NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

MODELING GLOBAL NAVIGATION SATELLITE SYSTEM POSITIONAL ERROR EFFECTS ON GEOGRAPHIC ROUTING PROTOCOLS

by

Benjamin T. Harper

June 2020

Thesis Advisor: Justin P. Rohrer
Co-Advisor: Robert Beverly

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 2020	3. REPORT TYPE AND DATES COVERED Master's thesis

4. TITLE AND SUBTITLE MODELING GLOBAL NAVIGATION SATELLITE SYSTEM POSITIONAL ERROR EFFECTS ON GEOGRAPHIC ROUTING PROTOCOLS	5. FUNDING NUMBERS
6. AUTHOR(S) Benjamin T. Harper	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT (maximum 200 words)

Recent work on geographic routing protocols has shown that Global Navigation Satellite System (GNSS) Position Navigation and Timing (PNT) positional error directly affects message delivery rates (MDRs) within Delay and Disruption Tolerant Networks (DTNs). Although research within this area continues, optimization techniques and the potential facilitation of new protocols that are immune or relatively impervious to PNT positional error have not been thoroughly explored, nor has their application been studied within the Department of Defense (DOD). Accuracy within DTNs, especially in urban and natural canyons, that is complementary with higher MDRs is a desirable characteristic for the warfighter operating in challenging environments.

This thesis examined four geographic routing protocols, specifically looking at their current state, potential optimizations, and performance after artificially introducing PNT position error. Analysis of the results showed that certain geographic routing protocol performance metrics are drastically degraded when positional error is introduced, whereas others remained unaffected or were improved. Specifically, we observed a surprising result for certain protocols where performance was improved when PNT positional error was introduced.

14. SUBJECT TERMS BDS, centroid, Delay and Disruption Tolerant Networks, DTNs, DTN routing, DTN simulation, epidemic, Global Navigation Satellite System, GNSS, GPS, GALILEO, networking, network simulator, ONE, Position Navigation and Timing, PNT, vector, message delivery rates, MDRs	15. NUMBER OF PAGES 129
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

MODELING GLOBAL NAVIGATION SATELLITE SYSTEM POSITIONAL ERROR EFFECTS ON GEOGRAPHIC ROUTING PROTOCOLS

Benjamin T. Harper
Lieutenant Commander, United States Navy
BS, U.S. Naval Academy, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN CYBER SYSTEMS AND OPERATIONS

from the

NAVAL POSTGRADUATE SCHOOL
June 2020

Approved by: Justin P. Rohrer
Advisor

Robert Beverly
Co-Advisor

Thomas J. Housel
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Recent work on geographic routing protocols has shown that Global Navigation Satellite System (GNSS) Position Navigation and Timing (PNT) positional error directly affects message delivery rates (MDRs) within Delay and Disruption Tolerant Networks (DTNs). Although research within this area continues, optimization techniques and the potential facilitation of new protocols that are immune or relatively impervious to PNT positional error have not been thoroughly explored, nor has their application been studied within the Department of Defense (DOD). Accuracy within DTNs, especially in urban and natural canyons, that is complementary with higher MDRs is a desirable characteristic for the warfighter operating in challenging environments.

This thesis examines four geographic routing protocols, specifically looking at their current state, potential optimizations, and performance after artificially introducing PNT position error. Analysis of the results showed that certain geographic routing protocol performance metrics are drastically degraded when positional error is introduced, whereas others remained unaffected or were improved. Specifically, we observed a surprising result for certain protocols where performance was improved when PNT positional error was introduced.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

List of Acronyms and Abbreviations

API	Application Programming Interface
ARPANET	Advanced Research Projects Agency Network
ASCII	American Standard Code for Information Interchange
BDS	BeiDou Navigation Satellite System
BP	Bundle Protocol
CLI	Command Line Interface
CPS	Cyber Physical Systems
CWC	current window counter
DARPA	Defense Advanced Research Projects Agency
DP	delivery predictabilities
DTN	Delay and Disruption Tolerant Network
DTNRG	Delay and Disruption Tolerant Network Research Group
DOD	Department of Defense
DOS	Denial of Service
EBR	Encounter Based Routing
EDAQ	Earliest Delivery with All Queues
EDLQ	Earliest Delivery with Local Queuing
ESA	European Space Agency
EU	European Union
EV	encounter value
FIFO	First In First Out
FOC	Full Operational Capability
GAPR	Geolocation Assisted Predictive Routing

GAPR2	Geolocation Assisted Predictive Routing 2
GEO	Geostationary Orbit
GIS	Geographic Information System
GLONASS	Globalnaya Navigazionnaya Sputnikovaya Sistema
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GUI	Graphical User Interface
HEO	Highly Elliptical Orbit
IGCN	Intergalactic Computer Network
IGSO	Inclined Geosynchronous Orbit
IPN	Interplanetary Internet
IPNRG	Interplanetary Networking Research Group
IPNSIG	Interplanetary Networking Special Interest Group
IoT	Internet of Things
IP	Internet Protocol
JPL	Jet Propulsion Laboratory
LAN	Local Area Network
LCAC	Landing Craft Air Cushion
LEO	Low Earth Orbit
LOS	Line of Sight
MANET	Mobile Ad-hoc Network
MAX	MaxProp
MDR	Message Delivery Ratio
MED	Minimum Expected Delay
MEO	Medium Earth Orbit
METOC	Meteorology and Oceanography

MEU	Marine Expeditionary Unit
MFP	Most Forward Progress
NASA	National Aeronautics and Space Administration
NavIC	Navigation with Indian Constellation
NEO	Noncombatant Evacuation Operations
NFC	Near Field Communication
NPS	Naval Postgraduate School
ns-3	Network Simulator 3
NTP	Network Timing Protocol
OCX	Next-Generation Operational Control Segment
OMN	Opportunistic Mobile Networks
ONE	Opportunistic Network Environment
PNT	Position Navigation and Timing
PRC	Peoples Republic of China
PRoPHET	Probability Routing Protocol using History of Encounters and Transitivity
QZSS	Quasi Zenith Satellite System
RAPID	Resource Allocation Protocol for Intentional DTN
RFC	Requests for Comments
RREQ	Route Request
RU	Russian Federation
SnW	Spray and Wait
SUMO	Simulation of Urban MObility
TCP	Transmission Control Protocol
TTC	Telemetry, Tracking and Command
TTL	Time to Live
USN	U.S. Navy

UDP	User Datagram Protocol
US	United States
USG	United States government
USN	United States Navy
USMC	United States Marine Corp
UTM	Universal Turing Machine
VANET	Vehicular Ad-hoc Network
WAN	Wide Area Network
WWII	World War II

Acknowledgments

It is hard to believe that I have made it through countless nights staying up late, studying for the next exam, putting the finishing touches on a paper, or finishing a project all to get to the point where I can finally walk away with a master's degree. All I can say is that it was a hard-fought achievement, and I certainly could not have done it alone.

I cannot express the love that I have for my wife. Your patience and understanding transcends words, and without you in my life and embarking on the Journey with me, I would be lost. You are truly the rock that gives me strength in all aspects of my life. To my daughter; God blessed us with your presence at the perfect time. I have watched you grow up and explore the world with wonder over the past year. No matter the stressors I was feeling, you always had a way to make me smile and think about how awesome it is to be a father. I love you both with all my heart.

To my cohort – thanks for being there through all the tough times. You are some of the best people and professionals I have had the opportunity to serve with. You made the entire experience one to remember. We started this journey together, struggled, and somehow got through. You are, and will always be my friends. I hope that our paths cross again someday.

Lastly, to Dr. Rohrer and Dr. Beverly – your ability to teach in a way that makes the topic enjoyable was the primary reason I asked you both to advise me. You never lacked the patience to hear my questions or concerns and worked with me to find a way to figure out the code or answer the questions I didn't even know I was asking. Without your support, I do not think I would be leaving with a degree in hand. Thank you.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

The purpose of this chapter is to briefly introduce the area of research we conducted in this thesis. We begin by discussing the fundamental differences between network routing characteristics of the Internet and Delay and Disruption Tolerant Networks (DTNs). We then briefly introduce Geolocation-Assisted Routing Protocols, a subset field of DTN research. Scope and limitations are then discussed, followed by an outline of the rest of the thesis. This chapter sets the framework of this thesis so that the reader thoroughly understands the problem we are addressing. Those interested in a more in-depth background, should jump to Chapter 2, and those strictly interested in our analysis of the results, should read Chapter 4.

1.1 Delay and Disruption Tolerant Network (DTN)

The decreasing size and cost of computing in the late 1970s led researchers to begin prototyping and fielding commercially viable wireless technologies. The 1980s and the 1990s brought forth the field of ad-hoc technologies like Mobile Ad-hoc Networks (MANETs) and Vehicular Ad-hoc Networks (VANETs). Parallel to this was the need to communicate to man-made objects in space where the connection is often intermittent, prone to high latencies, and less predictable than terrestrial communication mediums. The Interplanetary Internet (IPN), a project funded by Defense Advanced Research Projects Agency (DARPA) and owned by National Aeronautics and Space Administration (NASA), was created to address the challenges of communication in space [1]. The marrying of these technologies eventually led Kevin Fall [2] to coin the term DTN.

