclusion

In this chapter I have closely examined the problem of data advertisement in ad-hoc ICDTNs from an implementation perspective and presented a general procedure for approaching solutions to this problem along with specific details of implementing this within the NDN architecture. Through extensive simulation I have shown the benefits of applying network coding to this problem, as well as the trade-offs and parameter settings desirable for different types of networks with a variety of environments, different numbers of nodes, and several motion models. The heuristic shown in this chapter is generally applicable to *any* multi-source RLNC problem and is slightly more justified than the experimentally set "four items per generation" used elsewhere.

As a final remark, this chapter is *under*estimating the benefits of network coding for many applications. Network coding also adds a level of redundancy valuable for unreliable links and, in a wireless environment, overhearing an encoded announcement is much more likely to benefit nodes than overhearing an unencoded announcement. This will be briefly touched on in Chapter 6.

Chapter 6: Analyzing Data Announcement Protocols for Ad Hoc ICDTNs

This chapter develops a general-purpose probabilistic model for measuring key performance parameters of the data advertisement problem and shows how to use system level abstractions for multi-source broadcasting in mobile ad hoc networks as a means of analyzing ICDTN protocols. The model characterizes how to reduce the expected time to receive a new announcement, as well as how to shrink the "advertisement period", the time when nodes need to continue to exchange messages. I also quantify the design trade-offs for different protocols for solving the ICDTN data announcement problem, and studies three unique protocols. This provides future designers a mechanism to analytically understand the trade-offs for any future protocols and compare them to the ones proposed in this work.

Sections 6.1-6.3 detail the model and its assumptions, while in Section 6.4 I experimentally evaluate the accuracy of this approach by comparing the model's predictive value to the actual results under a number of topologies and disruption scenarios, utilizing standard network models. My results show that network coding offers clear advantages in denser networks when shortening the announcement period, while smart flooding benefits sparser networks or networks trying to expand the utilization window. I also empirically evaluate the impact of wireless overhearing, whereby a passive listener gains new data by eavesdropping on its neighbors' communication, and find that eavesdropping can offer tremendous performance benefits and that network coding in these environments can outperform smart flooding for utilization time. A shorter version of this chapter has been submitted for publication and is currently under review.

6.1 Model Overview

In this chapter I will be examining a general model for the data advertisement problem to make the problem tractable and understandable. I will specifically be examining the problem in networks with long term stable states: regular patterns of movement or contact which lead to stationary probabilities of two devices connecting ($P(connect)$) and predictable intercontact times (Γ_{I}). This is most easily observed in the use cases for the "node face" layer discussed in previous chapters: data mules that regularly move around a network, bus systems with regular routes, or stationary nodes with regular connection interference. It also is applicable for the "geo-region" face type if there are massive, predictable fluctuations of movement in the network, such as a metro-area which moves the majority of devices into a city during the day and then outward from the city in the evening. In short, any predictable (but not controllable) movement which causes convergent, stationary, connection probabilities.

I will be using the PUSH model discussed in Section 2.4. This is the model often adopted for studying the coupon collector problem where a single node is able to send to one neighbor at each time step ("round"). In this variation, however, we will limit nodes to only communicating with a subset of the network which they can regularly meet. This differs from the traditional coupon collector problem where nodes can communicate with any other node in the network.

The connection probabilities will be provided by one of two types of model: static connection probability networks and two-state connection probability networks which I will abbreviate as SP and TSP for brevity. SP networks have a single probability of connecting in any given "round" while TSP networks have two possible states, a high connection probability state and a low connection probability state (see Section 2.1). At any given step, they may transition state (or not) and will connect (or not) with the probability for their given state. Both SP and TSP networks are given by the Home-MEG model [31]. In this model p is the probability of transitioning to a high-state and q is the probability of

transitioning to a low state. The connection probability in each of those states is α (when in a high-state) and γ (when in a low-state). SP models are provided simply by setting q to 0 and α to the connection probability desired.

It should be noted again (see Section 3.4) that empirical studies of human mobility patterns show that the intercontact times are well approximated by exponential [34] or power law + exponential tail distributions [31], and I am basing my analysis on this distribution class. These are caused by the regular, recurring actions in human behavior, such as people going to and from work, while occasionally moving in more unexpected ways, such as when visiting a more distant relative. As has been demonstrated, the Home-MEG model can specifically be fitted to mobility traces to provide intercontact distributions in the same way as the Gilbert-Elliot model it is based on. These distributions are defined as follows, where all probabilities are dependent *only* on p, q, α, and γ:

$$P(\Gamma_I = k) = P(High|Contact)P(High|t_k) + P(Low|Contact)P(Low|t_k) \tag{6.1}$$

Note that $P(High|t_k)$ and $P(Low|t_k)$ are each defined recursively and the final equation is fairly long. See [31] for the full equations.

For consistency, I will be using the terminology from Table 6.1 in the this chapter. When needed, subscripts are used on sets to indicate a specific place, e.g. A_n would be announcements at n and subscripts are used on elements of a set to indicate unique elements, as in n_1 and t_0.

The remainder of this chapter is arranged as follows: Section 6.2 specifics of the protocols being examined; Section 6.3 outlines a general framework for understanding the delays in data announcement protocols and an analysis for SP and TSP networks using the general framework; and Section 6.4 presents experimental verification of my analysis.

Term	Meaning and Notes
N	Set of all nodes, where $n \in N$.
$\Upsilon(n)$	Neighbors of n, where v used for a single neighbor. $\Upsilon(n) \subseteq N$.
A	Set of all announcements, where $a \in A$.
G	Set of all generations, where $g \in G$. g is a partition or subset of A.
$cr(a), exp(a)$	Creation / expiration time of a.
$rec(n, v, k)$	Time slot n receives k from v.
$dec(n, a)$	Time slot n decodes a.
$avail(n, a)$	Time slot n can begin using a
$last(n, a)$	Last time slot n received/decoded a.
$\Gamma_{\mathrm{C}}, \Gamma_{\mathrm{I}}$	Contact / inter-contact time.
D	Density of encoding.

Figure 6.1: Terminology and Symbols

6.2 ICDTN Announcement Protocols

I will be evaluating three protocols in this chapter: naive flooding, network coding, and smart flooding. The first performs basic (naive) flooding explored empirically in Chapters 4 and 5. This involves no exchange of information between nodes beforehand, they simply tell every neighbor about every announcement, and the appropriate routing table information is created from this transaction. Naive flooding has the advantage of being the simplest to implement, and the disadvantage of having excessive overhead.

The second uses network coding techniques introduced in Chapter 5 which allows routing information to "piggyback" with the vectors already required by network coding. The advantage of this approach is to potentially minimize the time to receive multiple announcements, while a disadvantage is that, depending upon when and how messages arrive, announcement decoding may be delayed preventing immediate utilization. In the previous chapter I empirically showed that this technique works and performs substantially better

than a naive flooding approach, but did not extensively explore there *why* (mathematically) this was the case.

Finally, the third uses the pre-processing handshake and vectors from the network coding solution but picks one un-encoded message as the payload (instead of a mix of encoded messages). This is "borrowing" the component of network coding where messages have to belong to a *generation* and they have to have a known ordering within the generation. I will call this technique "smart flooding," and the advantage is to reduce the number of unnecessary advertisement exchanges while a potential disadvantage is a possibility for redundant information to be passed. This technique is new to this chapter and was originally explored to isolate the benefits that come from generation management from those that come from coded messages, but it was discovered to be an advantageous technique in certain scenarios, as will be discussed later.

It is essential to highlight the differences between the standard network broadcasting problem and the data announcement advertising problem in a DTN environment (discussed in Section 2.4), and to understand the three protocols being studied before continuing. I will start by summarizing the differences with an example utilizing Figures 6.2-6.5. All these figures shows a set of nodes where n_0 and n_2 are connected to a larger network, but n_1 is not; this is an abstraction of the busses and consumer from the bus example in Section 2.4 Figure 2.2. In this scenario, there are three announcements received at n_0 and n_2 (the busses) which need to be exchanged with n_1 (the consumer). To the right of each diagram are the forwarding tables constructed during each time step (D=Data, R=Route, H=HopCount). When sending, announcements are marked with A0-2 (for D0-2).

Figure 6.2 shows the exchange with naive flooding. In this example, if an announcement a is received at n_1, then n_1 will need to send the announcement to both n_0 and n_2 to complete the exchange. Remembering that part of this problem is the construction forwarding tables for a network that is in flux, this allows n_0 and n_2 to store alternative routes which may be *potentially* usable at a later time. For example, if n_0 becomes disconnected from the larger network but not from n_1, it may choose to send to n_1, which can send to n_2 as

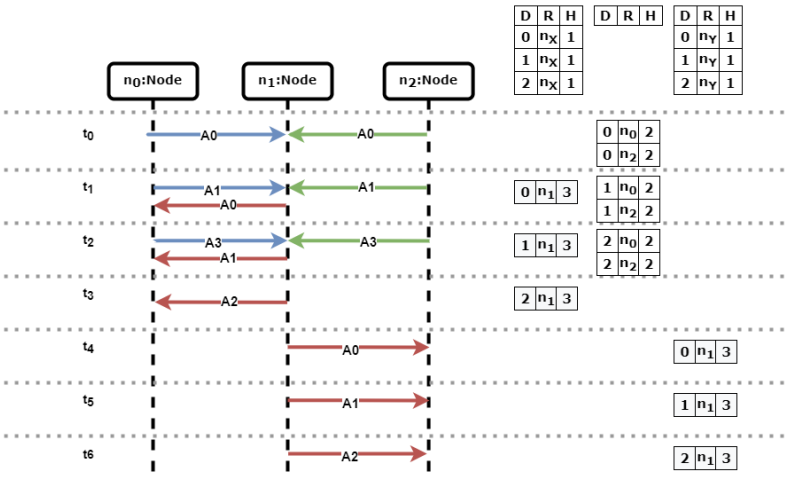


Figure 6.2: An exchange of three announcements with naive flooding.

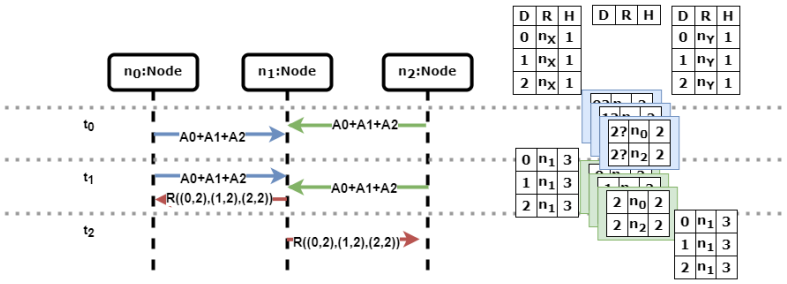


Figure 6.3: An exchange of three announcements with network coding.

an alternative route to the larger network. In the bus scenario, this would be helpful if the bus was used for a different route that passed the consumer but not the producer.

Figure 6.3 shows the exchange with network coding. Here announcements need to be exchanged until they are decoded, which is a much shorter process. The figure shows a question mark next to D in the table to indicate that the announcement metadata is still encoded (t_0) and then it is shown again without this when decoded (t_1). Note that route information can be passed before the announcement payload is decoded and piggybacked route information is shown as R(D,H) in the diagram.

Figure 6.4-6.5 shows the exchange with smart flooding in the worst-case and best-case,

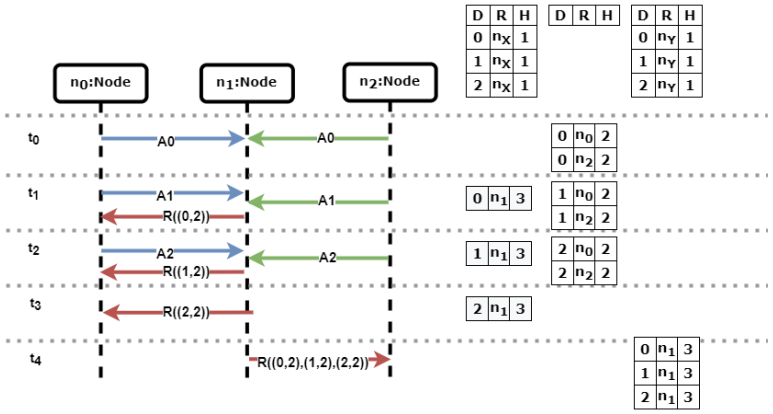


Figure 6.4: An exchange of three announcements with smart flooding (worst case).

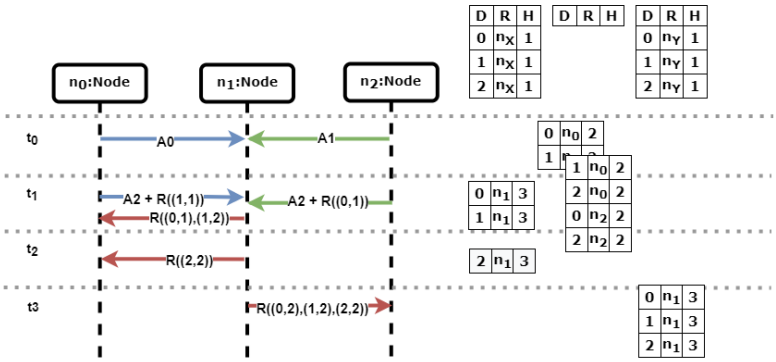


Figure 6.5: An exchange of three announcements with smart flooding (best case).

respectively. For smart flooding, nodes may send redundant announcement information or they may happen to send different information, this is what causes there to be worst and best case. This happens because they are unaware of what other nodes in the system are sending (similar to the hidden node problem, but with redundancy rather than interference). This is why network coding and naive flooding do not have this range of possibilities, their algorithms are not affected by other devices' choices. Naive flooding will always need to send, and encoded messages either contain new information or can be used for decoding, so there are no "wasted" packets with redundant information in either case.

6.3 Advertisement Exchange Analysis

In this section I will propose two possible objectives for the data advertisement process. The first measures the amount of time a node has to use each announcement. I call this objective the *utilization time* and the goal is to increase this value.

Objective 1: Utilization Time

$$\sum_{a \in A} \sum_{n \in N} \max\left(0, exp(a) - avail(n, a)\right)$$
$$\text{where: } avail(n, a) = \min_{\upsilon \in \Upsilon(n)} \left(rec(n, \upsilon, a) + dec(n, a)\right) \tag{6.2}$$

The second objective measures the time nodes spend exchanging advertisements. I will call this the *announcement period* and the goal would be to reduce this.

Objective 2: Announcement Period

$$\max_{a \in A, n \in N} last(n, a)$$
$$\text{where: } last(n, a) = \max_{\upsilon \in \Upsilon(n)} \left(rec(n, \upsilon, a) + dec(n, a)\right) \tag{6.3}$$

Given that we cannot change the $exp(a)$, as discussed in Section 3.3, the only way to optimize these equations is to determine the contributing factors for the receive time, $rec(n, \upsilon, a)$, and the decoding time, $dec(n, a)$. The first has to do with the DTN environment and the second is related to the network coding parameters chosen. If no network coding is used, $dec(n, a)$ is simply 0, so from here we can look more closely at the trade-off of receiving a message earlier with network coding compared to the time to decode a message.

I will examine $rec(n, \upsilon, a)$ first and describe the different conditions a node must wait for. Informally, the $rec(n, \upsilon, a)$ is equal to the time for υ to receive a, plus the time to connect n and υ, plus the time before sending. The base case for this recursive definition is $cr(a)$ for the producer of the announcement.

Time to Connect. After the neighbor has received the announcement, the two nodes must connect in the DTN environment. I will write this as the $E[connectTime]$. The equations for this are different for SP and TSP networks and will be discussed later on.

Time Before Sending. Just because two nodes *can* send to each other, does not mean that they will. The PUSH network model specifies sending at most one message to at most one neighbor at any given time slot, so nodes must choose who to send to and where to send it. I will write this as $sendTime(E[msgsBeforeSend])$ where $E[msgsBeforeSend]$ is the expected number of messages that will be sent *before* the announcement we're hoping for, and $sendTime(m)$ returns the expected time needed to send m messages from the queue.