The modern Internet relies on protocols that have been mature for decades. Transmission Control Protocol (TCP) provides stable transport-layer reliability between source and destination nodes [3], and User Datagram Protocol (UDP) capitalizes on little overhead to exchange data at the cost of reliability [4]. Both protocols rely on a fundamental principle of at least a single, stable path between the source and destination, and if broken or times out, the exchange is dropped. The need for a persistent end-to-end path is fundamental for

the exchange of data across the Internet, regardless of which protocols are implemented. DTNs address the challenge where end-to-end paths are either intermittent or non-existent, while providing reliable mechanisms for data exchange across the network. DTNs implement various mechanisms built upon the store-and-forward paradigm where devices act as both end-devices and routers simultaneously. The idea is that participating nodes act in concert with each other to process data from source to destination by replicating messages, hop-by-hop, even if a contemporaneous end-to-end path does not exist. Strategies have been implemented to maximize the successful delivery of a message from a source node. Flooding strategies take advantage of connections by replicating messages to every neighboring node. This strategy performs well if nodes have unlimited storage and power capabilities, which is not realistic because nodes have finite memory and power capabilities. Consequently, this flooding strategy comes with its challenges. As each participating node floods its message, the DTN can quickly become saturated and congested. Although mechanisms have been incorporated to prevent this, protocols like *Epidemic* [5] take advantage of any neighbor connection to forward traffic. This characteristic is synonymous with Opportunistic Mobile Networks (OMN) where participating nodes operate opportunistically to ensure maximum probability of message delivery.

Because DTN research is still in its infancy even after nearly two decades, there still is no agreed-upon standardized routing protocol; however, Requests for Comments (RFC) 4838 [6] and RFC 5050 [7] were both published in 2007, to define a common architecture that protocols could be built upon. The *Bundle* protocol defined a series of data blocks as bundles that hold enough meta-data to allow protocols to make routing decisions across the DTN. The idea is to push message decapsulation up to the application layer instead of relying on the lower layers to gather and aggregate messages. This methodology is important when the lower layers are often in a non-persistent state. More on the *Bundle* protocol is discussed in Chapter 2.

Even as the barrier to entry into the evolving world of mobile networks is comparatively smaller than it was at the turn of the century, new challenges have been generated. People have increased access to the Internet through the purchase of cheaper and more capable devices such as smartphones, especially in rural areas. Users will benefit from future DTN technologies where terrestrial mediums are either too costly to install or where aging infrastructure does not offer Internet Protocol (IP) services. The obvious positive impact to

the Department of Defense (DOD) is of particular interest. The unmounted Marine, Sailor, or Soldier operating in the most remote areas of the world, could benefit from DTNs that provide relatively reliable IP services where current technologies either weigh too much for long-term hauling in the field or require other means of transportation. Further, as peer and near-peer nations grow their technologies, the availability of traditional IP service methodologies can quickly become contested. More detailed background on DTNs can be found in Chapter 2.

1.2 Geolocation-Assisted Routing Protocols

Geolocation-Assisted Routing is a routing strategy found most commonly in wireless networking. Protocols take advantage of positional data of the source, a relay node, and the destination to aid in their routing decisions [8]. Many strategies incorporate positional data and typically are categorized as single-path, multi-path, and flooding. Each is described in further detail in Chapter 2; however, it must be noted that the fundamental characteristic of geolocation-assisted routing protocols depend on underlying positional data from another source, specifically a satellite in space.

Because DTNs operate under the assumption of intermittent end-to-end paths, they employ varying methodologies to assist them in making routing decisions. Many DTN protocols incorporate positional data from Global Navigation Satellite Systems (GNSSs) to aid in their message forwarding decisions. Geolocation-assisted routing protocols take advantage of positional data by forwarding traffic to the node closest to the destination. Since DTNs are often employed in constrained or contested environments, the positional signal acquired may introduce positional errors affecting a protocol's routing decision [9]. These errors may be artificially introduced by a sophisticated actor or caused atmospheric effects as the signal travels from the satellite in space to the node on the ground. Because protocols such as Vector [10], Centroid [11], Geolocation Assisted Predictive Routing 2 (GAPR2) [12], and others each use positional data to assist in their routing decisions, any degradation of the Position Navigation and Timing (PNT) signal has the potential to degrade DTN performance and must be studied further. This degradation is at the heart of the problem we address in this thesis. Note, for the remainder of the thesis, geolocation-assisted routing protocols, are referred to as geographic routing protocols, as the terms are used interchangeably.

1.3 Motivation and Corresponding Objectives

Recent work on three geographic routing protocols have shown that PNT positional error directly affects Message Delivery Ratios (MDRs) within DTNs [9]. Although research within this area continues, optimization techniques and the potential facilitation of new protocols that are immune or are relatively impervious to PNT positional error have not been explored thoroughly, neither has their application been studied within DOD DTN networks. Accuracy within DTNs, especially in urban and natural canyons, that are complementary with higher MDRs, decreased network latency, and provide higher goodput, are desirable characteristics that the warfighter operating in challenging environments can benefit from.

The motivation of this thesis stems from research previously conducted at the Naval Postgraduate School (NPS) in Monterey, California. Rohrer's paper titled *Effects of GPS Error on Geographic Routing* demonstrated the negative effects of positional error on DTN geographic routing protocols [9]. By artificially introducing Global Positioning System (GPS) positional error into Vector [10], Centroid [11], and CenterMass [11] DTN protocols, and by running numerous simulations in the Opportunistic Network Environment (ONE) simulator [13], he concluded that both Centroid and CenterMass were immune to positional error; yet, in other protocols like Vector, performance was drastically degraded. Specifically, Centroid and CenterMass both outperformed Vector by roughly 20% when error was introduced [9].

This thesis expands on Dr. Rohrer's initial findings by artificially introducing positional error into four geographic routing protocol primitives. Positional data acquired from GNSS systems can be degraded in two distinct ways. The first is based on the effects of signal strength degradation caused by natural or urban canyons (i.e., signal obstruction/Line of Sight (LOS)), and orbital elements of the GNSS systems. The second is based on how the signal propagates from the transmitter in space, through the Earth's atmosphere, to the receiver on the ground. Chapter 2 will discuss each in greater detail.

Since current DTN geographic routing protocols make routing decisions without the benefit of consistent global routing information that GNSS provides, the ability to incorporate positional information into DTN routing algorithms further the advancement and maturation of the current state of existing protocols. It should be noted that many of the current DTN protocols have been designed around specific models or scenarios and are not genuinely

universal OMN. There currently is no accepted standardized DTN geographic routing protocol, as the field of research is still relatively in its infancy even with more than a decade of research completed.

To that end, we capture performance metrics of four geographic routing protocols, Vector [10], Centroid [11], Geolocation Assisted Predictive Routing (GAPR) [14], and GAPR2 [12], through the use of a discrete-even network simulator, Network Simulator 3 (ns-3) [15]. The reasoning behind why we chose these protocols is complementary to why we chose ns-3 as the simulator of choice. Three of the protocols have been studied in a similar manner using the ONE [13] simulator [9] but have not been simulated in ns-3 which is arguably a more realistic simulator if researchers are interested in all levels of the network stack. More on why ns-3 was chosen over the ONE is outlined in Section 3.1 of Chapter 3. With the exception of Vector, the other three protocols have roots at NPS and, as such, documentation and access to the source code is available [11], [12], [14].

To expand on Rohrer's research [9], we introduce unclassified and publicly available accuracy bounds of GNSSs through code manipulations inside the core of our simulator and run simulations against their unaltered baselines. Through a comparative analysis of each protocol and the artificial positional error that we introduce, we determine how conflicting global routing information affects performance. Specifically, we look at MDRs, latency, goodput, and hop counts of each protocol. Through a repetitive process of simulation and comparison, we are able to model which of the protocols we studied performed the best.

The importance of research into DTNs by the DOD cannot be overstated. The ability to maintain connectivity and route traffic in contested environments are desirable characteristics for operators both in the field and at sea. As adversaries continue to close the technological gap, the United Sates must add reliable communication methods to its arsenal. Although DOD research has yet to produce battle-tested and operational sound DTN capabilities, the need is clearly there. Low cost, low power, reliably connected devices capable of message delivery in contested environments provide the warfighter the competitive advantage over peer and near-peer nations. This thesis is just one part of that research and is a step closer to the realization of DTN technologies used by the DOD in future operations.

1.4 Significant Findings

This study finds that the introduction of GNSS PNT positional error has both positive and negative effects on geographic routing protocols. In some scenarios, the introduction of positional error seems to cause a protocol to make bad routing decisions, improving its performance at the same time. Both GAPR and GAPR2's MDR and goodput performance when running the Helsinki and Bold Alligator mobility scenarios, showed an increase in performance when error was introduced. Vector, GAPR, and GAPR2 showed a decrease in performance when error was introduced in the Omaha scenario. This suggests that introducing error does have an effect on geographic routing protocols both positively and negatively; however, a protocol's performance is more affected by the mobility scenario that it is running. Ultimately, we concluded that a single geographic routing protocol does not fit all mobility scenarios, and that protocols should be tailored (or fit) to a specific scenario.

1.5 Scope and Limitations

The scope of this thesis is explicitly limited to the introduction of PNT positional error on DTN geographic routing protocols described previously. Other DTN architectures and protocols that are currently being fielded or are being researched are briefly discussed in Chapter 2, as they are supplementary, but are not directly, within the scope of this thesis.

Actual location data is simulated in ns-3 through configurable parameters that follow a normal distribution based on empirical data in the field [16] and on unclassified publicly available GNSS accuracies. We disregard the conversion of latitudes and longitudes to a Mercator coordinate system (flat 2d representation of Earth), and instead, introduce a simple x,y coordinate reference system. Note, that ns-3 has specific modules built into the baseline installation package coverts latitude and longitude into Cartesian coordinates (i.e. `geographic-position.cc`) [15].

We do not consider DTN security mechanisms in this research. Although the secure transmission of data, especially in military operations, is important, it can be an entire thesis on its own; therefore, we recommend it as an area of future study. Section 2.11 of Chapter 2 briefly describes known vulnerabilities of DTNs for those who are interested.