So the receive time of an announcement is more formally:

$$\begin{aligned} E[rec(n,v,a)] = & \min_{v' \in \Upsilon(v)} E[rec(v,v',a)] \\ & + E[connectTime] \\ & + sendTime(E[msgsBeforeSend]) \end{aligned} \tag{6.4}$$

It is important to reiterate at this point that the $E[rec(n,v,a)]$, $E[connectTime]$, and all other "time" values are being measured in time slots, or "rounds" in the PUSH model, not in continuous time. Additionally, we are working with probabilities of a given action at a given time slot, such as connecting with $P(connected)$, so the action occurs or does not occur at that time. This allows us to have expected values for these time measurements.

In the following subsections I first examine the specifics of calculating $E[connectTime]$, $E[msgsBeforeSend]$, and $sendTime(m)$ for SP and TSP networks, then discuss the decode time for messages sent with network coding.

6.3.1 Expected Receive Time

In order to determine $E[connectTime]$ and $E[msgsBeforeSend]$, we need to know some information about the network: (1) the probability that any two neighbors are connected,

$P(connected)$, and that $1 - P(connected)$ is $P(disconnected)$, (2) how many, neighbors a node has, $|\Upsilon_n|$, and (3) the intercontact time Γ_{I}. These parameters are reasonable for networks where the motion models create predictable connections – buses go on routes, people go to work and home, satellites orbit things, and so forth. The Home-MEG mode can determine $P(connected)$ for SP and TSP networks and we will show how to use these parameters to determine the $E[connectTime]$. The number of neighbors can be determined from a data trace or pre-set with knowledge of node movements.

Connection Delays. The $E[connectTime]$ for two *particular* nodes to meet depends on the type of network. The PUSH model defines connections as single events that do not have a duration. In this case, if the network can be modeled as an SP network, then $\frac{1}{P(connected)}$ is appropriate. In a TSP model, node pairs have some chance of transitioning to one or more states, each with a certain probability of connecting. This makes the connection times for two *particular* nodes in the Home-MEG model:

$$E[connectTime] = \frac{p}{p+q} * \frac{1}{\alpha} + \frac{q}{p+q} * min\left(\frac{1}{\gamma}, \frac{1}{p} + \frac{1}{\alpha}\right) \tag{6.5}$$

Where we're taking the minimum time to transition to connect in the not-home state or the time to transition to the home state and connect there.

Sending Delays. To calculate the time delay caused by a node deciding *not* to send a particular message to a particular node we need to look at the number of pending messages. There are several possible models for deciding who to send to. For example, nodes could choose one neighbor who needs an announcement, then choose which announcement to send (this "fairly" treats neighbors who may be waiting for only a single announcement). This results in the following probability for sending:

$$P(\text{sending from } \upsilon \text{ to } n_i) = \frac{1}{\sum\limits_{\upsilon' \in \Upsilon(\upsilon)} [|A_\upsilon - A_{\upsilon'}| \neq 0]} \cdot \frac{1}{|A_\upsilon - A_{\upsilon'}|} \tag{6.6}$$

Alternatively, for flooding, nodes can choose to send any announcement that is pending rather than choosing a neighbor and thus neighbors who need more announcements will have a higher chance of receiving something. This results in the following alternative:

$$P(\text{sending from } \upsilon \text{ to } n_i) = \frac{1}{\sum_{\upsilon' \in \Upsilon(\upsilon)} |A_\upsilon - A_{\upsilon'}|} \tag{6.7}$$

This option is more complicated for network coding which would need to weigh in the density parameter and the number of announcements currently at υ if it wanted to treat each announcement "equally".

For analysis, I make a few common simplifying assumptions: (1) nodes choose a neighbor to send to using a uniform distribution, and (2) that they always have something to send to their neighbors. This makes the calculation more tractable since it doesn't depend on the current state of each node's queue and decoding matrix, instead it is dependent on $|\Upsilon(\upsilon)^t|$. If nodes have a uniform distribution on the numbers of neighbors, this gives an expected value as follows when a node has at least one connection:

$$E[connections] = \frac{E[|\Upsilon|] \cdot (P(connected))}{P(disconnected)^{E[|\Upsilon|]}} \tag{6.8}$$

In other words, the expected number of connections would be the percentage of the neighbors the node has and is connected to. The denominator scales this for a DTN environment where *many* nodes are completely disconnected (meaning the connections that do exist are distributed less evenly). This means that the probability of sending to a particular neighbor ($P(sending)$) is simply $\frac{1}{E[connections]}$.

I represent the queue size for all neighbors, by scaling the queue size for *one* neighbor ($E[queueSize]$) by the probability of sending to that particular node:

$$E[msgsBeforeSend] = \frac{E[queueSize]}{P(sending)} \tag{6.9}$$

The number of messages in queue *for a specific neighbor* is $E[queueSize]$, which is scaled by the probability of sending to any specific node to account for additional messages in the queue for other nodes. We could, in this case, multiply by $E[connections]$, but using the probability of sending allows a more general formulation which I will use later when discussing "overhearing" messages in the experimental results section.

It is this $E[queueSize]$ where we will see the major differences between naive flooding, smart flooding, and network coding. The queue size for naive flooding grows quickly since each unseen announcement received at n generates $|\Upsilon(n)|$ messages to send. If the announcements are generated very quickly relative to the $E[|\Upsilon|]$, the queue will grow very quickly as well. Smart flooding and network coding, on the other hand, will distribute the responsibility of sending the main body of the announcement amongst all neighbors. I will give some back of the envelope math for estimating $E[queueSize]$ for each technique in Section 6.4.

Sending Time. Based upon the previous discussion, I will next examine $sendTime(m)$ for m messages. This is m units of time if (1) every node can send one message at each time slot – an assumption of the model, (2) the queue is FIFO so that no new announcements are inserted, and (3) nodes do not become disconnected before sending all messages. This last may or may not be realistic for a DTN. If nodes *can* becomes disconnected one can factor that in using the following formula for m messages:

$$sendTime(m) = m + \left[\frac{m}{E[\Gamma_{\mathrm{C}}]} - 1\right] \cdot E[connectTime] \tag{6.10}$$

For this chapter I assume the simpler formula of $sendTime(x) = x$ and measure $E[msgsBeforeSend]$ for comparison purposes between different types of networks in the results section.

6.3.2 Expected Decode Time

For $avail(n, a)$ and $last(n, a)$ in Equations 6.2 and 6.3, we are still missing $dec(n, a)$ for network coding. With flooding, after receiving an announcement a node can immediately utilize it. However, in the case of network coding, the messages must be decoded from the decoding matrix first which requires receiving a certain number of coded messages containing a given symbol.

Given the discussion of these parameters in Sections 3.3 and 5.3, I start to formalize $dec(n, a)$ by building up $P(encoding)$, the probability that a node chooses a given announcement to "mix in", chosen based on a density parameter (D) which is appropriate for the ad hoc environment. This means that the number of encoded symbols at any time is $genSize \cdot D$ where $genSize$ is $E[|ann(gen(a))|]$. Which also happens to be the number of messages n needs to receive containing any one symbol to *guarantee* decoding.

Choosing one generation and sending only a percentage of that generation makes the probability of encoding any one announcement a:

$$P(encoding) = \frac{1}{|G|} \cdot \frac{numEnc}{genSize} = \frac{D}{|G|} \tag{6.11}$$

Messages used for decoding can come from any neighbor, so this would mean that the number of messages to fully decode would be $\frac{numEnc}{P(encoding)}$. Assuming that any one message sent is on average half way through receiving all the messages from a generation, we have:

$$E[msgToDecode] = \frac{genSize \cdot |G|}{2} \tag{6.12}$$

In a network where each node receives approximately one message per round, the time to decode can be estimated using:

$$E[dec(n, a)] = \frac{(genSize \cdot |G|) - 1}{2 \cdot P(disconnected)^{E[|\Upsilon|]}} \tag{6.13}$$

Note that we have one less message because we've already sent one encoded and, as with Equation 6.8, we need to scale the time up to account for disconnected nodes.

This would seem to indicate that it is unimportant what the density parameter is set to, but $E[dec(n, a)]$ assumes all messages need to be *fully decoded* before being used and ignores the potential for partial decoding. The impact of partial decoding was covered in more detail in Section 5.3. However, using the formulation in this chapter gives a type of upper bound on how we can expect network coding to perform.

In this chapter I examine a "vanilla" implementation of network coding, where I choose to put all messages in a single generation and have a density of 1. However, this does not mean that nodes will always encode all announcements together in every message. If a node doesn't have an announcement, for example, the message can't be mixed in, so we will be seeing that the average messages needed to decode are about half the maximum possible, per the above formulation.

6.4 Empirical Analysis and Comparison

In this section I validate my analytical approach from the previous section. I then show which protocol — naive flooding, smart flooding, or network coding — performs best for our two objectives from Equations 6.2 and 6.3. To do this, we need to be able to choose appropriate parameters for $P(connected)$ and $E[\Gamma_{\mathrm{I}}]$. Since we examine SP and TSP networks in this section using the Home-MEG model, the probability of being in each state, as well as the other required statistics can be computed from the four parameters p, q, α, and γ (see [31]).

I also need to be able to estimate the queue size for the three communication models. This is non-trivial since every previous contact impacts the items a node has to send to any new contact. I give some "back of the envelope" estimates here to get *rough estimates* of

the queue of messages for a *particular* neighbor, excluding the item of interest:

$$E[maxQ] = \begin{cases} |A| - 1 & \text{NF} \\ 1 + \frac{|A|-2}{E[|\Upsilon|]} & \text{SF} \\ \frac{1}{P(encoding)} & \text{NC} \end{cases} \tag{6.14}$$

$$E[queueSize] = \frac{max(1, min(|A| - 1, E[maxQ]))}{2}$$

These estimates are based on some simplification of how the queues are behaving. For naive flooding a node has to send all announcements to their neighbor, for smart flooding nodes have to send at least one message to share table info, but announcement contents can be evenly received from each neighbor, and for network coding all messages can be sent at once, but for any given announcement the $P(encoding)$ determines if it will be randomly selected. The result is bounded by 1 and the number of announcements-1 (for the announcement being sent in this time step). This particular formalization means that the $E[msgBeforeSend]$ will need to have 0.5 removed to counter balance the minimum of (1/2) to give 0 after scaling for $P(sending)$. Finally, this estimate assumes most nodes have about half their queue at any given time.

6.4.1 Experimental Setup

In my experiments I looked at 60 different network types, examining a number of different SP networks with $P(connected)$ set to 1.0, 0.5, 0.25, and 0.125, as well as TSP networks with $p/q/\alpha/\gamma$ set to 0.5/0.25/1/0, 0.01/0.1/1/0, 0.5/0.25/0.66/0.01, 0.25/0.5/0.66/0.01, 0.05/0.1/0.75/0.25. I examined varying network sizes of 10, 20, 78, 100, 150, or 300 nodes, and various expected neighbor values from $|N|$-1 to ($|N|$-1)/30. For clarity, not all networks are shown, but rather a sample of representative results.

As mentioned in Section 2.1, the Home-MEG model can be "fit" to different data sets, so I evolved parameters ($p/q/\alpha/\gamma$) for the non-static nodes in the InfoCom06 conference

trace [83], which is a new data set not shown in [31][1]. The subset of the trace used had 78 nodes, with an expected number of 74 neighbors. As in [31] I chose to minimize the mean square error in log scale between the data trace and the ccdf for the Home-MEG model and used this as the "fitness". The resulting $p/q/\alpha/\gamma$ was 0.00232/0.0171/0.119/0.000401 and $P(connected)$ was 0.014569. This parameter set is labeled as "InfoCom06" in the results.

For each experiment I created a random network of nodes using the parameters given and created/broke connections while simulating announcement message passing at each time step. The connections were transitioned according to the connection probabilities or the Home-MEG model transition equations. Each experiment was run 10 times, and the averages are shown along with 95% confidence intervals where interesting.

I measured the three major contributing factors to the equations for $E[rec(n, v, a)]$: (1) $E[connectTime]$ from Equation 6.5, (2) $E[msgsBeforeSend]$ from Equation 6.9, and (3) $E[msgToDecode]$ from Equation 6.12 to ensure a fair comparison between network types. Next I measured the average minimum receive times (Equation 6.2) and the announcement period (Equation 6.3). All results, where applicable, are compared with the values from plugging in the appropriate parameters to the equations in the previous section. This is listed as "predicted" in the results.

6.4.2 Experimental Results and Discussion

Component Equations. For the component equation measurements, Figure 6.6 shows the time to become connected to a neighbor after the neighbor receives the announcement for a sample of networks examined. This measures the results of $E[connectTime]$ from Equation 6.4.

Figure 6.7 shows the number of times a node could have sent a message, but chose not to for a selection of networks examined; top and middle rows show various SP networks, while the bottom row shows various TSP networks. This measures the results of $E[msgsBeforeSend]$ from Equation 6.4 using the estimates from Equation 6.14.

[1]The data set in [31] referred to as Infocom06 is from the previous year by the same group.

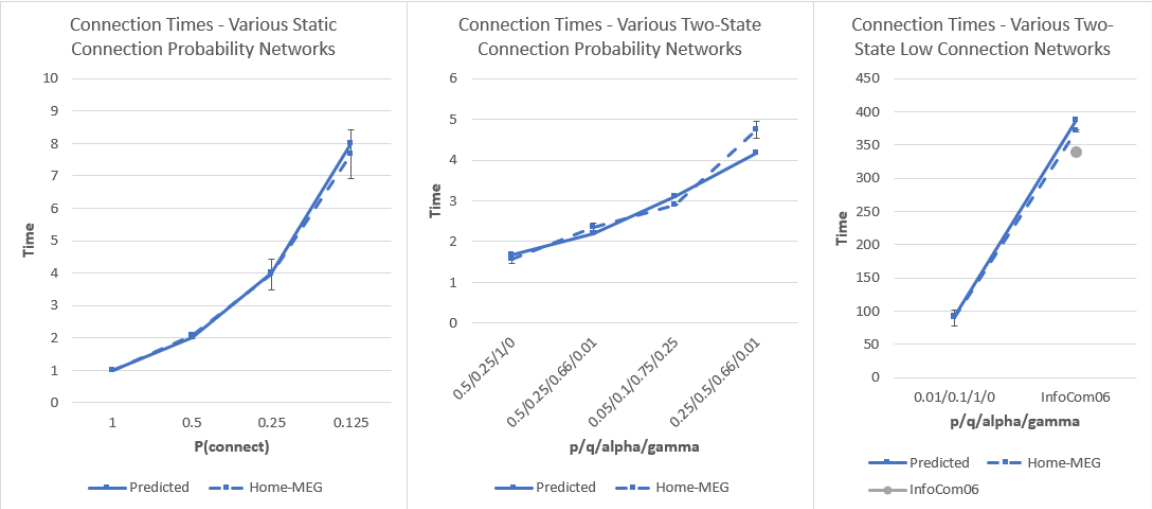


Figure 6.6: Results for $E[connectTime]$ from Equation 6.4.

Figure 6.8 shows the number of messages to decode for InfoCom06 parameters with the Home-MEG model and the InfoCom06 trace. This measures the results of $E[msgsToDecode]$ from Equation 6.12.

The time for a neighbor to connect after receiving a message is well fit by the equations, though we can see that for the sending opportunities before success, which uses the "back of the envelope" math for the queue size, is only a close approximation for some networks. The messages to decode are also consistent with the equations and $E[dec(n, a)]$ is consistent with Equation 6.13.