1.6 Thesis Structure and Breakdown

The following is a breakdown of subsequent chapters and a brief description of what is covered in each. Individuals seeking a more detailed description and understanding of DTN technologies and the current state of research in the field should read Chapter 2, background and related works, and Chapter 4, data collection, and analysis.

Chapter 2 discusses the Internet and DTNs, introduces fundamental orbital mechanics, orbit motion, and classic orbital elements to the reader. A simple understanding of GNSS orbits will bridge the knowledge gap between signal propagation and acquisition by DTN nodes on the ground. A discussion of current DTN protocol methodologies follows. Other protocols are briefly discussed as supplemental information as they are considered important to the basic understanding of DTN functionally. Furthermore, Chapter 2 will outline the past and present state of DTN research through a comprehensive literature review.

Chapter 3 details our experimental design and describes how we used ns-3 to simulate GNSS PNT positional inputs against each DTN protocol. Constraints and assumptions are also presented within the context of the simulation. Last, specific command-line arguments and settings are provided so that our methodology can be replicated by others conducting related DTN research in the future.

Chapter 4 describes how our data was collected, processed, and analyzed. This is the data that is derived from Chapter 3. Analysis of data in this chapter formulates our conclusions and suggestions for areas of future study pertinent to DTN research specifically, geographic routing and related protocols, and it is outlined in Chapter 5.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background and Related Work

Connectivity is one of the most critical aspects of the Information Age. The ability to communicate and exchange information across vast distances, nearly instantaneously, has become a regular aspect of everyday life. In the 1960s, the concept of bridging isolated computers together through a connection medium laid the foundation of the modern Internet. The Advanced Research Projects Agency Network (ARPANET) Project connected academia and government agency networks that were hundreds of miles apart, forever changing the landscape of technological collaboration [17]. Although it is debatable whether ARPANET was envisioned to be strictly an academic research network or a military command and control network that would survive a nuclear attack, it clearly exceeded the initial goals of its creators. It shaped how information has been exchanged since its creation [18], [19].

Although this newfound "connectivity" ushered in an era of technological leaps and ease of collaboration that had typically been reserved for the authors of Science Fiction novels, it forced the materialization of unique challenges in its wake. The decreasing cost and size of personal computing devices that were mobile gave energy to researchers, ultimately leading to the development of protocols used in MANETs and VANETs in the late 1990s and early 2000s [20], [21].

As new ways of connecting computers across the Internet continued to propagate throughout the community of computer science, so did the imagination and innovation of researchers. Technologies that seemed so far out of reach decades earlier were no longer outside the current state of the field. In 2001, a joint venture between NASA and the Jet Propulsion Laboratory (JPL), led by Vint Cerf (critical to the success of the ARPANET Project and considered a founding father of the modern Internet), formed the Interplanetary Networking Special Interest Group (IPNSIG), formally Interplanetary Networking Research Group (IPNRG), to address communication challenges in space [1], [22]. Under Cerf's leadership, IPNSIG produced the IPN architectural design, the precursor, and motivation to the Delay and Disruption Tolerant Network Research Group (DTNRG), (no longer active as of April 5th, 2016) [23] and its subsequent delve into DTN research in the early 2000s.

The IPN attempted to create a solution to deep-space communication where transmissions are significantly delayed, disrupted, and are prone to errors. The premise of DTNs stems from similar challenges faced by IPNs, specifically, DTN's attempt to address network traffic in challenging environments where typical Internet heterogeneity fails due to delays, disruptions, or errors. The need for a network capable of connectivity, regardless of the environment, that is ad-hoc by design, is self-forming and self-healing, are desirable characteristics researchers have been studying for nearly two decades.

Adding to this challenge, is the rise of emerging technologies that continue to proliferate around the world. The growth of the Internet of Things (IoT), the inclusion of Cyber Physical Systems (CPS) on the Internet, and 3.50 billion mobile smartphone users in the world today [24], further drive the need for effective DTN protocols that meet the demands of increased connectivity. Rural areas where infrastructure is limited [25], [26], natural disasters where infrastructure has been destroyed [27], military operations where infrastructure is contested [28], and even the study of Zebra migrations [29] all provide possible applications of DTN technologies. Indeed, the desire for effective DTNs is readily apparent.

In this chapter, we begin with a history of the Internet, followed by a discussion of DTNs and Geographic Routing Protocols. A discussion on PNT follows. We then introduce basic orbital mechanics, orbital motion, and classic orbital elements to provide the reader background on how signals reach devices on the ground from satellites in space. We then discuss the six on-orbit GNSS systems by providing unclassified, publicly available characteristics, and accuracy parameters that we used in our simulations. A discussion of related areas of research of DTN protocols follows. Finally, we wrap up the chapter with a discussion of the protocols pertinent to this thesis and briefly examine that state DTN security.

2.1 The Internet

The modern Internet spans almost a century of technological growth and advances. Alan Turing, the father of computer science [30], through his creation of the Universal Turing Machine (UTM), created the environment that was a prerequisite for the Internet to exist. As early as the 1960s, a network of computers that were decentralized and could exchange information rapidly, had been the driving force for most of today's network innovations.

The latter half of the 20th century gave way to networking advances that have changed the very nature of how information was and has been distributed.

Figure 2.1 represents decades of achievements critical to the current state of the modern Internet. Although not inclusive of every milestone, it captures essential elements that the Internet and DTNs are built upon. As the Internet continues to mature over time, so does its complexity. New challenges continue to present themselves as the broader application of the Internet is used in uniquely different ways. One of these challenges is producing the Internet-like functionality when its core tenant of end-to-end connections are delayed, disrupted, or broken altogether.

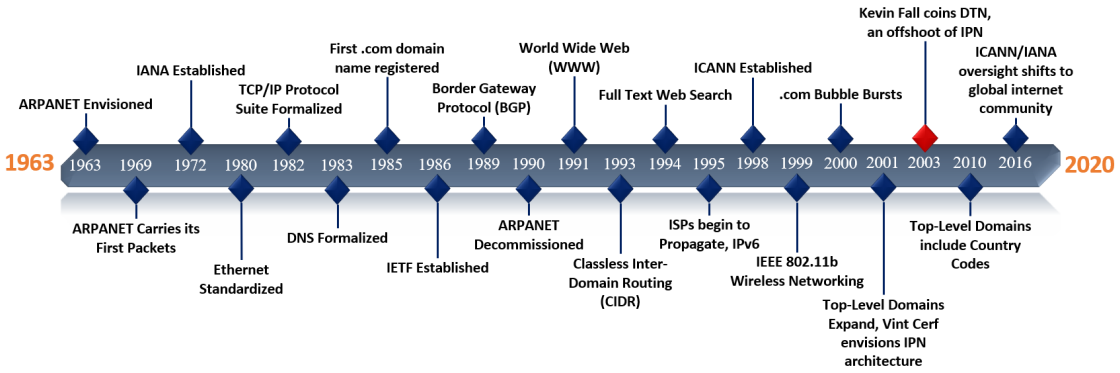


Figure 2.1. The History of the Internet.

To understand precisely what DTNs are, we must first understand how the Internet functions at the most fundamental level. Abstracting away the global nature of the Internet and focusing on an individual subset, a Local Area Network (LAN), we can better paint a picture of how the Internet scales to serve billions of devices and users.

The Internet is a network of networks. A network is defined by a set of nodes and a set of edges. Every node in the network forms a subset of edges that connects them to all the other nodes on the network; therefore, a network is essentially a group of nodes where communication takes place over edges. The more edges that connect nodes together, the more connected the network is. There are many network topologies that provide varying levels of connectivity among the nodes, each with differentiating scalability. A full-mesh topology, where every node is connected to every other node via a direct edge, is considered

the most connected network. Although a full-mesh is the epitome of network topologies because it is a set that contains all objects, including itself, it does not scale well within larger networks. It would be unrealistic to expect that every node on the Internet is connected to every other node, and the costs would be astronomical.

The addition of routers allows for scalability within the Internet. Nodes at the edge of a network are serviced by routers that route traffic from sources to destinations through effective addressing mechanisms and connect LANs together to create Wide Area Networks (WANs), thus creating a network of networks, (i.e., the Internet) [31]. Figure 2.2 depicts the spatial scope of different network types, where the spatial scope is the total number of nodes serviced.

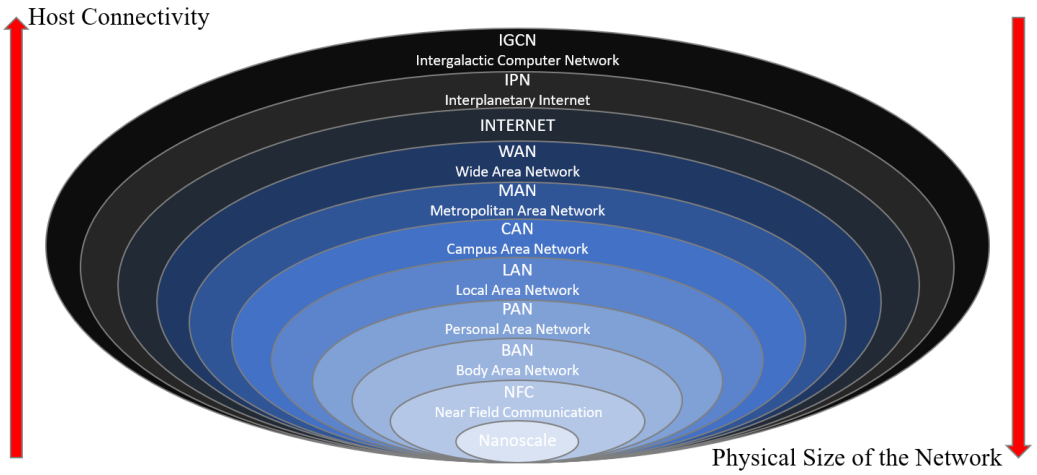


Figure 2.2. Network Types by Spatial Scope.