Data Trace. Since the Home-MEG model was fit to a data trace, I also ran the simulation using this trace. The network described by the trace is extremely heterogeneous: the $|\Upsilon_n|$ varies from 51-77, nodes do not have an even $P(connect)$ meeting between 1 and 221 times during the trace, the number of connections varies from 331 to 2573 contacts per node, and the trace is not long enough to reach the end of the announcement period, so the statistics lean towards the nodes in denser areas of the network. Nevertheless, the equations are a good match for the time to connect, shown in Figure 6.6 (right) as a small dot, as well as the decode times in Figure 6.8. The delays for sending were the major difference since

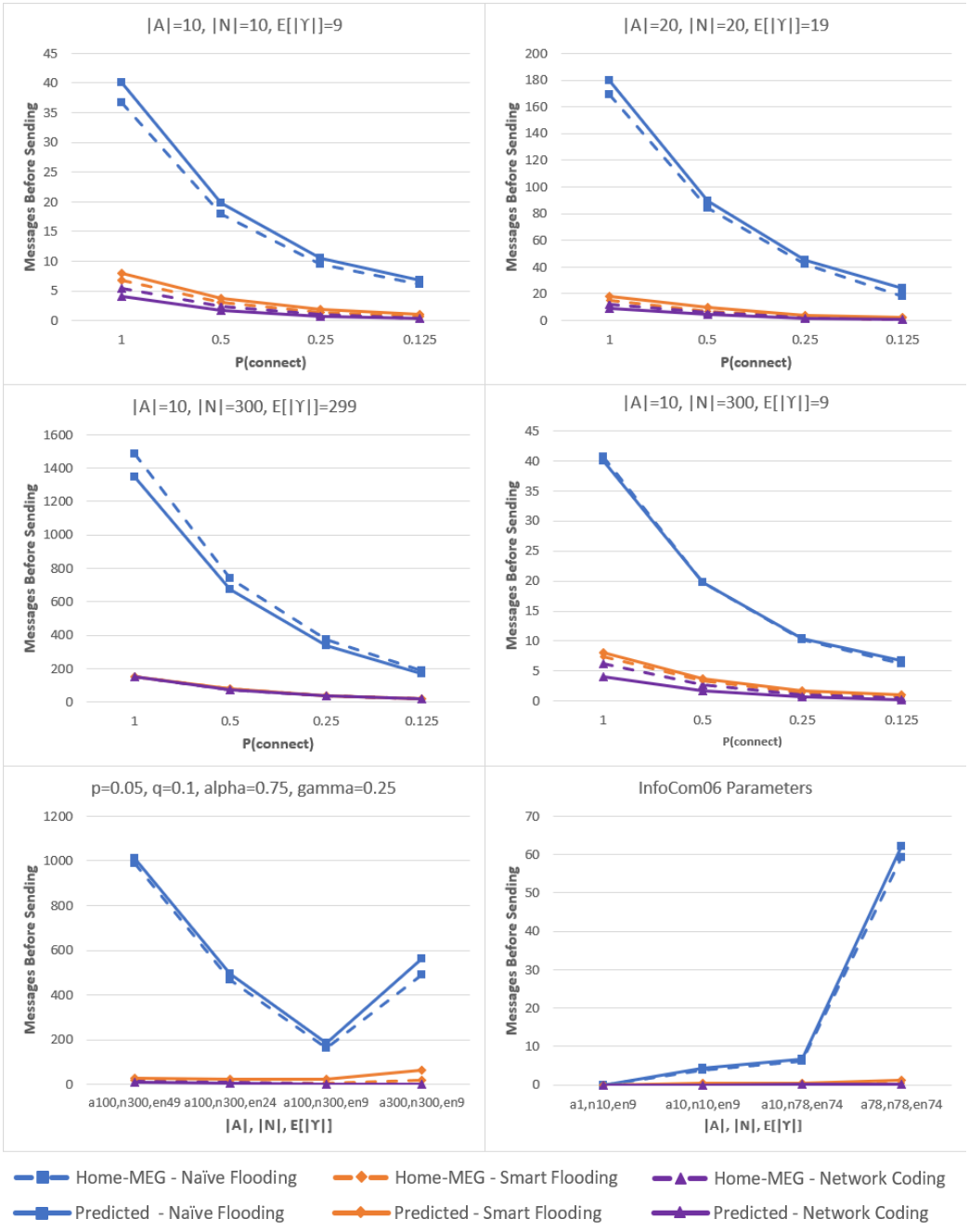


Figure 6.7: Results for $E[msgsBeforeSend]$ from Equation 6.4.

the network has a number of "bottleneck nodes" and therefore the $E[queueSize]$ is larger, $\frac{|A|-1}{1.5}$ produces the correct results for a=1, 10, and 39, and for a=78, $\frac{|A|-1}{1.25}$. Figure 6.9

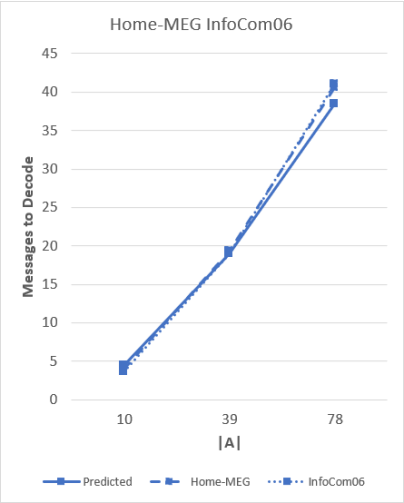


Figure 6.8: Results for $E[msgsToDecode]$ from Equation 6.12.

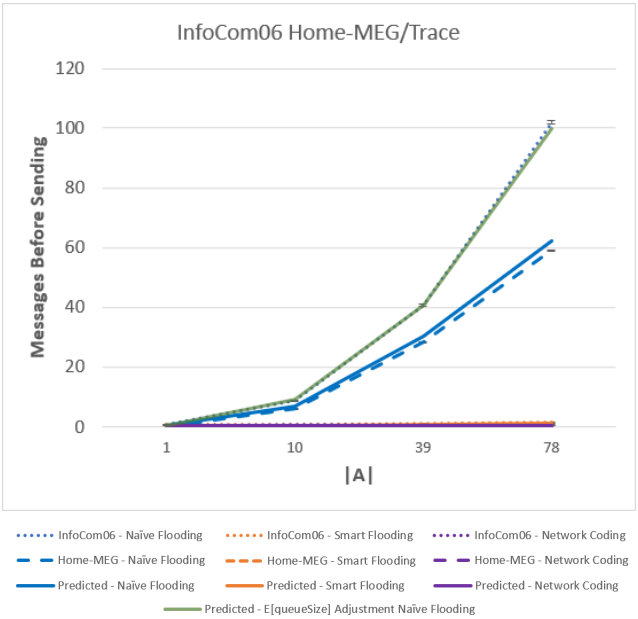


Figure 6.9: Results for $E[msgsBeforeSend]$ for the InfoCom06 trace.

shows these results. All values for smart flooding are below 1.5 messages and all values for network coding are below 1.

Utilization Time and Announcement Period. From the component equation results, we

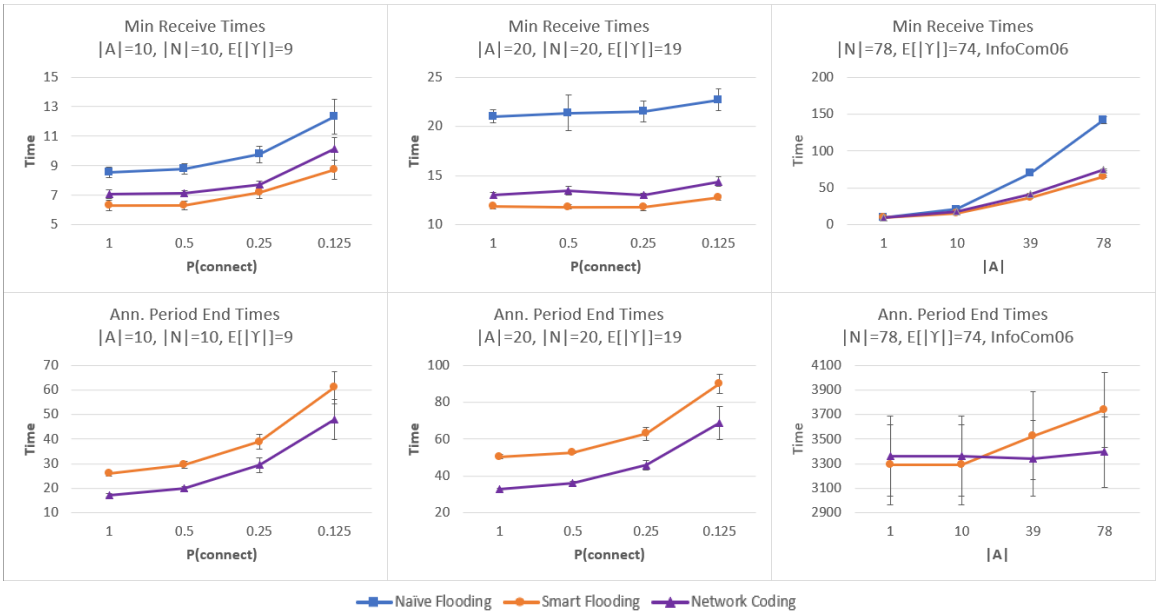


Figure 6.10: Results for Equation 6.2 and Equation 6.3.

can see clear trends for each part of the overall equations, but it is hard to see how they all "come together". For this, Figure 6.10 shows the average minimum receive time (top) as well as the announcement period end times (bottom) for some representative networks. This measures the results of Equation 6.2 and Equation 6.3 respectively. The completion times are not shown for naive flooding as it is always significantly worse.

From this we can see that reducing the announcement period (Equation 6.3) can be achieved with a simple implementation of network coding when the network is somewhat dense, whereas if the goal is expanding the utilization time *during* an announcement period (Equation 6.2), nodes should keep messages unencoded. For sparse networks, there may still be some advantage to coding if the proper parameters can be found to minimize the decode time such as using smaller generations, lower density, and so forth, but just naively encoding all possible messages together does not seem to have an advantage unless the number of announcements to send is extreme.

Overhearing. Finally, as I have noted a number of times in this work such as in Sections 2.5 and 5.4, there is an additional benefit of network coding that we have not explored:

network coding can increase the benefits of allowing nodes to wirelessly overhear neighborhood transmissions because sends to one node may be of interest to other nodes in the area. This is especially helpful in noisy networks where transmissions may be unsuccessful due to interference.

Put briefly, when $P(overhearing)$=1 a node can hear everything sent by any of its neighbors, so $P(sending)$=1 and the expected queue size needs to be adjusted for sparse networks where there is a low probability of two neighbors sending identical information, empirically dividing the queue size by $E[connections]$ is a good estimate in these cases. Loss is a reduction of $P(sending)$ from the receiver's point of view. Figure 6.11 shows the minimum receive times, when there is a probability of overhearing (rows 1 and 3), and there is a 100% chance of overhearing ($P(overhearing)$=1) but a lossy environment (rows 2 and 4). Note that losses can occur in overhearing and naive flooding is omitted from announcement period end times it is always substantially worse. As can be seen, under virtually all circumstances overhearing reduces the minimum receive time, but I note with interest that it also allows network coding to outperform smart flooding when maximizing the utilization time (Equation 6.2).

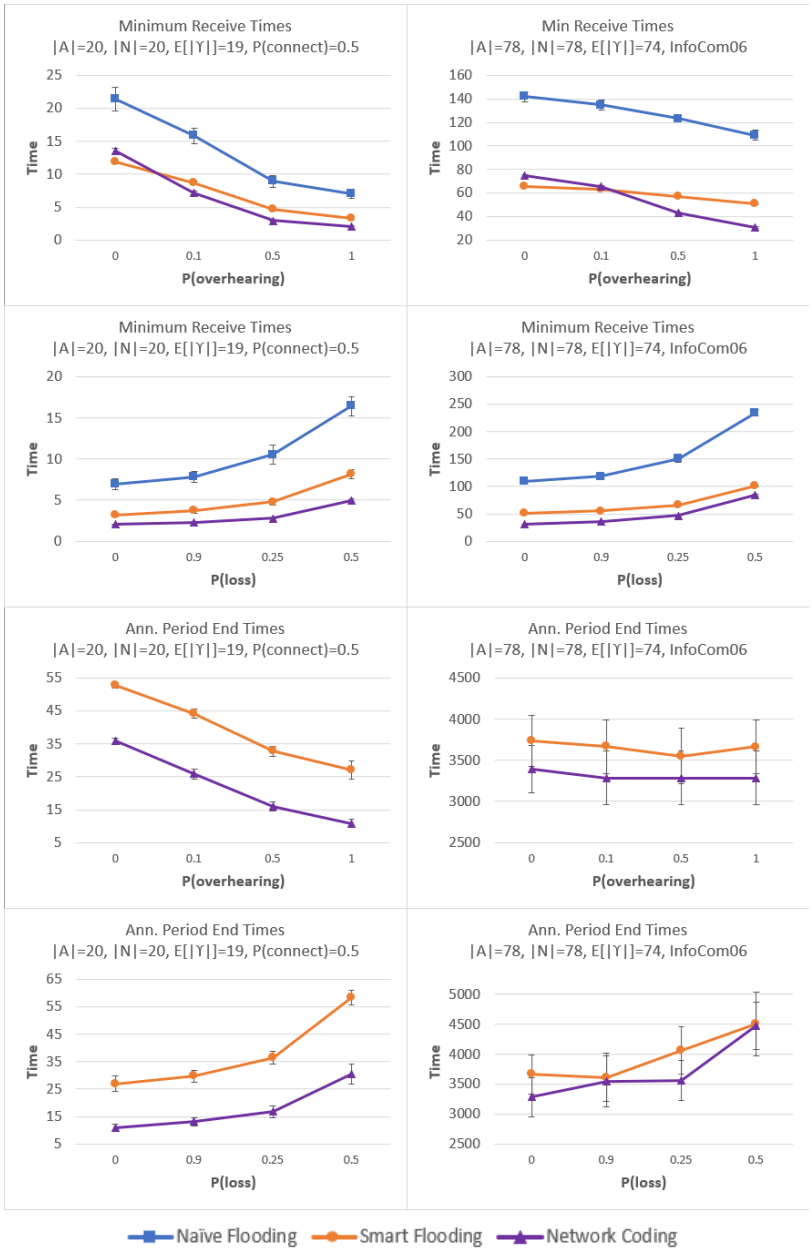


Figure 6.11: The effects of a overhearing on the average minimum receive time.

6.5 Conclusion

In this chapter I have focused on addressing the gap between the theory of data advertisement in ICDTNs, which can be seen as a unique variation of several well studied problems,

and practical implementations to solve this problem discussed earlier. The design tradeoffs inherent in any solution have been quantified into system level abstractions which can be used to compare different protocols, and I examined three alternative solutions using these metrics. Both analytically and empirically I have shown that there are clear benefits to using network coding, and even network coding inspired techniques. Taking only the generation management from network coding with the "smart flooding" protocol, we can see a substantial improvement over naive methods and found that this solution is even preferable to optimize the minimum receive time during the announcement period. On the other hand, full network coding solution perform better to reduce the announcement period and empirically offers additional gains in eavesdropping networks.

Chapter 7: Ad Hoc ICDTNs for a Metro Level Disaster Response

One of the motivating applications of my work is to support metro level networking during infrastructure collapse, and to allow communication to continue in an ad hoc manner during these times. Research on city-scale networks is made more challenging by fact that many of the application areas, such as natural or human-made disasters, are relatively rare, high risk, and sometimes highly politicized. Testing new and interesting networks on a devastated population, even disregarding the financial and political difficulties, often have moral implications prohibitive on their own.

With that said, it is still *absolutely* necessary to study these networks at this scale to understand the unique challenges of a city-scale network. Simulation is the obvious alternative, but full network simulators for millions of devices are not readily available. Additionally, for a non-static network, not only do the devices themselves need to be simulated, but also the people, cars, airplanes, or other "agents" carrying the devices, as well as the interplay between the network and the agents. This last is important since the network must change due to agent movement, needs, and preferences, and in turn the agents must change their movement, needs, and preferences based on the information they receive from the network, or even in some cases in order to help the network.

Therefore, to make simulation of city-scale networks tractable, many simplifying assumptions about the world need to be made. Simplified network models are common networking research, for obvious reasons, but agent-based models are available in this community. Happily, many other disciplines, such as Artificial Intelligence (AI) and Geographical Information Systems (GIS), have such models. The most appropriate choice here is Agent Based Models (ABMs). As introduced in Section 2.6, ABMs offer models for "agents" (actors in

the world) to respond to their environment, for example, moving towards food when hungry or sleeping when tired. These models can be extremely simple, or very complex depending on scenario being simulated.

Unfortunately, while models *exist*, combining agent and network models in a non-trivial example is not an easy task. Simulators designed to handle over a million city residents are not well equipped to handle even very simplistic network models, and certainly not low-level network simulation code. For the same reason, network simulators aren't well equipped to scale to a city-size number of devices. Therefore it is hard not to trivialize, idealize, or otherwise "abstract away" important network components during simulation.

This chapter discusses a successful city-scale simulation which was done by taking an existing GIS and agent-based model for humanitarian assistance in a city of 1.3 million people (agents), and providing a network simulation layer that approximates the geo-region architecture discussed in Chapter 4. This combination of models is able to simulate the full set of agents participating in the ad hoc ICDTN, while still keeping the core features of the architecture.

This agent-based simulation offers several contributions to the current field of research. First, it shows that the hybrid architecture proposed in Chapter 4 *is* capable of scaling with the "geo-region face" middle layer to deliver up-to-date information in an ad hoc manner in a large scale disaster scenario. Second, it exposes and examines several challenges not addressed in the literature which are unique to city-scale ICDTNs, including the interplay of NDN parameters, the effects of local caching, and the challenges with mass human movement. And third, it presents a successful integration of networking and ABMs for the study of network performance in a disaster scenario. Section 7.1 discusses the scaling issues for the geo-region architecture, Section 7.2 discusses the details of the ABM simulation, and Section 7.3 presents an experimental evaluation of for a variety of interesting network and simulation parameters.

7.1 Scaling the Geo-Region Architecture

In this section I will discuss the interplay between scale and the geo-region architecture discussed in Chapter 4, as well as the simplifying assumptions needed to simulate this at a large scale. Additionally, in a city-wide disaster scenario there are additional challenges which simply do not manifest given only a few hundred or even a few thousand nodes.

First, at a small scale it is possible for all devices to receive data from any given producer, but the reality is that it is quite possible for data to expire *on the way to* some node in a city-size ICDTN. The delays caused by the disrupted environment actually make this a very likely situation if data regularly expires and must be refreshed.

Second, most motion models create movement such that the areas which start with nodes will continue to have nodes. This is because random movement and human movement models tend to produce a spatial distribution of nodes which is fairly stable. In a real disaster scenario, however, large portions of the devices might be carried away from devastated areas and parts of the world become "bare" or abandoned. This means that the network needs to reconfigure to handle these missing areas of the world.

Third, there is some interesting interplay between the in-network caching and the network's performance. If the data expires regularly, as one would expect for a disaster scenario where updates are frequent, then there is an interesting interplay between the previous two problems (expiration and abandoned areas) and ICN in-network caching.

These three major challenges are addressed in the next sections: Section 7.1.1 discusses the relationship between message timeouts for each type of message, Section 7.1.2 addresses issues arising from mass movement of agents in the environment, and Section 7.1.3 covers the impact of the content store on the previous two issues.

7.1.1 Timeouts and Lifetimes of NDN Advertisements, Interests, and Data

Looking first at the interplay of timeouts, we have three major components to examine: the data advertisement lifetime, the data lifetime, and the interest lifetime. The advertisement

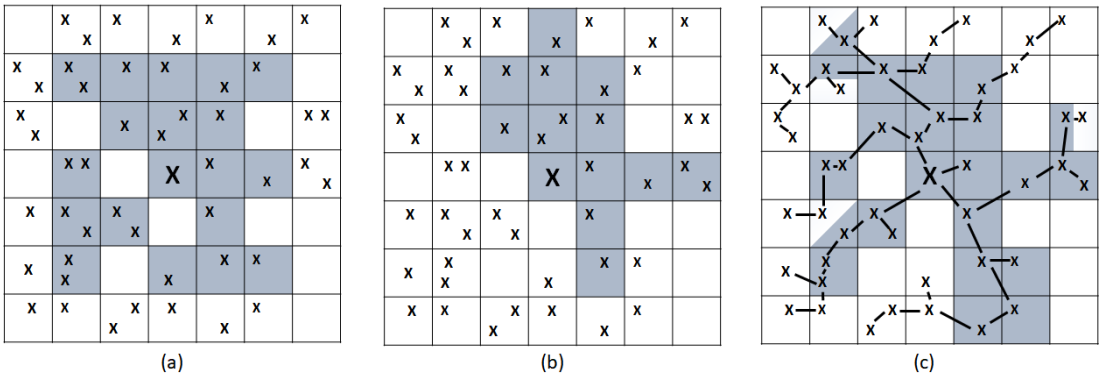


Figure 7.1: Announcement leap limits and resulting area coverage.

lifetime controls how far away a node can be and still get information in the network. As has been noted elsewhere, announcements need to have a "valid period" indicating a time at which nodes should add or delete the entries from their FIB table, and this is related to, but not the same as, the announcement message expiration, which controls how far the announcement message can be carried from the data producer (see Section 3.2).

Keeping with the architecture described in Chapter 4 I will be examining the lifetime as the number of "leaps" at the NDN level of the network (the "leap-limit") which is analogous to the more standard hop-limit. Depending on the concept of leap being used (see Section 4.1) this can have a large impact on the advertisement area. Figure 7.1 shows different types of network leaps with three different leap-limits and the resulting area of coverage. Figure 7.1(a) shows a two-leap geo-region-face network with Moore neighborhoods (8 neighboring regions), (b) shows a three-leap geo-region-face network with von Neumann neighborhoods (4 neighboring regions, no diagonals), and (c) shows a non-mobile four-leap node-face network. For (c) the minimum spanning tree when all connections are present also shown for clarity. Data producer shown as large 'X' in the center, and potential data consumers are shown with smaller 'x'. Shaded areas indicate geographic regions where the announcements are available. Note that in the node-face network, an entire geo-region is not necessarily able to receive the announcement.

The relationship between the leap-limit for an advertisement and the data lifetime is a little complex. If the announcement leap-limit is too long, relative to the data lifetime, some nodes in the network will be able to request the data but will never be able to receive it (they are what I will call "over informed"). This is because the data will expire *on the way to* the consumer. On the other hand, if the announcement leap-limit is too short relative to the data lifetime, some nodes are unaware of the data that could (physically) get to them without timing out (they are "under informed").

The relationship between the data lifetime and the interest lifetime is more straightforward. If the content store is not being used to cache data (I will discuss the alternative later), then the only way for an interest to be satisfied is by one of the two scenarios: (1) the interest has to travel all the way to the original data producer then all the way back to the consumer, or (2) the interest happens to join up with a pending interest request already sent towards the data producer and only needs to wait for that reply. Figure 7.2 shows these two scenarios for an interest from consumer 'C' for data produced at 'P'. Interest paths along with data reply paths shown as dotted red lines. For the two scenarios, (a) shows what happens when there are no pending interests along the same path – worst case – and (b) shows what happens when there are already pending interests along the same path – best case.

The intuitive thing to do, given the above information, to set the interest lifetime to double the data lifetime to cover the worst-case scenario. This does, however, produce an interesting problem when it comes to NDN's path reinforcement: if a path becomes invalid for some reason, and a node waits for the interest lifetime to expire before updating the face statistics and choosing a new path, then the speed at which the network can respond to broken paths can be hampered by the interest lifetime. Nodes who *should* be giving up on paths that are invalid think they are just waiting for their request to get to the producer and back. On the other hand, if the interest lifetime is set shorter, then some nodes may become over-informed (they are aware of information they can't get) as with a misconfiguration of the data lifetime. In this case, however, they cannot get the data because their interest in

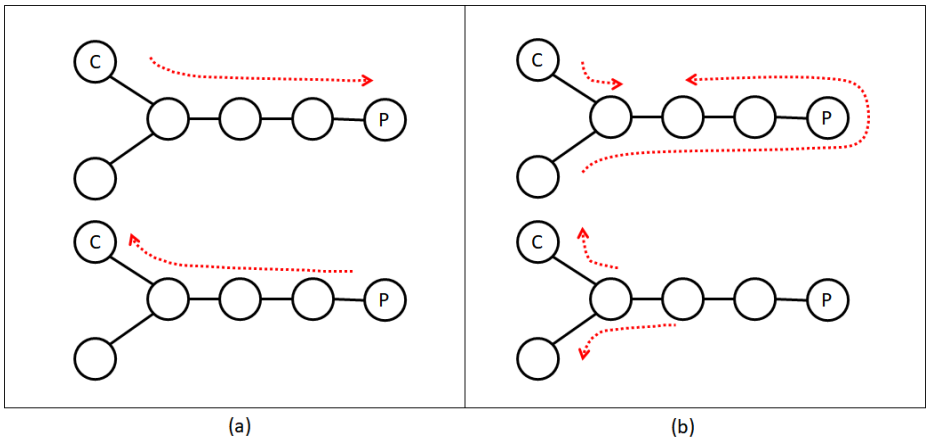


Figure 7.2: Two scenarios for sending an interest.

it expires before the data can be received.

Additionally, even if the advertisement leap-limit is set relative to the data lifetime and interest lifetime such that nodes are never over- or under-informed, data at the extremes of the network will have to constantly be refreshing their data since it will time out very shortly after arriving. If multiple nodes are in each region, this will produce the "pulsing" discussed in Section 4.1 and shown in Figure 4.8 (page 53).

In certain application areas, there may be *no understandable relationship* between NDN leaps and data or between NDN leaps and interest in the data, such as a network providing music services. In these cases, there is likely no way to avoid the over-informed or under-informed problems mentioned above. Measuring advertisements with lifetimes (rather than leap-limits) is not sensible in a DTN environment because of the common issue discussed throughout this work: the shortest path at any given time in a DTN is not the shortest path at all times in a DTN. Therefore, one cannot take a snapshot of message delivery times during the data advertisement period and assume it represents the network throughout its lifetime.

While the above discussion makes it seem nearly impossible to set data and interest

lifetimes without having over- or under-informed nodes, there are some convenient simplifications that should be noted. First, if the data is relevant to a certain geographic area, then the data lifetime can be measured/translated into leaps rather than time. For example, a temporary emergency medical center is likely interested in serving a local community rather than an entire city, so they may want their information kept up to date locally, while advertising other information which changes less frequently to the larger community. Second, if the data lifetime is leap-relative, then then interest lifetime can also be translated in the same way for the same reason. This work shows examples of this in the experimental section of this chapter. Future applications built upon this framework would most likely want to provide NDN applications with tools for choosing between the data expiration times and the geographical area they want to serve if the data is geo-region dependent.

7.1.2 Handling Mass Movement

In Chapter 4 the movement of agents was relatively regular such that regions of the world never became completely devoid of agents with devices. However, in disaster scenarios it is reasonable the evacuations and other mass movements of the population would cause geo-regions of the world to become depopulated ("bare"). This is especially true if geo-regions are based on some human defined boundaries, such as the building boundaries, which may become hazardous or destroyed. Similarly, if regions of the world can become bare, then it makes sense that regions might repopulate as events change.

Therefore, with a city-scale network, the geo-region architecture needs to be able to remove routes through essentially "missing" regions and also add new regions as they appear. Luckily, if the first task is accomplished, the second will already be handed using the announcement exchange mechanisms. There are three options for handling empty regions that *previously* had nodes: (1) ignore/PIT-timeout, (2) push, (3) poll-timeout.

To help explain these three mechanisms, Figure 7.3 shows an example where a consumer in geo-region (C) get a timeout after sending to the to the producer's geo-region (P). Figure 7.3(a) shows the original path which was previously working, while (b) shows a single area

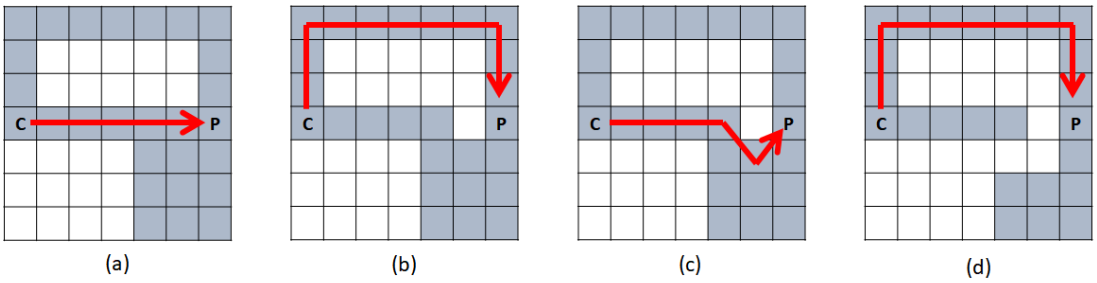


Figure 7.3: Example of a geo-region where one region is removed along a path.

becoming bare and a reroute at the consumer, and (c) shows a single area becoming bare and handled locally. Figure 7.3(d) shows multiple areas becoming bare and the reroute happening at the consumer.

The first option, (1) ignore/PIT-timeout, ignores the fact that regions have become empty and messages are sent along the original path and out "into the ether" by the network (they are sent into the bare region). In this case timeouts on interests in the PIT inform the network that the route chosen has timed out and regions may choose to update their paths. The down side of this is that regions are unaware of *where* the path is broken and they don't know *why* the path has timed out. In this case, if region (C) got a timeout, it may try to route around by sending north, as in Figure 7.3(b). This is not the optimal work around, since the path only has a small break that can be repaired further east.

The second option, (2) push, a technique where the last node leaving a geo-region pushes a notification out to adjacent regions before it leaves, basically declaring that the region is going bare. This allows neighboring regions to remove and FIB and PIT entries related to this region and resend interest pending in the PIT along a different route. This can potentially repair the network before regions further away are aware of the problem, and produces a repair that looks like Figure 7.3(c). However, this mechanism only works if geo-regions are less than half the communication range of nodes at their widest point (e.g. corner to corner for a grid), so that a single node leaving one geo-region can always contact a single node in a neighboring region. This also assumes that there are no misbehaving nodes

or nodes which "die" leaving the network bare without traveling to another geo-region. Violation of either of these requirements falls back to the first option where PIT timeouts control rerouting.

The third option, (3) poll-timeout, requires that nodes within communication range of a neighboring geo-region report in when they sense nodes in that region. This allows the intra-region communication to decide that a region has gone bare and not wait on push notifications, thus this is another solution that produces Figure 7.3(c). Compared with option (2), this allows geo-regions to be much larger, as long as the intra-region communication is (relatively) timely, and it can also assist in situations where geo-regions are not actually bare, but communication between the two regions has become impossible (e.g. if some sort of barrier gets erected between two regions). Again, the fallback for this is PIT timeouts.

Option two and three are useful for helping to repair *minor* fluxes in the distribution of nodes within geo-regions, but if major network upheavals occur, the fallback to option (1) is still important as shown in Figure 7.3(d) where the network has possible local work-around to the east of the consumer's geo-region. While the later experimental sections of this chapter do not simulate the intra-region communication, they do assume that option two or option three is being used such that regions neighboring bare regions are able to update their PIT and FIB entries accordingly.

7.1.3 Interplay with the Content Store

As has been discussed in Section 2.2, most ICNs provide some sort of caching function and NDN's version of this is the Content Store (CS). The previous discussions in this chapter have assumed that this functionality is used to serve data to nodes within a geo-region, but not to serve nodes from neighboring geo-regions, or that the CS is disabled entirely but nodes can share data using their intra-region communication. Enabling the CS in NDN to serve neighboring regions needs to be done carefully, however, as it adds complexity to the discussions from earlier.

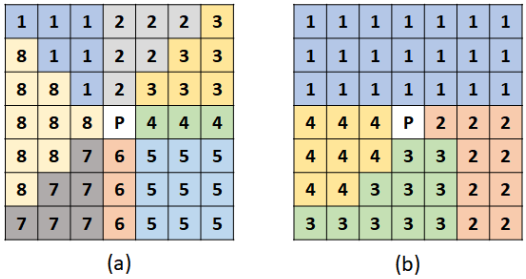


Figure 7.4: Possible data-copy distributions for two different neighborhood types.

First, if the CS is on, nodes can generally assume that data might be cached between the current location and the data producer, so the interest lifetimes can be shortened relative to the data lifetimes. This in turn means that there are shorter timeouts on PIT entries and rerouting around severely damaged networks can happen faster. At the same time, it introduces an interesting issue for geo-regions where all paths to a data producer go through a small set of geo-regions: only a small number of copies are produced and served to the entire network.

Figure 7.4 shows one *possible* picture of the data distribution when the CS is used to serve neighboring geo-regions for (a) Moore neighborhoods and (b) von Neumann neighborhoods. Numbers indicate the particular copy of the data received. If each copy of the data is produced at a similar time, then the entire network's data will tend to expire at the same time, causing many geo-regions to suddenly request new data. This is same behavior that can be seen with node-faces when a producer is behind a single intermediate node as shown in Figure 4.6(f-g) on page 52 and Figure 4.7(h-i) on page 53.