Each sub-network is connected to the same level of sub-network through the larger network type. The smallest of the spatial scopes is Near Field Communication (NFC), where a single point-to-point connection is established between two devices. Although only conceptualized, the largest of these is the Intergalactic Computer Network (IGCN) that would connect networks from within a solar system to networks or devices outside the solar system [32]. The smallest of these networks, physically, is the Nanoscale network, where nanomachines communicate with each other to complete tasks microscopically [33]. They may have significantly more hosts that are connected than found in other spatial scopes, but are physically much smaller.

At its core, the Internet operates on an assumed end-to-end connection between two nodes where at least one path exists between them. Protocols and mechanisms are implemented to ensure the success of communication and traffic delivery between two nodes [34]. Conversely, DTNs attempt to communicate where end-to-end connectivity is scarcely present.

2.2 Understanding DTNs

DTNs are fundamentally different in their approach to routing traffic than traditional Internet routing. Very long delay paths, frequent network partitions, intermittent connectivity or limited connection windows, and intermittent end-to-end connectivity are all facets of DTN characteristics [2], [35]. Furthermore, nodes within a DTN may be subject to limited power or memory resources that make the routing of traffic burdensome.

Kevin Fall, another Internet pioneer, first coined the term DTN [2]. He describes an architecture that provides interoperability among heterogeneous networks and a shift from traditional Internet Store-and-Forward to Store-Carry-and-Forward paradigms [35], as depicted in Figure 2.3. The idea was to create an architecture that serves as an overlay on top of the transport layers of the underlying networks that operate under, "reliable message routing" versus "best-effort packet switching."

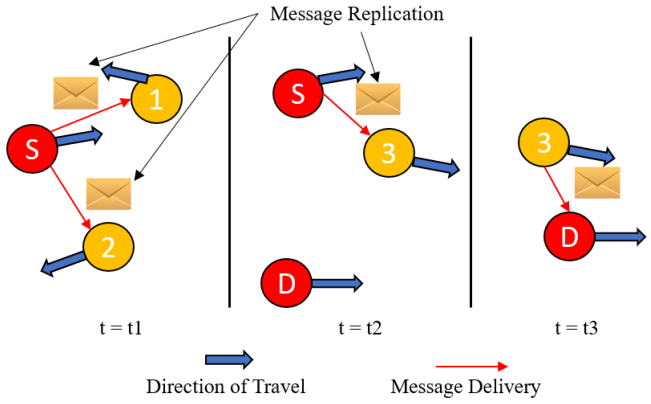


Figure 2.3. DTN Store-Carry-and-Forward Paradigm. Adapted from [35].

Further, DTNs use message replications through opportunistic routing mechanisms (described in further detail in Section 2.9) to deliver messages reliably when connections are

available. Simply put, a source node sends its message to all neighboring nodes within range with which it has a stable connection, (e.g., node relays). These relays store the message in a buffer and then continues the process until the destination finally receives the message. This Store-Carry-and-Forward paradigm is the core principle that DTNs operate on. The critical difference between traditional Internet and DTN routing protocols is that each node within a DTN serves as both a host and a router, working with other nodes in concert to get the message from the source to the destination reliably. This paradigm also creates multiple copies of the original message within the DTN along the way [36]. Lastly, as nodes join or leave a DTN, dynamic convergence of the network happens automatically, producing the "self-forming" and "self-healing" effect characteristic of MANETs.

Because DTNs store messages, each node in the network must determine which message to forward due to the lack of unlimited onboard storage capacity. Simply forwarding every message a node receives will quickly saturate the network, preventing any successful delivery of messages across the network. DTN protocols, like Epidemic, follow this flooding paradigm, and if nodes in the DTN have unlimited storage, transmission, and power capabilities, they have the highest probability of message delivery [5]. This paradigm is not realistic, and nodes must effectively manage their resources to ensure message delivery based on their individual characteristics. Other protocols such as Direct Delivery [37], Epidemic [5], Spray and Wait (SnW) [38], Probability Routing Protocol using History of Encounters and Transitivity (PRoPHET) [39], MaxProp (MAX) [40], and Resource Allocation Protocol for Intentional DTN (RAPID) [41] use knowledge of previous encounters with other nodes and location information to limit message replication and achieve higher probabilities of message delivery.

2.3 Geographic Routing Explained

Geographic routing is a method of delivering a message to a node, over multiple hops, through the use of positional information. As with other DTN routing protocol strategies, geographic routing does not rely on typical Internet addressing and the use of routing tables to route messages across a network. Instead, nodes use positional information to make routing decisions and route messages towards a destination's location. This is done through three fundamental assumptions [8].

1. A node can determine its own position.
2. A node is aware of its neighbors' positions.
3. The position of the destination is known.

The ability of a node to retain positional data is a desirable characteristic of DTNs because it reduces the burden of storing network routing information for the entire network and increases the capacity of nodes to Store-Carry-and-Forward messages as they move; however, the capacity of geographic routing protocols to effectively route messages is entirely dependent on the node's ability to maintain GNSS signal acquisition. This can be difficult when nodes are operating in challenging environments such as natural or urban canyons where the LOS to a satellite can be obstructed, or when atmospheric conditions degrade the signal itself. Further, certain protocols, such as greedy forwarding [42] where a node always transmits its message to a node closer to the destination, creates a local minimum problem [8]. The local minimum problem exists when a node transmits to a node closer to the destination's position without knowing if it has any neighbors closer to the destination, essentially reaching a dead end. Methods have been developed to overcome the local minimum problem and are discussed in Section 2.9.

2.4 Position, Navigation, and Timing (PNT)

Geographic routing protocols rely on geolocation information to assist in making routing decisions. PNT services come from GNSS systems in space and are predominately associated with vehicle navigation systems. However, timing data is also a critical service provided by GNSS systems. Timing is a crucial element in computing. Networks utilize timing to synchronize routers across large networks where "clock drift" occurs. This drift is caused by individual device clocks counting time differently [43]. Large enterprise networks require timing synchronization across geographically dispersed routers and often employ Network Timing Protocol (NTP) server technologies utilizing GNSS timing data to address clock drift, ensuring clock synchronization throughout the network [44]. Clock synchronization is critical for Internet protocols using Time to Live (TTL) fields because any difference in timing over a network has the potential for packets being dropped. Timing also is a crucial element in providing positional data. Because of delays in signal propagation from space to the devices on the ground, timing offsets are incorporated into the signal acquisition process to provide the most accurate positional data possible. Navigation data from GNSS is similar

to positional data but includes both the starting point and destination, and apply corrections to course, orientation, and speed to attain the desired position. Positional data provides accurate and precise location information in either two or three dimensions [45]. Positional data is the element of PNT that we focus on in this thesis and is used by geographic routing protocols to make routing decisions. Specifics on GNSS systems currently in operation, their associated accuracies, and other characteristics are discussed in Section 2.6.

2.5 On Orbital Mechanics

Satellite orbits are complex. From building mission payloads and getting them to space, to maintaining the system while on orbit through Telemetry, Tracking and Command (TTC), hundreds of personnel are involved in the operation of just a single satellite. The orbital mechanics of satellites in space need to be precise, and thus require considerable mathematics. We will discuss certain characteristics of orbital mechanics to lay a foundation of understanding. To explain why ground segment devices, such as DTN nodes, receive signals at varying degrees of strength, we must understand the characteristics of signal propagation and their degradation caused by atmospherics.

2.5.1 Orbit Shape and Size

Satellite orbit shape and size are subject to several factors. The characteristics of a satellite's orbit is specific to its mission, and depending on the need, orbital parameters must be calculated and set to achieve the desired shape and size. The complexity of orbital mechanics is not trivial. Orbits must contend with an n-body problem (i.e., the Sun, the Moon, and Earth) each with a gravitational force acting on a satellite. Furthermore, Earth is not completely spherical but an oblate spheroid. This oblateness produces varying gravitational pulls each of which perturbs the orbit of the satellite. Although modeling these variables precisely is important to calculating their effect on GNSS orbits and their PNT signal strengths, such modeling is outside the scope of this thesis because we are only concerned with each individual GNSS accuracies. The background provided is to give the reader a high-level explanation of satellite orbits, while recognizing that orbit's shapes and sizes are complex.

First, we must define two critical elements of orbits. *Apogee* is the point in the orbit furthest

away from Earth. Conversely, *Perigee* is the point in the orbit closest to the Earth. The size of an orbit is related to its specific mechanical energy and is contingent on the gravitational pull of the body a satellite is orbiting. An orbit size is called the major axis and is denoted as $2a$. *Eccentricity* (e) specifies the shape of an orbit and is calculated by dividing the distance between *Apogee* and *Perigee*, $2c$ by the major axis $2a$ or $e = \frac{2c}{2a}$. Table 2.1 lists the shapes of orbits based on *Eccentricity* [46].

Table 2.1. Eccentricity of Orbits. Adapted from [46].

Shape	**Eccentricity**
Circle	$e = 0$
Ellipse	$0 < e < 1$
Parabola	$e = 1$
Hyperbola	$e = e > 1$

2.5.2 Orbit Orientation

Next, we must understand how a satellite's orbit is orientated around Earth. This is the orbital plane tilt observed from the equator and is called *Inclination* (i). The equatorial orbit is the path a satellite takes that is orientated directly above the equator. Polar orbits are orientated from pole to pole. Most satellites follow either a *Prograde* orbit (using an inertial frame of reference, the satellite revolves in the same direction as the rotation of the orbited object on its axis), or a *Retrograde* orbit (using an inertial frame of reference, the satellite revolves in the opposite direction as the rotation of the orbited object on its axis), depending on the mission requirement. For example, having sunlight and following against the Earth's rotation in the retrograde orbit is useful for imagery satellites because shadows allow the analyst to determine heights of objects. Communication satellites typically follow prograde orbits in the direction of the Earth's rotation for faster revisit times overhead to provide persistent communication connections. Table 2.2 characterizes these four orbits and the latitudes off the equatorial plane that they follow [46].

2.5.3 Orbital Motion

With no force acting upon it, an object in motion will continue moving in its original direction. Without sufficient inertia to maintain a stable orbit, the Earth's gravitational pull

Table 2.2. Inclination of Orbits. Adapted from [46].

Inclination	Orbital Type
0° or 180°	Equatorial
90°	Polar
$0° \leq i < 90°$	Prograde
$90° < i \leq 180°$	Retrograde

would cause a satellite to fall back to the surface. For a satellite to stay in orbit, it must be traveling fast enough to continuously fall around the Earth but not fast enough to escape the Earth's gravitational pull. Depending on their speed, satellites can either maintain a geosynchronous orbit over a fixed point or traverse large swaths of the Earth repeatedly revisiting the same path within a certain period. Because other celestial bodies, such as the Sun and Moon, exert a gravitational pull on satellites, some satellites in orbit must adjust course to ensure that they maintain the desired orbit. For the most part, satellites can do this autonomously; however, occasionally TTC commands may be issued from the control segment to force a re-position [46]. It should be noted that there are other aspects to the maintenance of satellite systems, but these aspects are outside of the scope of this thesis.

2.5.4 Classical Orbital Elements

With the basic understanding of orbit eccentricity, the inclination of orbital planes, and the role of Earth's gravity plays, we can now bring all the orbital elements together. Figure 2.4 illustrates these classical orbital elements and are discussed next. The minor axis, $2b$ is the width of the ellipse. The farthest approach, or radius of apogee R_a, is the furthest approach of the satellite to the Earth. The closest approach, or radius of perigee, R_p is the closest point a satellite orbits from the center of Earth. $\vec{R}$ is the satellite positional vector measured from perigee and constantly changes as it follows the orbit. $\vec{V}$ is the satellite's velocity. ϕ is the flight path angle based on the local horizontal of the satellite, which is normally oriented and fixed towards the Earth's surface. F and F' are the primary and vacant foci of the orbit, where F is centered on Earth. v, or the true anomaly, is the angular distance on the orbital path from perigee and determines the location of the satellite on orbit [46]. Note that when $2b$ and $2a$ are equal, F and F' are positioned over the center of the Earth, the orbit, to a close approximation, is circular. If we consider the n-body problem where

gravitational pulls from other celestial bodies acting on a satellite, and the non-spherical (or oblate) shape of Earth, the circular orbit will have perturbations, therefore, the term circular may only be used as an approximation.

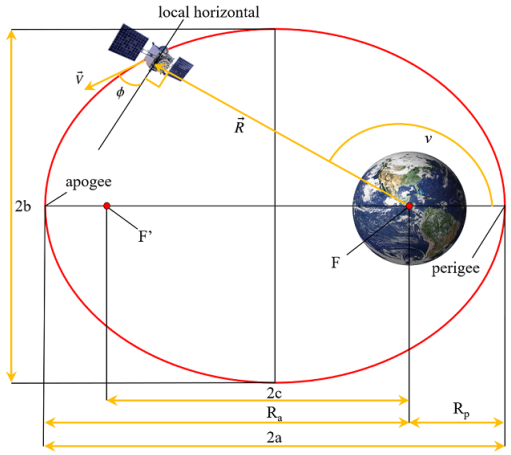


Figure 2.4. Geometrical Parameters of Satellite Orbits. Adapted from [46].

With the classical orbital elements defined, we now understand the characters of satellites that GNSS systems follow. This is critical to understanding why the signal acquisition is difficult in certain environments. When ground segment nodes at the poles have low horizon look angles to a satellite in space or where direct LOS is obstructed by natural or urban canyons, signal acquisition and strength is degraded. The signal must traverse a large portion of the atmosphere, hitting air and water particles along the way, causing signal absorption. Refraction can occur if the atmosphere the signal is traveling through varies in density causing a scintillation (or blinking effect) further impairing signal acquisition. When a satellite is at zenith, LOS provides for the best possible signal strength.

2.5.5 Types of Orbits

Satellite orbits are categorized into five distinct orbital types. Depending on the mission need, satellites are placed at certain distances (in km) from the center of the Earth. These missions range from imagery at Low Earth Orbit (LEO) to communication and intelligence at Highly Elliptical Orbit (HEO) (also known as Molniya). Specifically related to this thesis, nearly all the GNSSs are currently at Medium Earth Orbit (MEO) (or semi-synchronous

orbit) with some exceptions as described in Section 2.6. Table 2.3 depicts the various missions of satellites in each orbit, their orbital types, distance from F (occupied by Earth), orbital periods, inclinations, and their eccentricity.

Table 2.3. Orbit Types by Mission. Adapted from [46].

Mission	Orbital Type	Semimajor Axis	Orbital Period	Inclination	Other
Communication Early Warning Nuclear Detection	Geostationary	$42,158km$	$\sim 24hr$	$\sim 0°$	$e \cong 0$
Remote Sensing Weather	Sun-synchronous Geostationary	$\sim 6,500 - 7,300km$ $42,158km$	$\sim 90min$ $\sim 24hr$	$\sim 95°$ $\sim 0°$	$e \cong 0$ $e \cong 0$
Navigation	Semi-synchronous (Medium-Earth Orbit)	$26,610km$	$\sim 12hr$	$55°$	$e \cong 0$
International Space Station	Low-Earth Orbit	$\sim 6,700km$	$\sim 90min$	$28.5°$, $39°$, $51°$, or $57°$	$e \cong 0$
Communication Intelligence	Highly-Elliptical Orbit (Molniya)	$26,571km$ $(R_p = 5,791km; R_a = 45,170km)$	$12hr$	$63.4°$	$e \cong 0.7$

2.6 Global Navigation Satellite System (GNSS)

There are currently six GNSS systems that serve devices either globally or regionally. Each have varying levels of precision, accuracy, power transmission properties, and security built into their operation, and depending on the system, have restricted signals solely for the use of their owning government. Some systems provide encryption capabilities that are jam resistant thus ensuring PNT capabilities are present in contested environments. Note that for the purposes of this thesis, we simulated GPS, Globalnaya Navigazionnaya Sputnikovaya Sistema (GLONASS), and Quasi Zenith Satellite System (QZSS) to give diversification to our model based on the uniqueness of each satellite constellation. The Galileo, BeiDou Navigation Satellite System (BDS), and Navigation with Indian Constellation (NavIC) GNSS systems provide relatively similar performance as the three systems we modeled. These systems are discussed only to aid as supplementary information to this thesis. Table 2.4 lists the characteristics of each system, and it provides the levels of accuracy based on unclassified, publicly available information. We define the assumed max as the maximum GNSS positional error that a ground node receives. The assumed max is based on signal degradations that quickly decay to nearly 200 m, with errors typically occurring orthogonally

Table 2.4. GNSS Characteristics.

System	Country	Coverage	Accuracy(Min)	Assumed Max
BeiDou	PRC	Global	10 m	200 m
Galileo	EU	Global	1 m	200 m
GLONASS	RU	Global	4.5 - 7.4 m	200 m
GPS II (Series)	US	Global	2 m	200 m
NavIC	INDIA	Regional	10 m	200 m
QZSS	JAPAN	Regional	1 m	200 m
GPS III (Series)	US	Global	1 m	200 m

Adapted from [48]–[53].

to the direction of travel [47]. Any error beyond 200 m is assumed to have the same effect; therefore, this is the max error bound we introduce to our model as described in Section 3.3 of Chapter 3.

2.6.1 Global Positioning System (GPS)

The United States owns and operates GPS. It was developed and deployed in 1973, to overcome limitations of previous navigation systems that previously had been in use, and was meant to aid users in government and military operations [54]. Initially reserved for sole use by the DOD, civilian use was granted in 1980, thus making it the first commercially available system providing PNT services and has expanded into a wide variety of commercial markets. Selected availability was built into the system to allow the United States government (USG) assets preferential access and increased accuracy. This restriction was rescinded in 2007, by President Bush [55], effectively removing artificial accuracy incurred on commercial users. The entire constellation consists of 33 satellites, on six orbital plans, on a semi-synchronous MEO, and requires four satellites for signal acquisition. Three satellites are needed to triangulate a position of a node on the ground and a fourth is necessary to provide elevation offsets. If a user were at sea level, typically, three satellites will suffice for nominal accuracy. As a user gains elevation, the accuracy of a fix can be hundreds of meters off [54]. Note that the GPS III series satellites have been replacing the aging GPS II series satellites since 2018. The GPS III constellation comes with new navigation signals, splitting out two unique civilian L1 and L2 codes from the military M-code and safety of life L5 codes that boast an impressive accuracy down to 1 m. Because of complications and budget overruns

associated with the controlling segment of the system, Next-Generation Operational Control Segment (OCX), has prevented GPS III's from achieving Full Operational Capability (FOC). This series III system is expected to reach FOC by 2023 [56].