Because of this interplay, it is important to know if the CS will be serving neighboring regions or only nodes within a given geo-region so that interest lifetimes can be set appropriately. Additionally, if sudden drops in fresh data are not acceptable in the network, then the CS will need to be off or another mechanism explored for avoiding the bottleneck that produces a limited number of copies.

7.2 Geo-Region Architecture in a City-Scale ABM

In the introduction to this chapter the challenge of city-scale simulations was introduced, and in this section I will discuss the integration of the geo-region architecture with MASON (see Section 2.6) and the specific assumptions which were necessary to do so. As will be discussed in the experimental results section, I chose to integrate the geo-region architecture with an existing ABM model in MASON designed to use GIS and Volunteered Geographic Information (VGI) data for a disaster scenario in Haiti. The simulation has 1.3 million agents and covers a geographical region of roughly 8km by 6km area of Port-au-Prince [76].

7.2.1 Simplifying the Network Model for City-Scale Simulation

To scale to city-level, the geo-region architecture has an advantage over other face-type models in that it can be greatly simplified with a single assumption: the nodes within a region work together and act as a unit when sending/receiving message and that messages are sent between regions at fixed intervals. This is a design trade off which allows simulating the 1.3 million devices in Haiti as only tens of thousands of regions, and creates substantially fewer interactions between regions (the exact number depends on the size of the geo-region chosen). This makes the problem much more tractable, but does introduce some side effects that need to be examined closely.

First, this assumption obviously means that all nodes within a geo-region are capable of coordinating, for example, they have setup a WiFi Direct network and have an appropriate consensus algorithm in place for conflicting PIT, FIB, and CS entries. Loosely coordinated networks are possible, but a strongly coordinated group will more closely mimic the assumption above.

Second, this also means that geo-regions will perform the same regardless of how many devices are actually within a region. This is likely to slows down communication in well-populated areas (which are likely less disrupted), and speed up the communication in sparsely populated areas (which are likely more disrupted). As an example of this, in

reality a single node performing store-and carry across a geo-region would traverse the network slower than a network message jumping from device to device in a well-populated region.

Third, this assumption ignores the fact that it is not possible to *guess* the relative speed of agent movement to message movement if the agent movement is not controllable. One might think that the *slowest* a message could traverse a geo-region would be the walking speed of agents, but that is not the case as there is nothing preventing a device from being walked in circles at the center of a large geo-region and never getting near any neighboring geo-region to pass messages. So while the general behavior of the network can be simulated, the speed relative to agent movement is just a *guess* even in the best of circumstances.

Therefore, the assumption is enforcing a type of geo-region heterogeneity that might or might not exist in the network and a speed of transmission between regions which must be chosen by the simulation designer.

7.2.2 Specifics of Integration

MASON is written in Java, as is ONE, so the code from Chapter 4 was adapted to work with MASON. The PIT code was adapted almost unchanged with a few integration hooks. However, the FIB was dramatically simplified for the geo-region architecture rather than the general structure used to support multiple face types (such as node-faces, geo-direction faces, and so forth). Additionally, the CS was greatly simplified to only store a small number of data values and faster lookup. These simplifications of the FIB and CS were necessary to speed up the simulation and dedicating sufficient resources for properly simulating the PIT which is the core of the NDN protocol.

MASON allows for significant parameterization of its simulations and many parameters were added during this integration. Relevant to this section, the agent speed and network speed (in simulation steps) were added as parameters, as well as the network scale (the number of MASON grid spaces a geo-region is across). Figure 7.5 shows a screen shot of the interface for the simulation parameters where the agents speed and the network speed

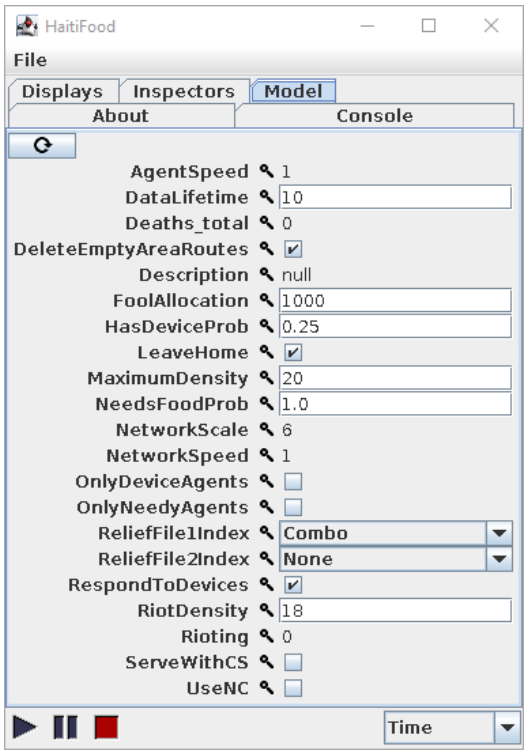


Figure 7.5: MASON simulation parameter interface for geo-region network integration.

are both set to one, meaning that agents move and evaluate their decisions every other step (1 step pause between decisions) and the inter-area messages are passed at each step. The network scale in this example is set to 6 creating a 6x6 geo-region made out of 36 MASON grid-spaces. For comparison, up to 20 agents can be on one MASON grid square at a time.

7.3 Experimental Evaluation

The interactions between parameters, difficulties with mass-movement, effect of the CS, and the basic challenges of combining the network model and ABM models discussed in the previous sections were uncovered by the experimental evaluation discussed in this section. In this section I detail the experimental evaluation that lead to an understanding of these challenges and provide additional commentary on the results of these experiments. It is important to mention here that the goal of this simulation was *not* to solve the food crisis

in Haiti, which is the disaster being simulated, but rather to study the ICDTN performance at scale.

The original Haiti model was designed to show the integration of GIS with ABMs by integrating VGI to assist in post-disaster relief [76]. The original authors imported road information and population information from publicly available sources, and then integrated volunteered damage information to inform their model. After a disaster, people (agents) attempt to maximize their energy by acquiring food if they think they are able to. Their decisions are based on their current energy levels and word-of-mouth knowledge about emergency centers. See [76] for more detailed information on this model; I will only outline *changes* to the model in the following section.

7.3.1 Experimental Setup

This section outlines the changes and parameters used in the experiments with this simulation. It is important to keep in mind while reading this section that the goal was *not* to solve the food distribution problem in Haiti, but rather to study a city-scale simulation and observe the scaling of the geo-region architecture along with other protocols outlined in my research.

Emergency Centers. The emergency center number and placement were chosen to be interesting from a network perspective while essentially sampling the "good" and "bad" placement from the original model. To this end, 8 were used instead of the 1 or 4 from the original paper, centers were in a mix of well populated and underpopulated areas, and two centers were placed very close to each other.

Agent Behaviors. In the original model, agents would travel towards food they became aware of through the rumor system; agents near emergency centers could observe the emergency centers, then agents in the same and neighboring MASON gird square exchanged information about known centers every two steps [1]. Emergency centers could run out of

[1]The storage of this information is a good example of one of many optimizations put in place for the simulation of millions of agents. A single integer in Java represents the entire knowledge of centers, where in integer is acting as an array of boolean bits for known/not-known status. This allows the simulation to use bitwise operations to drastically improve the information exchange step. Small implementation details

food or become so crowded that they would be unable to serve food, an occurrence observed in real life aid efforts.

An ad hoc ICDTN could obviously carry information about emergency centers as well, but in the interest of observing changes to the network as agents move around, some additional information was added to the model: whether an emergency center was closed. This allows agents to travel towards emergency centers for a period, but also to return to their homes if they are disappointed (in the original model agents just continued trying to "press in" towards a closed center). Emergency centers decide to close under the following situations: (1) there is no food remaining at the center, (2) there is such intense overcrowding that centers are unable to serve food, (3) no agents have requested food for an extended period of time. Overcrowding and undercrowding closing was set to 10 steps multiplied by the agent speed, which was 1 in this case, so that 10 consecutive steps without being able to serve food, for whatever reason, triggered the center to close.

Agents could become aware of closed centers and would go home if they were not aware of any centers that were open. Agents became aware of closings: (1) if they went to a closed center, (2) their network informed them, (3) the rumor system, or (4) a combination of the previous items. For example, one agent might become aware of the center closing from their network device, then tell his neighbor that information.

Announcements. To give a rough estimate of the impact of announcement protocols discussed in Chapter 5 and 6, the geo-regions were limited to sending one announcement message to each neighboring region per time step using either the "Smart Flooding" or "Network Coding" protocols discussed in Chapters 5 and 6.

Because there was no reasonable way to implement full network coding protocols for a simulation of this scale, the following approximation was used: if an encoded packet containing a completely new announcement a, a counter is set at that region to the number of announcements mixed with a, each subsequent mix of announcements containing a subtracts one from the timer, when the timer reaches 0 a is considered to be decoded. This

like this are essential to this scale of simulation.

is an approximation based on the knowledge that you will need 10 messages to guarantee decoding of any one message if you mix the same 10 messages together and send them with network coding. Since geo-regions are not mobile and can send announcements each round, they are unlikely to increase the number of messages mixed in at any given step (unless the centers were extremely close together in the city, which they are not). So the first encoded message is most likely to be the "most densely coded" a region is likely to receive.

One "Step" of the Network. At each interval the network is able to simulate certain behaviors. In short, each geo-region network determines the announcement messages it will send, the outgoing data messages it needs to continue, and the outgoing interests it needs to generate or continue. These three actions happen independently to replicate the behaviors of multiple loosely coordinated nodes working together as shown in Figure 4.8 on page 53. After all messages have been determined the messages are exchanged but no new messages are generated (as they would be in the traditional NDN pipeline); this avoids messages propagating across the network in one fell swoop.

Network Parameters. The following network-related parameters were variable in the simulation:

- Data Lifetime (In Steps) - How long a data response is valid after it is issued. The default was set to 10. Note: once a center is closed, the data announced has an expiration of "the end of time" since centers cannot reopen.
- Interest Lifetime (In Steps) - How long an interest was valid for after it was issued. The default was set to two times the data lifetime if the CS was turned off and 3 if the CS was on.
- Announcement Lifetime (In NDN Leaps) - How far an interest could travel from the producer. Since data and interest messages can move one NDN leap each time step this matches the Data Lifetime (the consequence of which is discussed earlier in this chapter).
- Delete Route (Boolean) - Whether or not geo-regions delete PIT and FIB entries for

neighboring bare regions. The default was True.

- CS Service (Boolean) - Whether or not the CS was used to serve neighboring geo-regions. If not, the CS still serves the nodes *within* the geo-region. Default was False.

- NC Announcements (Boolean) - Whether or not network coding was used. If not, the Smart Flooding protocol was used instead. The default was False.

- Network Speed (In Steps) - How often the network is allowed to take a step.

Population and Center Parameters. The following parameters pertain to the population of the city or the emergency centers.

- Device Probability (Percentage) - When agents are initialized, the probability that they have a network device. This is important for some semblance of reality since it is unlikely for 1.3 million people to all carry the same device with software capable of running an ad hoc ICDTN. The default was set to 25% of the population.

- Needy Probability(Percentage) - When agents are initialized, the probability that they need food. Default set to 100%. This probability is independent of device probability, such that if both were set to 50%, approximately 25% of the population would have both a device and need food, 25% of the population would not have a device and not need food, and 50% of the population have a device *or* need food.

- Respond to Devices (Boolean) - Whether or not agents respond to their network devices. The default is True. This setting is used to explore network operations while not influencing the agent behaviors.

- Food Allocation (In Units of Food) - How much food each center has at the beginning of the simulation. The vanilla model implementation of the model uses 100, but the default here is set to 1000 which was the value used in the original paper to ensure enough interesting behavior is recorded. One person is satisfied with one unit of food.

- Step On Toes (Boolean) - In the original model, agents were limited to 20 agents per MASON grid square to study the effect of crowding. This parameter allows nearly infinite agents to occupy a grid square. Default set to False.

- Agent Speed (In Steps) - How many steps *in between* when agents (re)evaluate their actions and make decisions. The default is 1, so agents make decisions every other step.

Simulation Settings. The following were general properties of the simulation.

- Only Devices (Boolean) - Whether or not to simulate only agents with network devices. Default is set to False.

- Only Needy (Boolean) - Whether or not to simulate only agents who need food. Default is set to False.

All simulations were run once for 150 steps with a seed of 1 for reproducibility.

7.3.2 Experimental Results

Running the simulation, it is easy to see from the MASON visualization of the network and the agents that the geo-region network is performing as expected. Figure 7.6 shows two snapshots of the visualization at 10 and 50 steps in. Shaded areas indicate NDN geo-regions with fresh data. Emergency centers are large colored squares toward the center of the shaded areas. Smaller, bright red dots are MASON grid squares containing agents who are not at home, and believe they know about an open emergency center. Dark red dots indicate MASON grid squares containing agents who are not at home, and do not know about any open emergency centers. Agents at home are not shown.

At the beginning of the simulation, the network radiates out from the emergency centers in roughly a square pattern due to the Moore neighborhood configuration. When agents leave their houses, they cause many geo-regions to go bare so that there are a few gaps in the square area, eventually leading to network layouts that look like the networks around

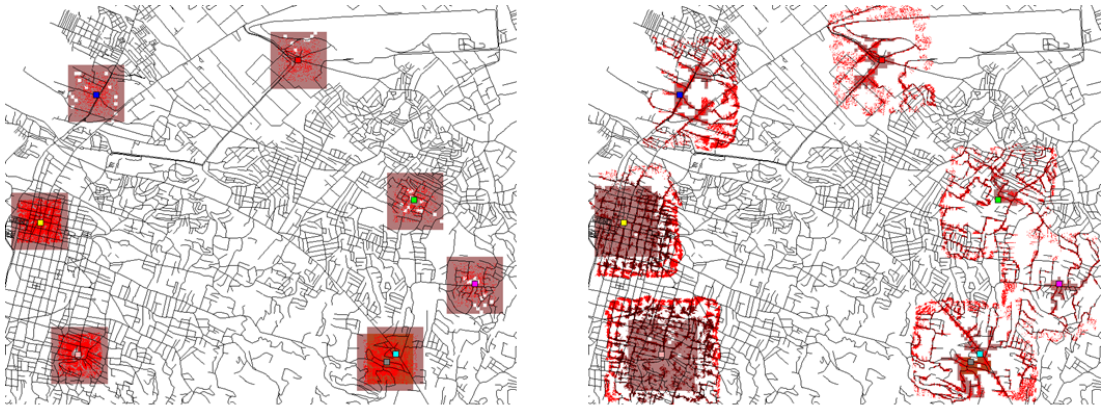


Figure 7.6: Two snapshots of the MASON visualization for the Haiti experiment.

the north-most and east-most emergency centers in Figure 7.6(b) where the network has reconfigured to pass data along the roads. However, when an emergency center shuts down, such as in the southwest most two centers, as agents return to their homes, the network travels back with them.

It can also be seen from this visualization that the rumor mill component stretches out beyond the network. Again, as we are not trying to solve the issue of food delivery in Haiti, this is merely an interesting observation needed to understand later results.

The remainder of this section will explore different simulation scenarios with commentary for understanding what the results mean for the city-scale scaling of ad hoc ICDTNs. This rounds off the "holistic approach" to these networks discussed in Chapter 1.

Base Cases and Agent Actions/Movement. There are three base cases for this section: Word-of-Mouth Only (data lifetime = 0, respond to devices = False), a Short-Range Network (data lifetime = 10), and a Long-Range Network (data lifetime = 40). These three types of networks are abbreviated as WoM, SR, and LR respectively in the remainder of this discussion. All other parameters are set to their default values. Figure 7.7 shows the agent activities for the three base cases.