2.6.2 Globalnaya Navigazionnaya Sputnikovaya Sistema (GLONASS)

Owned and operated by the Russian Federation (RU), the first GLONASS satellite was launched in 1982, with subsequent satellite launches over the years until in 2010, when it achieved a full orbital constellation of 24 satellites, in three orbital planes, with satellites in both semi-synchronous MEO and HEO (Molnyia). A considerable amount of the constellation is positioned above Russia, providing PNT services to both its government and its citizens in direct competition with GPS. Because it employs two types of orbits, GLONASS provides both global coverages through its MEO orbital planes and long overhead time above Russia with its HEO orbital planes.

2.6.3 Galileo

The European Union (EU), through the combined efforts of its member states via the European Space Agency (ESA), launched its final Galileo satellite in 2018, completing its 24 satellite constellation, in two orbital planes, at semi-synchronous MEO, ensuring global coverage for its users. As with GLONASS, a large portion of the Galileo constellation is positioned over Europe to provide reliable PNT services to its user base. It should be noted that concerns issued by the U.S. DOD about the EU's signal characteristics interfering with the GPS sparked a heated debate between the leaders of the EU and the U.S. ultimately resulting in the ESA launching regardless [57].

2.6.4 BeiDou Navigation Satellite System (BDS)

The BDS satellite system is the Peoples Republic of China (PRC) response to GPS and is expected to provide global coverage by 2020, completing its 35 satellite constellation. Due to its three orbital planes at semi-synchronous MEO, one orbital plane at Geostationary Orbit (GEO), and one orbital plane at Inclined Geosynchronous Orbit (IGSO), essentially a highly inclined MEO orbit, BDS is capable of constant overhead coverage comparable to that of GPS [51].

2.6.5 Navigation with Indian Constellation (NavIC)

India's NavIC was designed to provide a persistent LOS coverage overhead. Specifically, it utilizes a GEO orbit consisting of seven satellites in two orbital planes that provide 24/7 coverage over the all of India. Because the system is regional, it cannot provide any PNT globally and lacks encryption or the accuracy capabilities of the other GNSS systems [52].

2.6.6 Quasi Zenith Satellite System (QZSS)

Japan's QZSS is the most accurate of the six systems we discuss, however, because of the design requirements of providing near the zenith (directly above) persistent overhead coverage 24/7, and the fact that the constellation consists of only four satellites in four separate orbital planes, it does not provide global or truly regional coverage. The original intent of the system was to overcome the low look angle of GPS signals caused by the high latitude of the island of Japan relative to the constellation itself. Additionally, most of Japan's users reside in metropolitan areas where urban canyons disrupt and prevent the receipt of signals or cause multipath "bouncing" off the side of buildings. The QZSS receives and transmits GPS signals directly down to Japan overcoming these challenges. Because of this, it is the most precise GNSS system [53].

2.7 PNT Signal Distribution

The main objective of thesis is to observe the performance of geographic routing protocols when artificial positional error is introduced to a particular protocol. To do this, we first had to understand GNSS PNT signal distribution characteristics and, second, how to introduce the error into the protocol code. To solve the first problem, we conducted a comprehensive review of GNSS signal propagation characteristics and found that signal acquisition typically follows a *Uniform Normal Distribution* (or Gaussian "Bell Curve") [16]. The normal distribution is a probability distribution that is typically symmetric around a mean. In the case of this thesis, specific minimum accuracies (best accuracy) are based on Table 2.4 and are implemented in our simulations as decribed in Chapter 3. Further, normal distributions have values that typically fall within a 95% confidence interval of the data set or two standard deviations (std) from the mean.

$$\chi \sim N(\mu, \sigma^2) \tag{2.1}$$

Equation 2.1 defines the normal distribution where μ is the mean and σ^2 is the variance ($std = \sigma$). The mean determines the position that the distribution will be centered around. Variance describes the measure of spread of the data that is represented. Variance is typically set to 1 by default if the shape of the distribution is to be a close to a "Bell Curve" as possible. Specific parameters used in this thesis are discussed in Section 3.3 of Chapter 3. Figure 2.5 illustrates the error ranges and their respective probabilities of three versions of the GPS constellation. Note that since creation of Figure 2.5, the Block IIF, Block IIIA, and Block IIIF GPS systems have since been launched. The figure is a representation of how a typical GNSS signals follow a normal distribution. The figure represents GPS with is minimum accuracy (best) centered around the mean of 1 m.

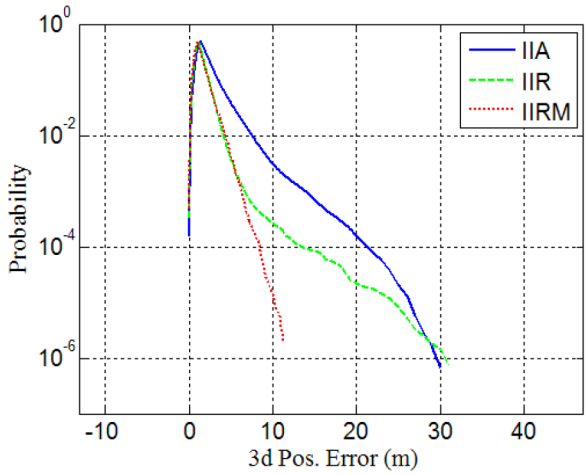


Figure 2.5. Distribution of GPS Signal Acquisition Accuracies. Used with permission [16].

2.8 Bundle Protocol

The Bundle Protocol (BP) is important because it describes a fundamental architecture that DTNs operate on. In RFC5050, Scott et al. proposed an "end-to-end protocol, block formats, and abstract service descriptions for the exchange of messages" (called bundles) [7]. The protocol is at the application layer, forming a store-and-forward overlay network that includes characteristics such as custody-based retransmission, a coping mechanism when intermittent connectivity is present, and the ability to adapt to various flooding and forwarding strategies. At its core is the bundle, which is the protocol data unit of the BP. Each bundle is comprised

of two or more sequences of blocks that serve various purposes. The BP provides service at the application layer for several types of Internets, where the term Internet is not necessarily associated with standard TCP/IP mechanisms [3], [7], [58].

Figure 2.6 illustrates where the BP sits on the network stack, and it demonstrates how a bundle originates from the source, is encapsulated, and then is forwarded to relays where it is torn down and then encapsulated as many times as necessary until it finally reaches the destination. Because the BP operates over various types of Internets as defined in the RFC, it incorporates a convergence layer adapter to handle the exchange between different types of network and transport layers. The BP makes no routing decisions on its own and relies on the underlying DTN protocol's strategies [7].

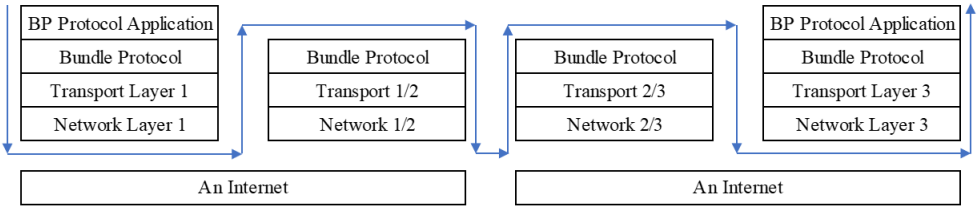


Figure 2.6. The Bundle Protocol Sits at the Application Layer of the Internet Model. Adapted from [7].

2.9 The Current State of DTN Protocol Research

Routing is an important aspect of any network protocol regardless of whether or not it is part of the wired Internet or DTNs. The most trivial of routing taxonomies is direct routing, where the source node directly delivers its messages to the destination itself. This is not a realistic characterization of DTNs since intermittent connectivity is subject to extremely high latencies and low MDRs. Another is first-contact routing where a node forwards its messages to the first node it comes in contact with. This methodology can cause network saturation caused by exponential message replication across the network. It ignores the local minimum problem where the sending node cannot determine if the node it sends its messages to has neighbors, or a bead on the destination, even if the distance between them is small, essentially hitting a dead end [37].

Figure 2.7 depicts the local minimum problem where node *A* sends its message to both *B* and

C. Although B is closer to the destination D, it fails to forward A's message because it has no path to D due to LOS obstruction. Considering the obvious limitations of direct contact and first-contact routing methodologies, and how the local-minimum problem creates challenges in message routing, we discuss current well-known DTN protocols in literature.

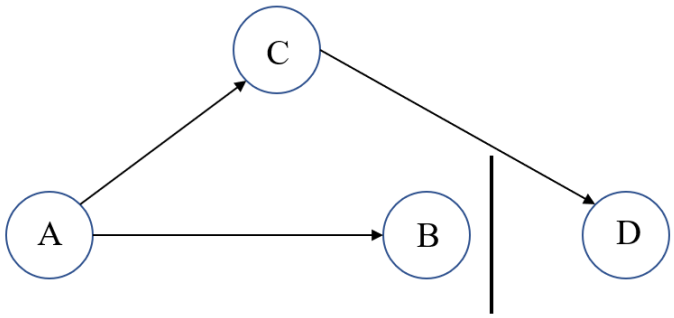


Figure 2.7. Location-based Routing Fails Due to a Local Minimum. Adapted from [37].

2.9.1 Routing Strategies

The field of DTNs research is filled with numerous routing strategies that attempt to increase performance metrics. Each provides certain advantages over others, but so far, no strategy captures all the best characteristics of each strategy. We will discuss several strategies that DTN protocols implement. It is important to understand the differences between those we introduce, as they provide unique methodologies and design frameworks pertinent to this thesis. It should be noted that there are many other DTN routing strategies outside the scope of this thesis that are not discussed.