There are some interesting things to note which will play a role later. First, in the WoM

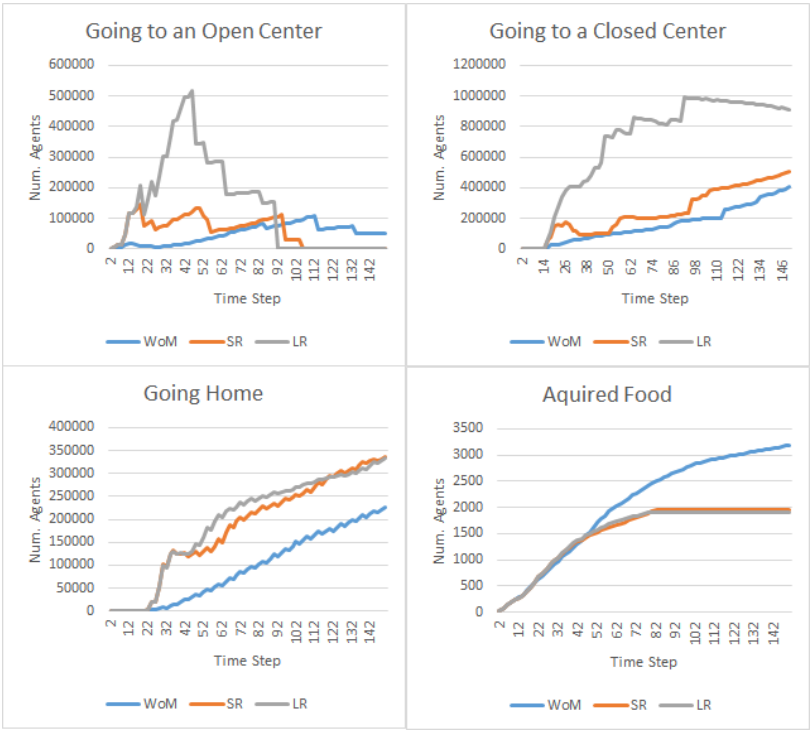


Figure 7.7: Base case Haiti simulation using a number of networks.

simulation, there are always fewer agents going to open centers than there are agents going to closed centers, except right at the beginning of the simulation. This is due to the slowness of the WoM system in getting information out to agents. This pattern is distinctly different compared to the SR and LR network patterns which have some interplay between these two since they can get information out faster. Eventually, all simulations have runaway rumor systems where agents get information about centers being open before the follow-up information that the center is closed, which is to be expected beyond the network range.

The second item of interest is that fewer agents are getting food in the SR and LR simulations, but the cause can be deduced by examining Figure 7.8 which displays emergency Center information for food levels (top), number of nearby agents (middle), and the numbers of centers closed (bottom) for the three base cases: WoM (left), SR (middle), and LR (right). The flat lines for center levels of food indicate that the center has closed. It is clear that more information about centers spreading through the network is causing excessive

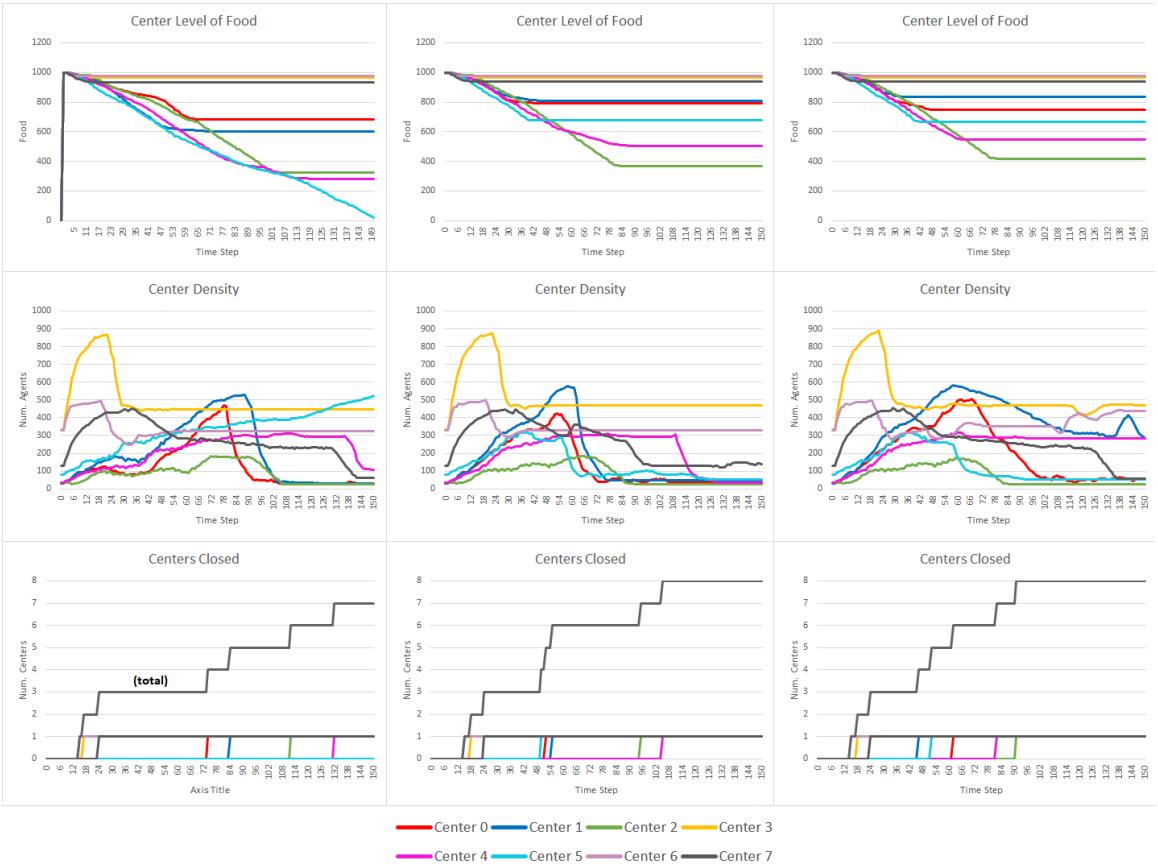


Figure 7.8: Emergency center information for base case Haiti simulation.

crowding near the centers.

As a "sanity check" that overcrowding is the main cause of reduced food delivery, Figure 7.9 shows what happens when only 10% of the network needs food (needy probability = 10%). In this case the centers remain open for longer, more food is distributed, and the network helps spread the information so that the agents acquire more food.

Network Performance Examination and Timeouts Relationships. The previous section was important to understand what was happening at the agent level so that the network actions are more understandable. Figures 7.10-7.11 show various information of interest about the SR and LR networks throughout the simulation. Figure 7.10 examines the number of agents "covered" by a geo-region with information (top) and number of geo-regions with

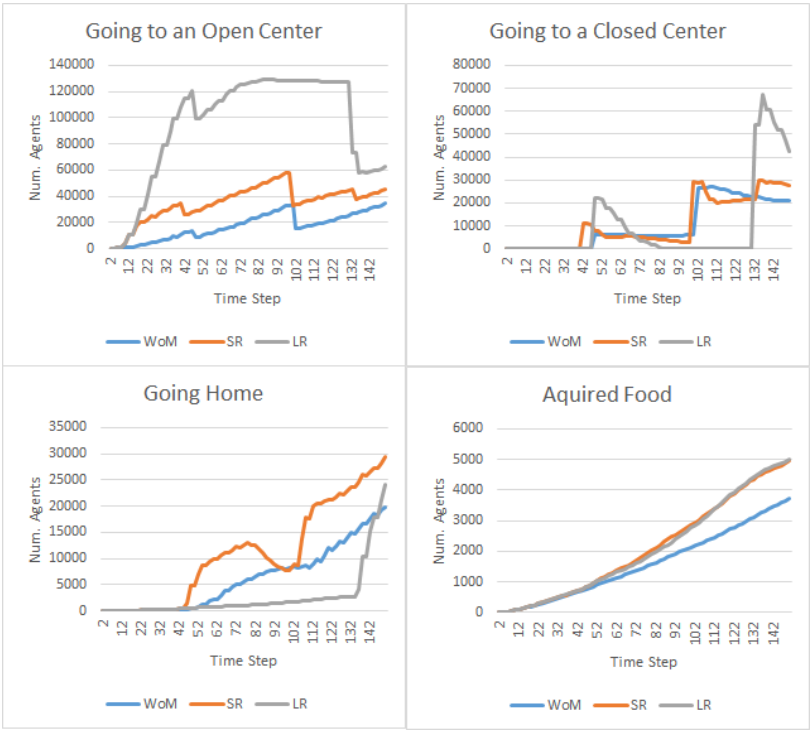


Figure 7.9: Agent and center information when the number of needy agents is only 10%.

information (bottom) where the information is a decoded announcement (left), any data (middle), and fresh data that has not expired (left) in the Short-Range Base Case simulation. While Figure 7.11 shows the same for the long range network. Note that the scale of these charts is drastically different.

It is clear from these results that the same numbers of agents are being "covered" by the network throughout the simulation, the "dip" in geographic regions with information is not matched by the number of agents, so this means that the network is still serving approximately the same number of people but that the people have a different geo-region configuration. Additionally, in the SR network, the data expires sooner (after 10 steps), so we see a big dip when the first data messages sent out expire while in the LR network the data expires later (after 40 steps). We can also clearly see that the the LR network has trouble "recovering" it's fresh data at first. This is exactly the problem discussed earlier, with the long interest lifetime preventing the network from recovering from drastic changes.

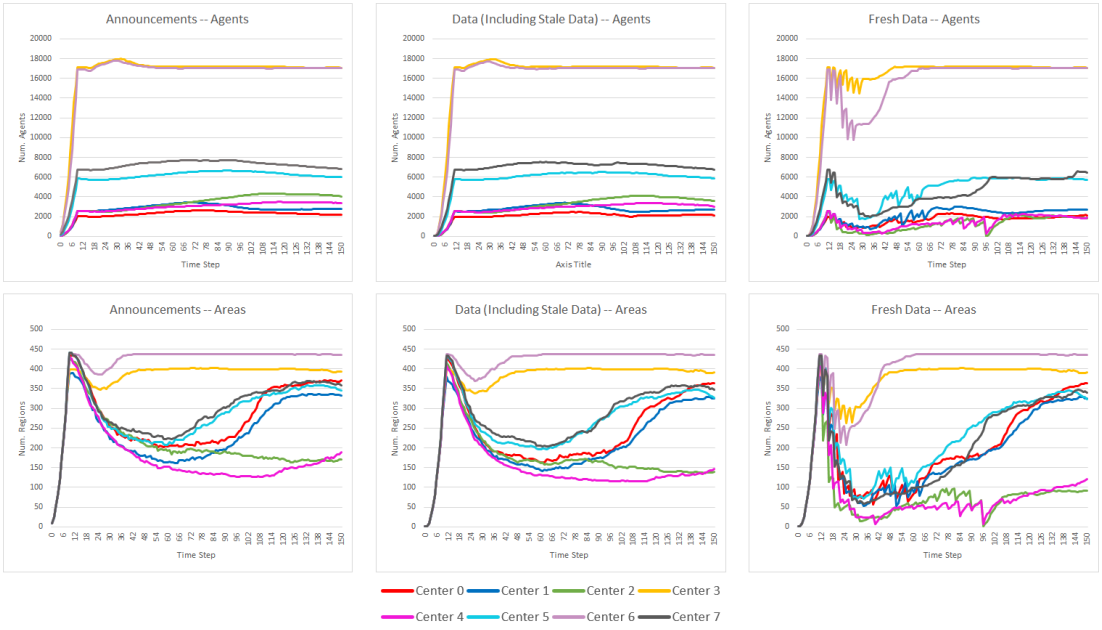


Figure 7.10: Examination of geo-region information with short range network.

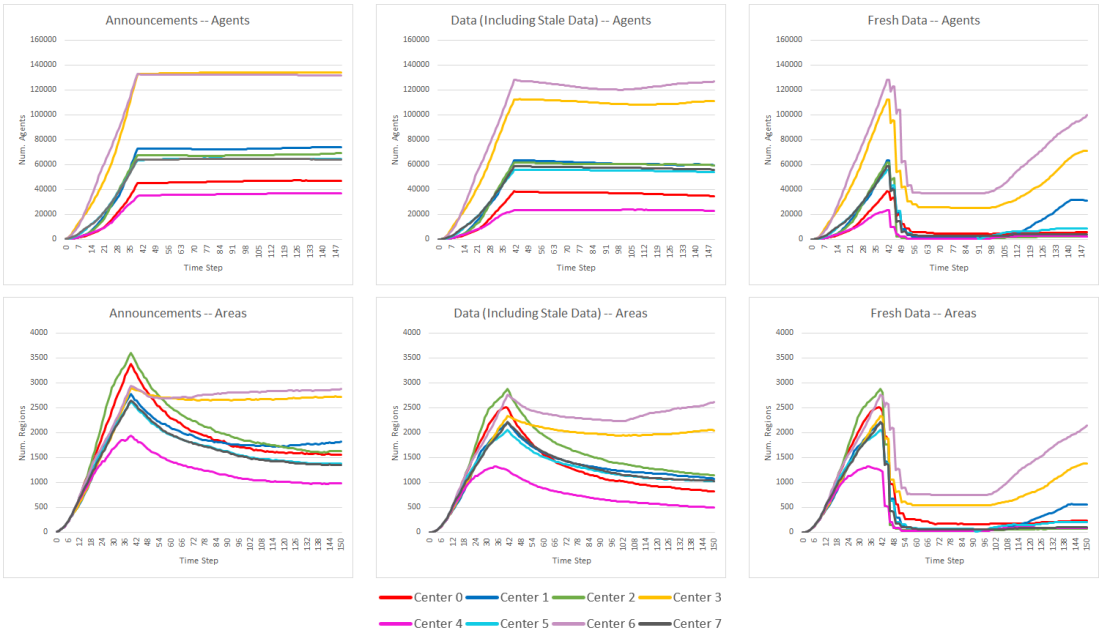


Figure 7.11: Examination of geo-region information with long range network.

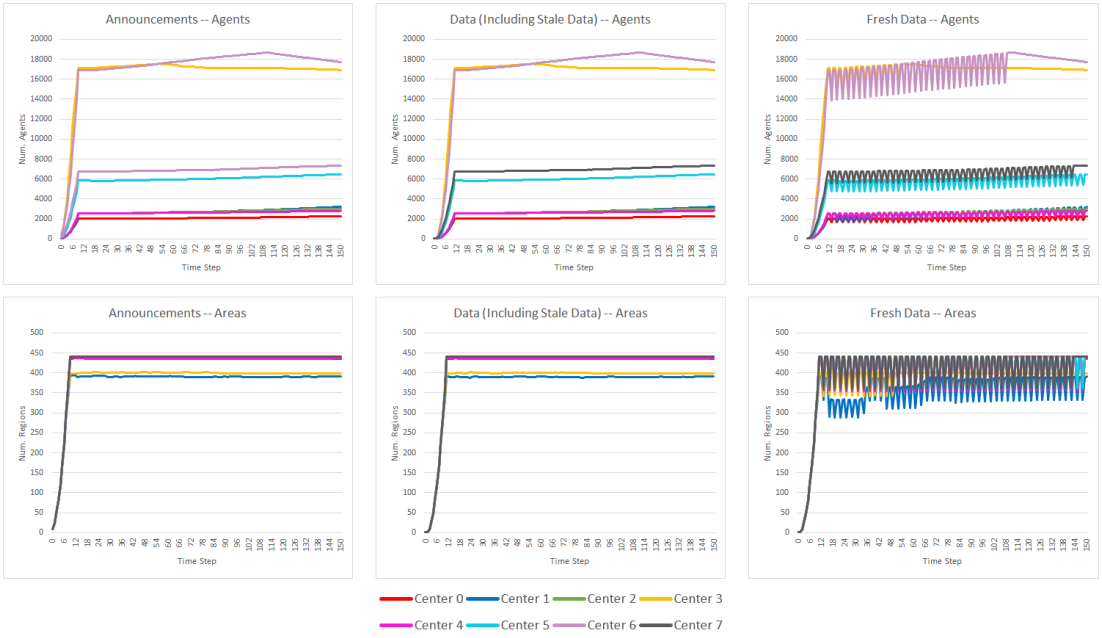


Figure 7.12: Geo-region information with short range network and 10% of the agents needy.

Another item of interest is that there is a pronounced "jitter" when the data is regularly expiring in the SR network. This was predicted by the discussions earlier related to uncoordinated pipelines in Section 4.1. This jitter is never seen when the data expiration is set to "the end of time" after a center closes.

In the simulation, if only 10% of the network needs food and the other 90% stay at home, this builds a relatively stable network that does not need to adapt much to the human movement (only a few geo-regions become bare). Figure fig:SimulationNeedy10-SR show what happens in the SR network (the LR network performs similarly). The jitter in particularly noticeable in this scenario since only data expiration causes an entire region to lose data (compared with expiration and regions becoming bare).