Single Path/Single Copy

Although the literature classifies single path DTN routing strategies as *forwarding* because it only sends a single copy of a message [59], the distinction between message forwarding and single path should be made. Forwarding strategies encompass multipath routing strategies as well, because a source node will replicate a copy of its message and forward it to multiple nodes. We classify single path routing as a single message that is forward from source to relays, and ultimately to the destination without any participating node making another copy of the original message. This is the most basic routing strategy if conditions were

perfect, and a single end-to-end path was available. This is certainly a plausible condition but unlikely considering the very nature of mobile networks and how DTNs operate under non-persistent connectivity.

Multipath/Multicopy

The multipath routing strategy balances mitigating the problems associated with flooding strategies (described next) and providing enough message redundancy through replication without saturating the DTN completely. Although closely related to flooding strategies, multipath routing strategies create multiple copies of the original message and forward them to a set of neighboring N nodes towards the destination. Through the use of Route Request (RREQ) messages, participating nodes propagate the message exactly once, hop-by-hop until the destination receives it. The destination then sends out a unicast message to all participating DTN nodes advertising the best paths at that moment in time [59]. Note that there may be two or more "optimal paths"; therefore, multiple copies of the original message are sent out to various paths.

Flooding Strategies

Flooding protocols deliver multiple copies of each message to a set of nodes within the network, called relays. Following the Store-Carry-and Forward paradigm, these relays store the messages until the relays are within range of the destination and then deliver the message. Early research of DTN protocols typically fall in this strategy and was developed even before Kevin Fall coined the term in 2003 [2]. Most of the research in flooding strategies evolved around MANETs, where there exists a high probability of a source coming in contact with the destination. This complemented with message replication strategies increase the likelihood of successful message delivery. Previous knowledge about the network is not needed in this strategy; however, innovative flooding strategy protocols incorporate knowledge of network typologies to assist in routing decisions [37].

Gradient Routing

Gradient routing assigns weights to nodes within the a DTN with the probabilities of message delivery. As a node holding a message in its buffer encounters a neighboring node with a better-weighted metric for delivery, it forwards the message to that node, which in turn, repeats the process. The improving weighted values at each encounter creates a gradient, hence the term gradient routing. Because each node requires sufficient information

that must be communicated across the network and stored for all participating nodes on the network, gradient routing requires more knowledge of the network than does location-routing. Performance metrics are malleable and are calculated based on power consumption, times of last encounters, and other parameters, and as such, is not a practical approach to message forwarding [60]. ZebraNet is one such network that employed gradient routing [29].

Link Metrics

Link metrics implement similar routing decision strategies as traditional networking protocols. Link weights are assigned to determine the best performance metric possible. These metrics attempt to provide the best MDR, lowest latencies, and highest bandwidth possible between a source and destination, and are predicated on each node in the network having prior knowledge of the network. This is achieved through various algorithms, most predominately through a modified version of Dijkstra's shortest path algorithm to minimize traffic over the network. The Minimum Expected Delay (MED) metric introduced by Jain et al. [61] assumes that queuing times are near-zero, and message propagation and transmission times are known precisely. In the same paper, they introduced the Earliest Delivery with Local Queuing (EDLQ) and Earliest Delivery with All Queues (EDAQ) metrics, which uses the average buffer capacities of each node in the DTN to estimate the queuing delays and network traffic demands to compute the exact queuing delay, respectively. This forwarding strategy assumes that the network link state is known ahead of time and is stable, where, in reality, DTNs routinely drop links.

Face Routing

Face Routing was first proposed in 1999, and was touted as being the first routing algorithm that guaranteed message delivery without flooding the network with replicated messages [62]. In this algorithm, a network is split in half, directly from the source to destination, forming the line segment sd where s is the source and d is the destination. Messages are always forwarded to the neighboring node to the right along the boundaries of the initial network face F_1. Once the message has reached a node that would forward across sd at p_n, it would then forward to the neighboring node to the right along the boundary of the next network face F_n. This process would repeat until the message is delivered to the destination. This algorithm does not perform well when nodes are highly mobile, although it does overcome the local minimum problem that arises from LOS obstruction. If a network face

F veers too far to the right, there is the potential of looping back on itself. Figure 2.8 illustrates the basic concept of face routing.

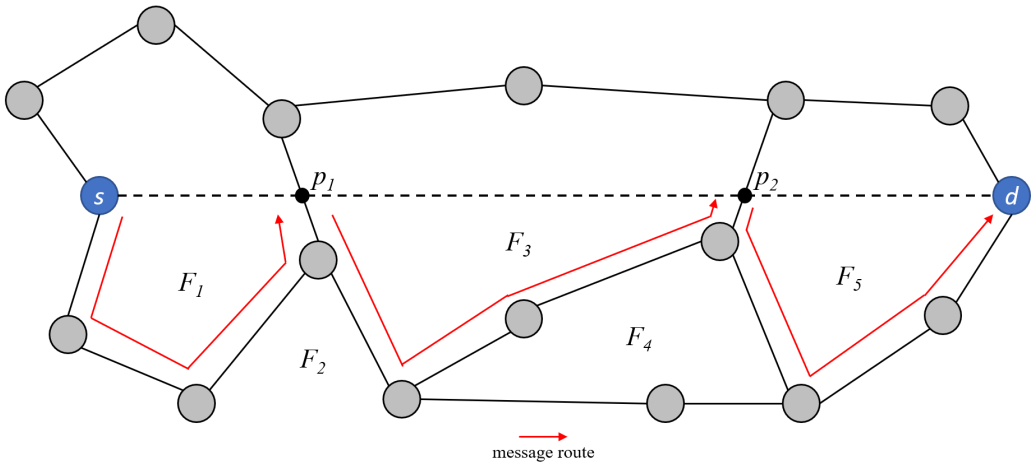


Figure 2.8. Example of Message Forwarding in Face Routing. Adapted from [62].

Greedy Forwarding

Greedy Forwarding is a form of flooding that incorporates the minimum angle between the source's neighbor and the destination. The strategy makes the distinction between the Most Forward Progress (MFP) and closest distance when making routing decisions. It always forwards messages to the node that is closest, regardless of whether that node has a path to the destination or not. Because of this, it fails to prevent the local minimum problem and has the potential to loop back on itself, especially when nodes participating in the DTN are constantly moving. Methodologies have been incorporated into this strategy by merging face routing and creating a recovery mechanism from the local minimum. The Greedy-Face-Greedy strategy takes advantage of the characteristics greedy forwarding provides by getting the message as close to the node as possible while providing a backout option in case a dead end is reached [42]. Figure 2.9 depicts the differences between face routing, MFP, and greedy strategies where s is the source and d is the destination. A, B, and C represent relay nodes where each strategy's source s node forwards messages.

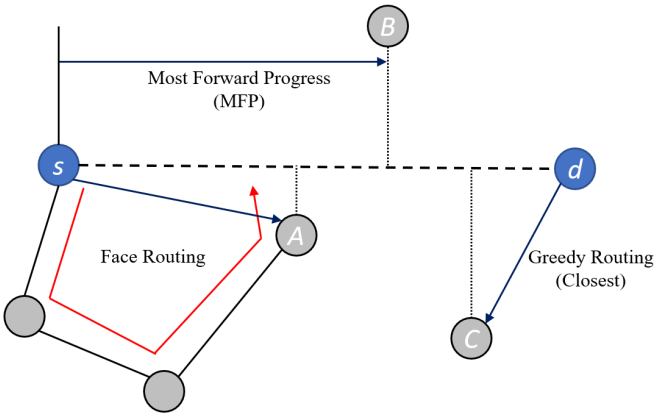


Figure 2.9. Differences between Face, MFP, and Greedy Routing Strategies. Adapted from [42].

2.9.2 DTN Protocols

Numerous protocols in the wild are currently being researched and fielded as viable solutions to the challenges DTNs bring to networking. Several prominent and promising protocols have been studied at length and continually improved by the merging of methodologies and strategies. The following protocols are essential because they cover the spectrum of DTN protocol methodologies and have proven performance metrics in both simulation and real-world applications.

Direct Contact

As discussed, direct contact is the least likely of flooding strategies to be employed in DTNs. It has precisely one relay, the destination, and only has one path between the source and destination. Because of its simplicity, it does not require significant resources to deliver its messages. Researchers have proven its potential in the Infostation architecture by proposing direct contact delivery between nodes and gateways to increase throughput and decrease costs. This only works with minimal participation within a DTN because as more nodes join, the capacity quickly drops to zero [37].

Two-Hop Relay

The two-hop relay methodology uses a modified flooding strategy where the source sends message replications to the first set of n nodes it comes in contact with. Both the source and

the relay then hold the message until they are within range of the destination and then deliver it. Because there are now $n+1$ copies of the original message in the network, more storage is used, and bandwidth is consumed. If the assumption that each node contacts the destination separately with probability p, then each message will be delivered with a probability of $1-(1-p)^{(n+1)}$. By increasing the number of copies, there is the potential for decreased latency across the network because if any of the $n+1$ nodes reaches the destination, the message is delivered. The local minimum problem still exists if any of the $n+1$ nodes never have contact with the destination. This approach has shown potential in sensor networks and has been proposed as a fallback strategy when ad-hoc routing cannot determine a path to the destination [37].

Tree-Based Flooding

Tree-based flooding extends the two-hop relay methodology by shifting the task of message replication to the relays. When a relay receives a message, the message comes with data that directs the number of replications that relay is to make. As each node replicates the number of pre-coordinated messages, hop-by-hop, it forms a tree from the source. Messages can either be restricted to a maximum number of n hops from the source preventing new branches from spawning or allowed to laterally replicate at the same level of the tree, limiting the number of copies to a maximum of $\sum_{i=0}^{n} m^n$. Additionally, one method is to distribute half of the replications to relays, while the source keeps the other half. Either way, tuning the distributions and message replication counts can be tricky, and this methodology requires constant adjustment as nodes enter or exit the DTN [37].