Examining a Smaller Population for More Disruptive Agent Movement. The next area of interest would be even sparser networks without a "stable" population. For this, the "only needy" parameter was set to True and the needy probability was again dropped to 10%. The results of this simulation are shown in Figure 7.13-7.14 at this point, the network

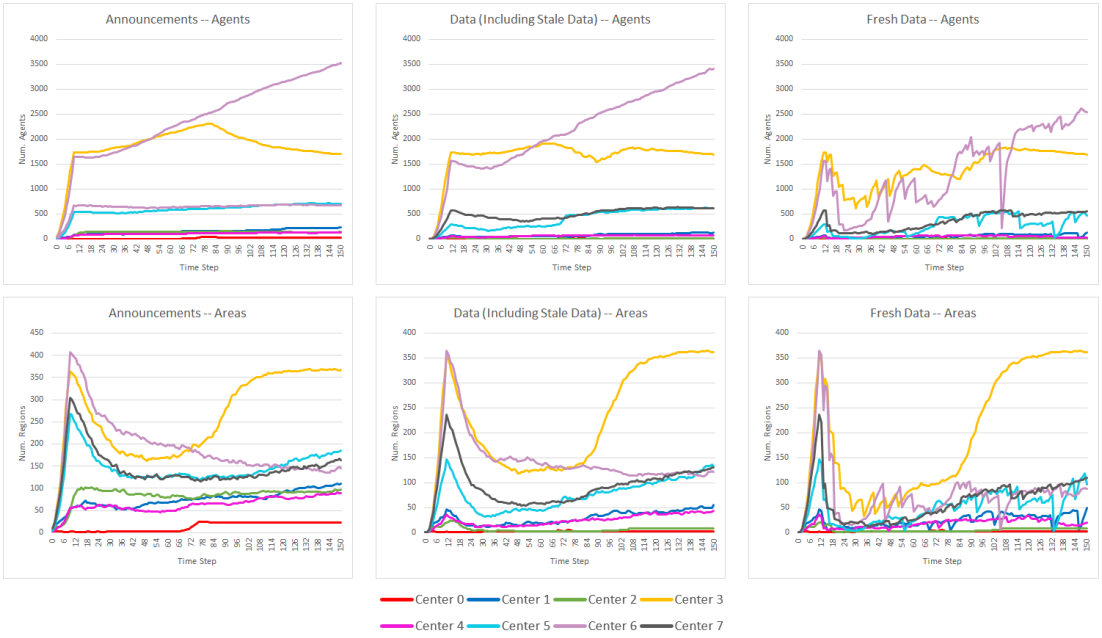


Figure 7.13: Examination of short range network maintained by only 10% of the population.

is sparse enough that it has trouble maintaining fresh data in the network and does not "bounce back" as quickly. This is likely the limit of networks such as this and larger geo-regions would need to be used to handle the sparseness of the agents. This, as discussed earlier in this section, would have substantial trade-offs and limitations depending on the communication range of agents due to the requirements on an intra-region communication mechanism.

Interplay with the Content Store. The next major item of interest concerns what happens when the CS is used to serve other geo-regions. Figures 7.15-7.15 show the network information for the SR and LR networks. As discussed, the bottlenecks are created producing only a few data copies out into the network. This results in regular significant drops in the amount of fresh data available (which again stops when the emergency centers close and have a much longer lifetime). We can see, especially in the SR network how the information is repeatedly redistributed successfully, however, after these dips. For the LR network, there is still a much slower ability to recover, but it is starting much earlier, since the interest

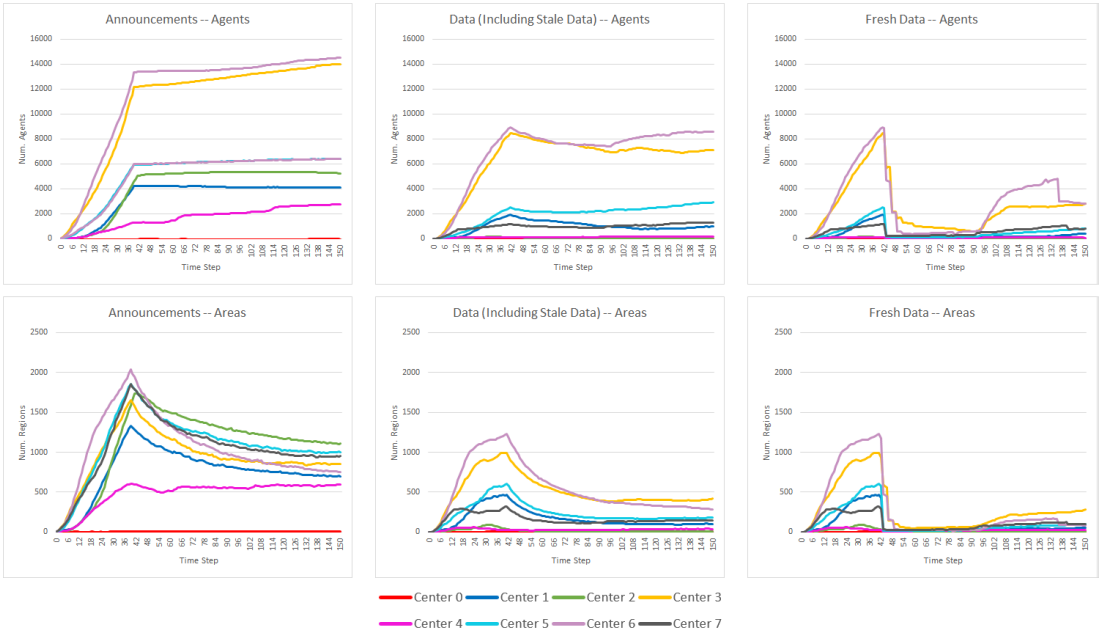


Figure 7.14: Examination of long range network maintained by only 10% of the population.

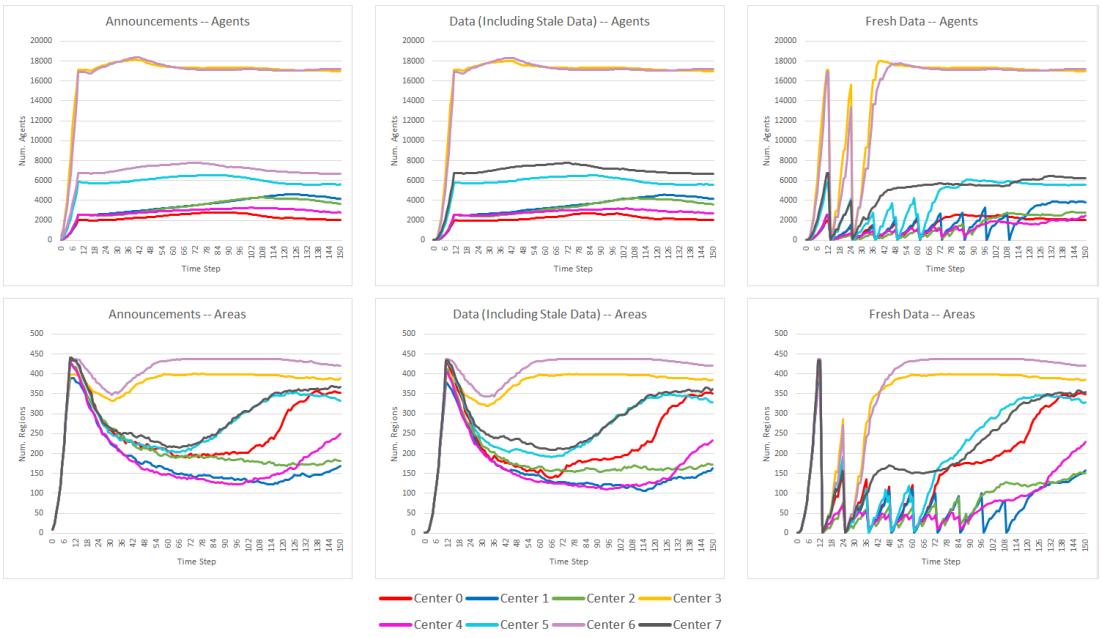


Figure 7.15: Examination of scenario with content store enabled for short range network.

lifetime is able to be set much shorter.

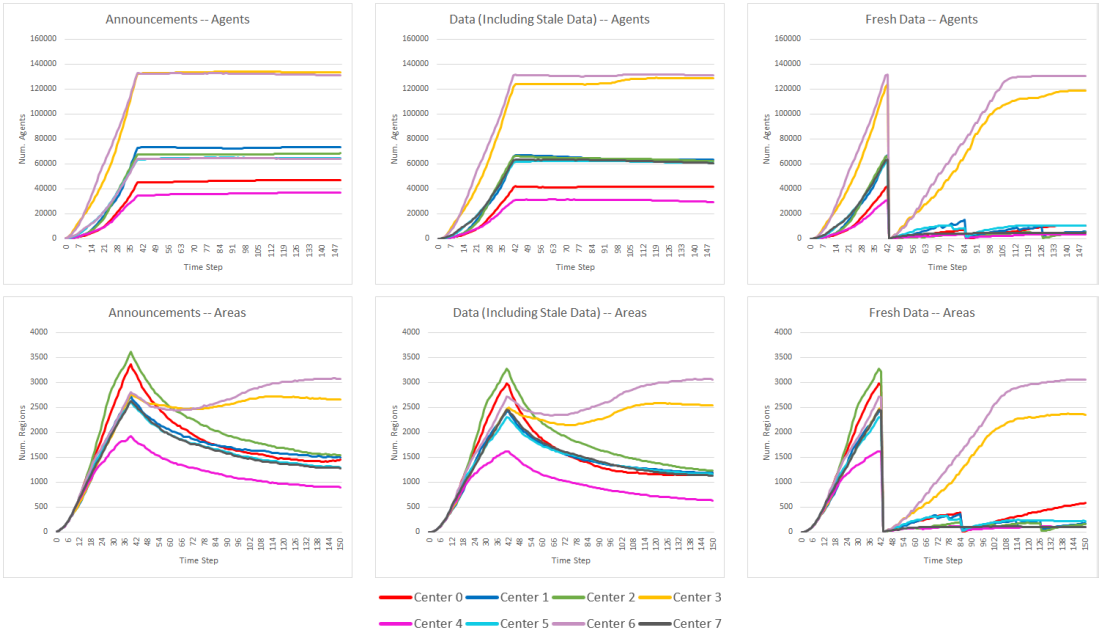


Figure 7.16: Examination of scenario with content store enabled for long range network

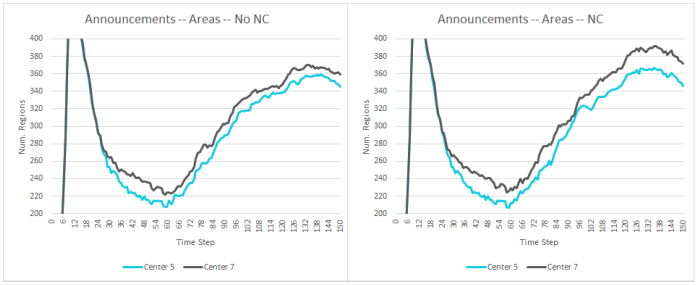


Figure 7.17: Examination of scenario with network coding announcement enabled.

Network Coding. Massive scaling hides much of the local benefits of network coding. However, we can observe, as shown in Figure 7.17 that the two centers very near each other are able to "fill in" re-established bare regions more quickly after a dip. The left figure shows the announcements without network coding and the right figure shows the announcements with network coding for an identical experiment.

7.4 Conclusion

In this chapter I have shown that the geo-region architecture proposed in Chapter 4 and the data advertisement protocols discussed throughout this work, can be used to create city-scale ICDTNs for disaster relief. I believe strongly that these networks will soon be a viable solution to infrastructure collapse, and this chapter has attempted to move to research into city-size ICDTNs to the next step by specifically addressing challenges of scale. In addition, by integrating with agent-based models and using an abstraction of the geo-region architecture, large scale simulations can be studied and analyzed before committing substantial resources to hardware and test-bed implementations.

Chapter 8: Conclusion and Future Work

Given that existing communication infrastructure can be unavailable during and after natural disasters, and even relatively common situations like power outages and large social gatherings, it has been suggested that small pocket-switch networks might be able to continue communication by forming an ad hoc network. Much research has been done on applying DTN and ICN techniques to the problem, and recently a new area of research has gained momentum: ICDTNs which combine the two technologies.

With the abundance of proposals for ICDTN architectures, it is apparent that, while there is much interest in the subject, the two architectures are fundamentally different. This work has given a simple solution to this challenge using a small middle layer between an ICN and DTN layer, allowing both existing frameworks to contribute to the solution without heavy modification of either protocol. This allows for a "best of both worlds" scenario where the ICN guides packets towards the desired data while the DTN handles disruption at the lower level.

This middle layer also allows a "tailored" solution for different disruption scenarios by allowing different middle layers to abstract away details of the underlying disruption. This work demonstrated that human movement could be handed with a geographically-aware middle layer, while more regular, predictable movement of devices can be handed with a node/device-aware middle layer. The scalability of the geo-region architecture was demonstrated in Chapters 4 and 7 for networks containing a thousand to over a million devices. Large, city-scale networks offer their own challenges, which were addressed throughout this work, and several unique ones relating to metro-scale disaster relief have been discussed and addressed in Chapter 7.

In addition to the above contributions, this work proposed and evaluated several protocols for the data advertisement problem, which is a foundational initialization step for ad

hoc ICDTNs. Previous work in this area has not addressed this problem, choosing to focus instead on performance after the network is established. Extensive work was done in this area with the hope of establishing real, implementable ICDTNs from the ground-up rather than assuming existing communication architectures managed the network initialization.

In short, this dissertation contributes many of the foundational components for creating ad hoc ICDTNs from the ground up. The next step of this research *must* focus on network security. Related to this I have relied on two assumptions in work which require additional investigation: the use of homomorphic signatures for mixing packets and the verification/validation of data and advertisements. For the first, I have stated that either intermediate nodes must be trusted or homomorphic signatures are used when mixing. This is a heavy burden on intermediate nodes who must either have an ad hoc trust scheme or be capable of creating and mixing signatures. At this point, these may not be reasonable burdens to place on small pocket-size devices which have limited space, computational power, and energy.

For the second item, the signatures on a given data advertisement and/or created piece of data in the network cannot be verified without some sort of centralized control. I have assumed in this work that some form of secure key distribution has been used, either by building keys into the hardware of the devices or by traditional key distribution mechanisms used prior to the ICDTN creation. This assumption is unfortunate, as I would like to have a fully "holistic" approach to ICDTN creation with security built in as a first principle.

Blockchain-based algorithms might be a good fit for this type problem, since they can be fully distributed, however the challenges of operating a blockchain in a heavily disrupted environment with small pocket-size devices does not appear to be currently feasible. There are two major challenges: first, a portion of the blockchain algorithm relies on waiting for a reasonable amount of local consensus, if this is scaled to accommodate the wait times for a DTN, this would prevent initializing networks quickly enough to be useful. Second, the storage and energy requirements of running a blockchain algorithm on the network devices is likely not realistic. The light node solutions currently proposed for IoT cannot be directly

applied since there is no assurance that *any* devices in the network would be capable of being full nodes (with the full blockchain), let alone enough to be locally useful.

There are also some additional items of future work I'd like to mention here. First, I would like to further investigate the density parameter heuristic proposed in Section 5.3. While I have shown that empirically this works well, I think a more formal assessment needs to be performed. Second, I think there could be interesting work in prioritizing data requests, such as for requests coming from emergency workers or the distribution of high impact data. This, I think, might be managed by providing an additional high-priority queue at the DTN layer to allow sending these messages first when contact times are particularly short. Third, there has been a lot of work on caching in ICN networks, but this work has shown only no-caching and all-caching models; limited caching and strategic caching strategies would be good to integrate into this architecture.

All this said, given the major contributions of this work, I hope that researchers can begin to consider city-scale implementations as a viable solution to infrastructure collapse. I firmly believe that the layering of ICN and DTN protocols is the next step forward for ICDTNs, that the geo-region approach is a simple and scalable approach to this problem, and that the foundational data advertisement stage can be addressed with the approaches given in this work.

Bibliography

Bibliography

[1] M. Hamblen, "'massive damage' from japan quake hits communications," *ComputerWorld*, 2011. [Online]. Available: https://www.computerworld.com/article/2506865/disaster-recovery/-massive-damage–from-japan-quake-hits-communications.html

[2] L. Campioni, M. Hauge, L. Landmark, N. Suri, and M. Tortonesi, "Considerations on the adoption of named data networking (ndn) in tactical environments," in *2019 International Conference on Military Communications and Information Systems (ICMCIS)*, 2019, pp. 1–8.

[3] C. Pogash and B. X. Chen, "California Blackouts Hit Cellphone Service, Fraying a Lifeline," *The New York Times*, 2019. [Online]. Available: https://www.nytimes.com/2019/10/28/business/energy-environment/california-cellular-blackout.html

[4] M. Reardon, "Women's march overwhelms mobile network in dc," *CNET*, 2017. [Online]. Available: https://www.cnet.com/news/womens-march-overwhelms-mobile-network-in-dc/

[5] F. C. Commission *et al.*, "Wireless E911 location accuracy requirements," *PS Docket*, no. 07-114, 2015.

[6] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-tolerant networking: an approach to interplanetary internet," *IEEE Communications Magazine*, vol. 41, no. 6, pp. 128–136, 2003.

[7] M. Conti, A. Gangwal, M. Hassan, C. Lal, and E. Losiouk, "The Road Ahead for Networking: A Survey on ICN-IP Coexistence Solutions," *arXiv:1903.07446 [cs]*, Mar. 2019, arXiv: 1903.07446.

[8] M. C. Domingo, "An overview of the internet of underwater things," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 1879–1890, Nov. 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804512001646

[9] M. Radenkovic, V. S. Ha Huynh, R. John, and P. Manzoni, "Enabling Real-time Communications and Services in Heterogeneous Networks of Drones and Vehicles," in *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2019, pp. 1–6, iSSN: 2160-4894.

[10] C. Sauer, M. Schmidt, and M. Sliskovic, "Delay Tolerant Networks in Industrial Applications," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2019, pp. 176–183, iSSN: 1946-0759.

[11] C. Tripp-Barba, A. Zaldvar-Colado, L. Urquiza-Aguiar, and J. A. Aguilar-Caldern, "Survey on Routing Protocols for Vehicular Ad Hoc Networks Based on Multimetrics," *Electronics*, vol. 8, no. 10, p. 1177, Oct. 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/10/1177

[12] S. Bounsiar, F. Z. Benhamida, A. Henni, D. L. d. Ipia, and D. C. Mansilla, "How to Enable Delay Tolerant Network Solutions for Internet of Things: From Taxonomy to Open Challenges," *Proceedings*, vol. 31, no. 1, p. 24, 2019. [Online]. Available: https://www.mdpi.com/2504-3900/31/1/24

[13] O. S. Oubbati, A. Lakas, F. Zhou, M. Gne, and M. B. Yagoubi, "A survey on position-based routing protocols for Flying Ad hoc Networks (FANETs)," *Vehicular Communications*, vol. 10, pp. 29–56, Oct. 2017.

[14] M. W. Kang, Y. Kim, and Y. W. Chung, "An opportunistic forwarding scheme for icn in disaster situations," in *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, 2017, pp. 948–950.

[15] G. Rizzo, S. Ristov, T. Fahringer, M. Gusev, M. Dzanko, I. Bilic, C. Esposito, and T. Braun, "Emergency networks for post-disaster scenarios," in *Guide to Disaster-Resilient Communication Networks*, ser. Computer Communications and Networks, J. Rak and D. Hutchison, Eds. Springer International Publishing, 2020, pp. 271–298.

[16] D. Trossen, A. Sathiaseelan, and J. Ott, "Towards an Information Centric Network Architecture for Universal Internet Access," *SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 1, pp. 44–49, Jan. 2016.

[17] C.-A. Sarros, S. Diamantopoulos, S. Rene, I. Psaras, A. Lertsinsrubtavee, C. Molina-Jimenez, P. Mendes, R. Sofia, A. Sathiaseelan, G. Pavlou, and others, "Connecting the Edges: A Universal, Mobile-Centric, and Opportunistic Communications Architecture," *IEEE Communication Magazine*, 2018.

[18] *Cisco Advances Open-Source Hybrid Information-Centric Networking for 5G*, 2019 (accessed July 1 2020). [Online]. Available: https://blogs.cisco.com/innovation/cisco-advances-open-source-hybrid-information-centric-networking-for-5g

[19] C. Fang, H. Yao, Z. Wang, W. Wu, X. Jin, and F. R. Yu, "A survey of mobile information-centric networking: Research issues and challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2353–2371, 2018.

[20] C. Li, H. Lu, Y. Xiang, and R. Gao, "Geo-dmp: A dtn-based mobile prototype for geospatial data retrieval," *ISPRS International Journal of Geo-Information*, vol. 9, no. 1, p. 8, 2020.

[21] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture," *Request for Comments*, 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc4838

[22] G. Grassi, D. Pesavento, G. Pau, L. Zhang, and S. Fdida, "Navigo: Interest forwarding by geolocations in vehicular Named Data Networking," in *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Jun. 2015, pp. 1–10.

[23] D. Li, T. Song, Y. Yang, and I. R. Ul, "A reliable and efficient forwarding strategy in vehicular named data networking," *Journal of Communications and Networks*, vol. 22, no. 4, pp. 348–357, 2020.

[24] Bundle protocol query extension block. [Online]. Available: https://tools.ietf.org/id/draft-farrell-dtnrg-bpq-00.html

[25] K. Russell and R. Simon, "Evaluation of a geo-region based architecture for information centric disruption tolerant networks," in *2019 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2019, pp. 911–917.

[26] ——, "Named data networking for scalable intermittently connected networks," 2019, named Data Networking Community Meeting, Gaithersburg, Maryland, USA.

[27] ——, "Efficient data advertisement in information centric disruption tolerant networks," in *2020 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2020, pp. 112–117.

[28] S. Batabyal and P. Bhaumik, "Mobility Models, Traces and Impact of Mobility on Opportunistic Routing Algorithms: A Survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1679–1707, 2015.

[29] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols," in *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, ser. MobiCom '98. New York, NY, USA: ACM, 1998, pp. 85–97.

[30] S. Kosta, A. Mei, and J. Stefa, "Large-Scale Synthetic Social Mobile Networks with SWIM," *IEEE Transactions on Mobile Computing*, vol. 13, no. 1, pp. 116–129, Jan. 2014.

[31] L. Becchetti, A. Clementi, F. Pasquale, G. Resta, P. Santi, and R. Silvestri, "Flooding time in opportunistic networks under power law and exponential intercontact times," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2297–2306, 2014.

[32] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 2, Mar. 2003, pp. 1312–1321 vol.2.

[33] C. Bettstetter, H. Hartenstein, and X. Prez-Costa, "Stochastic Properties of the Random Waypoint Mobility Model," *Wireless Networks*, vol. 10, no. 5, pp. 555–567, Sep. 2004.

[34] T. Osuki, K. Sakai, and S. Fukumoto, "Contact avoidance routing in delay tolerant networks," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[35] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell system technical journal*, vol. 39, no. 5, pp. 1253–1265, 1960.

[36] M. J. Khabbaz, C. M. Assi, and W. F. Fawaz, "Disruption-Tolerant Networking: A Comprehensive Survey on Recent Developments and Persisting Challenges," *IEEE Communications Surveys Tutorials*, vol. 14, no. 2, pp. 607–640, 2012.

[37] S. Cc, V. Raychoudhury, G. Marfia, and A. Singla, "A survey of routing and data dissemination in delay tolerant networks," *Journal of Network and Computer Applications*, vol. 67, pp. 128–146, 2016.

[38] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke University, Tech. Rep., 2000.

[39] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. ACM, 2005, pp. 252–259.

[40] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: modeling and analysis of a three-tier architecture for sparse sensor networks," *Ad Hoc Networks*, vol. 1, no. 2, pp. 215–233, Sep. 2003.

[41] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '00. New York, NY, USA: ACM, 2000, pp. 243–254.

[42] E. Kuiper and S. Nadjm-Tehrani, "Geographical Routing With Location Service in Intermittently Connected MANETs," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 2, pp. 592–604, Feb. 2011.

[43] V. N. G. J. Soares, J. J. P. C. Rodrigues, and F. Farahmand, "GeoSpray: A geographic routing protocol for vehicular delay-tolerant networks," *Information Fusion*, vol. 15, pp. 102–113, Jan. 2014.

[44] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[45] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. C. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014.

[46] Y. Zhang, Z. Xia, S. Mastorakis, and L. Zhang, "KITE: Producer Mobility Support in Named Data Networking," in *Proceedings of the 5th ACM Conference on Information-Centric Networking*, Boston, MA, 2018, p. 12.

[47] X. Mwangi and K. Sollins, "MNDN: Scalable Mobility Support in Named Data Networking," in *Proceedings of the 5th ACM Conference on Information-Centric Networking*, Boston, MA, 2018, p. 8.

[48] A. Arcia-Moret, I. Psaras, and J. Crowcroft, "Bringing Information Centric Networking to Challenged Environments: An overview of the Second Workshop on Future Internet Architecture for Developing Regions," *arXiv:1801.01824 [cs]*, Jan. 2018.

[49] H. M. A. Islam, A. Lukyanenko, S. Tarkoma, and A. Yla-Jaaski, "Towards disruption tolerant ICN," in *2015 IEEE Symposium on Computers and Communication (ISCC)*, Jul. 2015, pp. 212–219.

[50] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing Information Networking Further: From PSIRP to PURSUIT," in *Broadband Communications, Networks, and Systems*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, Berlin, Heidelberg, Oct. 2010, pp. 1–13.

[51] A. Afanasyev, X. Jiang, Y. Yu, J. Tan, Y. Xia, A. Mankin, and L. Zhang, "NDNS: A DNS-Like Name Service for NDN," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, Jul. 2017, pp. 1–9.

[52] A. K. M. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "NLSR: Named-data Link State Routing Protocol," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '13. New York, NY, USA: ACM, 2013, pp. 15–20.

[53] J. Shi, E. Newberry, and B. Zhang, "On broadcast-based self-learning in named data networking," in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, Jun. 2017, pp. 1–9.

[54] H. Ben Abraham, J. Parwatikar, J. DeHart, A. Drescher, and P. Crowley, "Decoupling Information and Connectivity via Information-Centric Transport," in *Computer Science and Engineering Publications and Presentations*, Boston, MA, Aug. 2018.

[55] A. Kulkatni and A. Seetharam, "QuickR: A novel routing strategy for wireless mobile information-centric networks," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, 2019, pp. 1–8.

[56] V. Sourlas, L. Tassiulas, I. Psaras, and G. Pavlou, "Information resilience through user-assisted caching in disruptive Content-Centric Networks," in *2015 IFIP Networking Conference*, May 2015, pp. 1–9.

[57] Y. Hayamizu, M. Yamamoto, and T. Yagyu, "Energy and Bandwidth Efficient Content Retrieval for Content Centric Networks in DTN Environment," in *2015 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2015, pp. 1–6.

[58] M. K. Kiskani and H. R. Sadjadpour, "A secure approach for caching contents in wireless ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 249–10 258, 2017.

[59] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, "Network Coding: An Instant Primer," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 63–68, Jan. 2006.

[60] S. Deb, M. Medard, and C. Choute, "Algebraic Gossip: A Network Coding Approach to Optimal Multiple Rumor Mongering," *IEEE/ACM Trans. Netw.*, vol. 14, no. SI, pp. 2486–2507, Jun. 2006.

[61] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *Proceedings 41st Annual Symposium on Foundations of Computer Science*, 2000, pp. 565–574.

[62] F. Chierichetti, G. Giakkoupis, S. Lattanzi, and A. Panconesi, "Rumor spreading and conductance," *J. ACM*, vol. 65, no. 4, Apr. 2018.

[63] A. Bennakhi, A. B. Nakhi, and A. Marafi, "Context-aware gossip in ad hoc vehicular networks," *Procedia Computer Science*, vol. 160, pp. 157 – 164, 2019.

[64] A. Cohen, A. Cohen, M. Mdard, and O. Gurewitz, "Secure multi-source multicast," *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 708–723, 2019.

[65] Y. Wu, "On Constructive Multi-Source Network Coding," in *2006 IEEE Int. Symposium on Information Theory*, Jul. 2006, pp. 1349–1353.

[66] A. Asterjadhi, E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "Toward network coding-based protocols for data broadcasting in wireless Ad Hoc networks," *IEEE Transactions on Wireless Communications*, vol. 9, no. 2, pp. 662–673, Feb. 2010.

[67] T. Kunz, K. Mahmood, and L. Li, "Broadcasting in multihop wireless networks: The case for multi-source network coding," in *2012 IEEE Int. Conference on Communications (ICC)*, Jun. 2012, pp. 5157–5162.

[68] N. Papanikos and E. Papapetrou, "Deterministic Broadcasting and Random Linear Network Coding in Mobile Ad Hoc Networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1540–1554, Jun. 2017.

[69] K. Xiong, P. Fan, H. Yang, and K. B. Letaief, "Space-time network coding with overhearing relays," *IEEE Transactions on Wireless Communications*, vol. 13, no. 7, pp. 3567–3582, 2014.

[70] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[71] A. Kernen, J. Ott, and T. Krkkinen, "The ONE Simulator for DTN Protocol Evaluation," in *Proc. of the 2nd Int. Conference on Simulation Tools and Techniques*, ser. Simutools '09. ICST, Brussels, Belgium, Belgium: ICST, 2009, pp. 55:1–55:10.

[72] "NFD Developer's Guide - Named Data Networking (NDN)." [Online]. Available: https://named-data.net/publications/techreports/ndn-0021-7-nfd-developer-guide/

[73] M. V. Pedersen, J. Heide, and F. Fitzek, "Kodo: An Open and Research Oriented Network Coding Library," in *Lect. Notes in Computer Science*, vol. 6827. Springer Publishing Company, May 2011, pp. 145–152.

[74] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, "Mason: A multiagent simulation environment," *Simulation*, vol. 81, no. 7, pp. 517–527, 2005.

[75] K. Sullivan, M. Coletti, and S. Luke, "Geomason: Geospatial support for mason," Department of Computer Science, George Mason University, Tech. Rep., 2010.

[76] A. T. Crooks and S. Wise, "GIS and agent-based models for humanitarian assistance," *Computers, Environment and Urban Systems*, vol. 41, pp. 100–111, Sep. 2013.

[77] F. Fiedrich and P. Burghardt, "Agent-based systems for disaster management," *Communications of the ACM*, vol. 50, no. 3, pp. 41–42, 2007.

[78] Z. Lu, G. Qu, and Z. Liu, "A Survey on Recent Advances in Vehicular Network Security, Trust, and Privacy," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 760–776, Feb. 2019.

[79] T. Li, W. Chen, Y. Tang, and H. Yan, "A homomorphic network coding signature scheme for multiple sources and its application in IoT," *Security and communication networks*, vol. 2018, 2018, publisher: Hindawi.

[80] X. Hu, S. Zheng, J. Gong, G. Cheng, G. Zhang, and R. Li, "Enabling linearly homomorphic signatures in network coding-based named data networking," in *Proceedings of the 14th International Conference on Future Internet Technologies*, ser. CFI'19. Association for Computing Machinery, 2019, pp. 1–4.

[81] C. Fragouli, J. Widmer, and J.-Y. Le Boudec, "Efficient Broadcasting Using Network Coding," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 450–463, Apr. 2008.

[82] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org , 2017.

[83] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD dataset cambridge/haggle (v. 2009-05-29)," Downloaded from https://crawdad.org/cambridge/haggle/20090529/imote, May 2009, traceset: imote.

Biography

Katherine (Raven) Russell received an AA in General Studies with a concentration in Massage Therapy, a BS in Computer Science, and an MS in Computer Science. She worked in industry for over a decade in various positions, from freelance web developer to senior developer, and at a various companies dedicated to supporting non-profits and government outreach programs. She currently teaches university computer science classes, she worked at the SPCA as a teenager, and is a nidan in kendo.