Epidemic

Vahdat and Becker proposed Epidemic routing in their paper *Epidemic Router for Partially-Connected ad-hoc Networks* in 2000 [5]. Sometimes referred to as the *Gossiping Protocol*, it was modeled after the spreading of pandemic diseases faced by populations in real life. The idea was that the node with message m is identified as patient zero, and any relay that it encounters not already having the message is infected with message m. Through maximum message replication, the probability of message delivery is elevated, and by doing so, the network latency is decreased at the same time. When two nodes are within range of each other, there is an exchange of summary vectors that identify which messages are present in the other's buffer. Figure 2.10 depicts the exchange between a source and a relay. If the relay node does not have the message in question, the source node then copies the message

and sends it to the relay, infecting it. This process repeats until the message is eventually delivered at infection denoted by the envelope in the third phase [5].

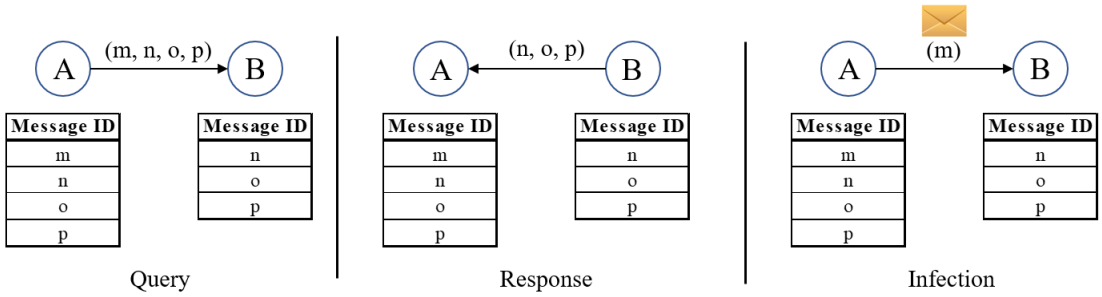


Figure 2.10. Example of Message Infection by Epidemic Routing. Adapted from [5].

In addition to the initial communication and subsequent message replication, Vahdat and Becker associated three properties to each exchange. First is an unique message identifier. Second is a hop count that is set to the TTL field ensuring that the message eventually drops if the hop count is exceeded. Last is an optional acknowledgment issued by the source to the destination once it has successfully received the message [5]. The optional acknowledgment request is often not used in DTN because it would incur additional latency on the network, but it is used in some cases as a clean-up mechanism. Because the increased size of node participation with *Epidemic* routing would quickly saturate the network, especially if an acknowledgment request was made, recovery schemes, such as *immunity* and *vaccine* have been proposed by researchers [63], [64]. The concept provides a solution to the continual replication of messages by relays after the message has successfully been delivered. If a node receives the immunity or vaccine, the node replicates it through the network telling the other relays to stop message replication, and prevents them from being infected by the same message again in the future.

Spray and Wait

Spray and Wait (SnW) implements characteristics of *Epidemic* but imposes a waiting period. Spyropoulus et al. [38] proposed SnW where a maximum limit on the number of possible message replications is fixed. SnW is broken into two distinct phases. In the spray phase, a message is spread to at most L different relays. In turn, each relay then sprays the message to L relays. There are different variations of spraying. First, similar to *Epidemic*, the source

node sprays only to those nodes which do not already have the message. If the source encounters the destination at any point before infecting more of the network, the spraying phase effectively terminates. The second varyiation is the binary spraying technique. This technique differs in that the source node initially has L copies of the message. If any node, including the source, has n copies of the message, $L \geq n > 1$. If this node encounters another node with no copies of the message, then the source exchanges $\lfloor n/2 \rfloor$ copies and retains $\lceil n/2 \rceil$. In the wait phase, if the destination node was not encountered during any part of the spray phase, the relays carrying the messages attempt a direct delivery. This method of SnW was optimum and reduced expected delays across the network. Spyropoulus et al. observed that SnW brings both the speed of *Epidemic* and the simplicity of direct delivery strategies [38].

Probabilistic Routing Protocol Using History of Encounters and Transitivity (PRoPHET) PRoPHET is a greedy algorithm, where a source forwards a message to a relay only if the relay has a higher probability of encountering the destination than itself. It uses the idea of *delivery predictabilities (DP)*, or the the chance of one node encountering another in the DTN. Every node using the PRoPHET protocol keeps a table of probabilities of encountering every other node within the network. If two nodes fail to encounter each other over a period, the values in the tables decrease appropriately. The ability of PRoPHET to update the DP is governed by the following three equations.

$$P_{(a,b)} = P_{(a,b)old} + (1 - P_{(a,b)old)} \times P_{init} \tag{2.2}$$

$$P_{(a,b)} = P_{(a,b)old} \times \gamma^k \tag{2.3}$$

$$P_{(a,c)} = P_{(a,c)old} + (1 - P_{(a,c)old)}) \times P_{(a,b)} \times P_{(b,c)} \times \beta \tag{2.4}$$

$P_{(a,b)}$ and $P_{(a,b)old}$ indicate the current and previous DPs of node A for another node $B, P_{(a,b)} \in [0, 1]$, where $P_{init} \in [0, 1]$ is an initialization constant. As time progresses and nodes do not encounter each other, the parameter k tracks the time units expired since the last update of the aging constant $\gamma \in [0, 1)$. The parameter $\beta \in [0, 1]$ then scales to control transitivity affecting the DPs. Transitivity can be explained as follows. If node A frequently encounters B, which in turn frequently encounters C, then by transitivity, A can send messages directly to B, knowing that there is a high probability that C will receive its message [39]. Note that PRoPHETv2 has been proposed to address problems with the DPs

not adapting to reflect human movement patterns [65].

MaxProp

MaxProp utilizes historical encounter data to aid in making routing decisions. As each node conducts *Epidemic* flooding of the message, it is complemented by connection history to prioritize with newly established neighbor connections, or drops message bundles if its buffer is full [40]. Each node A maintains a table of probabilities $f(A,i)$ for every other node i within the network. $f(A,i)$ is initialized to $\frac{1}{N-1}$, where N is the size of the network. When connecting with a neighbor, node A increments its f value by 1 and then subsequently normalizes the table so that all probabilities $f(A,i) = 1$. Once the table normalization is complete, all messages are then forwarded to the neighbor node, and table values are exchanged. As with *Epidemic* routing, acknowledgments are then flooding across the network, thus removing any successfully delivered messages. MaxProp also tracks message hop counts, and if any messages have not exceeded a threshold t, that threshold is set to the minimum hop count of the first message p in the buffer. MaxProp implements three drop strategies to avoid running out of buffer space. First, all messages that have been acknowledged are dropped. Second, any message that exceeds the hop-count threshold t, and has a probability of delivery below $f(A,i)$, is dropped. Lastly, a message with the lowest hop-count is dropped last [40].

Resource Allocation Problem for Intentional DTN (RAPID)

Instead of focusing on encounters, Balasubramanian et al. treated routing decisions as a resource allocation problem. The protocol explicitly called out the optimization as a routing metric. Through a control channel, RAPID exchanges various metadata, including expected contact time with other nodes, message delivery lists, and the average size of previous message transfer [41]. A message i is replicated in a way that locally optimizes the protocol's utility function $\delta U_{i/s_i}$ where U_i is the contribution of the message to the routing metric and s_i is its size. RAPID uses three components to achieve this. The *Selection Algorithm* is implemented when two nodes encounter each other and exchange their metadata. They then deliver any messages according to the utility functions in decreasing order. The relay node then carries the remaining messages to the destination. The *Interference Algorithm* is used to determine the utility of a message for the routing metric. Finally, the control channel distributes metadata from past encounters with the most up to date, recent, and accurate data [41].

Encounter Based Routing (EBR)

Nelson et al. proposed Encounter Based Routing (EBR) in an attempt to balance high delivery ratios with low overhead [66]. The premise of the protocol revolves around its ability to predict the future rate of node encounters through past metadata. They suggest that nodes that experience more encounters are more likely to successfully deliver a message to the destination, than are those which have fewer encounters. They provide examples like disaster recovery networks where first responders bridge gaps in network clusters, and vehicle-based networks where vehicle paths typically traverse well-traveled routes [66]. The protocol itself is coined to be a quota-based protocol that limits message replications. Nodes running EBR maintain past encounter rates to determine the number of messages that are to be replicated and exchanged. EBR achieves this by maintaining two pieces of local information. The first is the encounter value (EV) and the second is the current window counter (CWC), where EV represents the past rate of encounters, and the CWC tracks the number of encounters in a specified time interval. Updates are calculated using Equation 2.5.

$$EV \leftarrow \alpha \times CWC + (1 - \alpha) \times EV \tag{2.5}$$

α is defined as the popularity counter weighting constant that is set specifically to the model the protocol is running. The EV places emphasis on α by using the most recent capture of the CWC. The CWC is incremented at every encounter. If the current window update interval has expired, then the CWC is reset to zero. Nelson et al. observed that the EBR protocol produced comparable or somewhat increased performance over other flooding protocols, while using minimal node resources [66].

CenterMass

A continuation of Centroid (described in Section 2.10), CenterMass encompasses all the same mechanisms of Centroid but adds the mechanism of directional message forwarding in the direction of the destination. The nodes achieve this through the retention of a list of the Centroids for every neighbor a node encounters. At each encounter, these CenterMass Centroid lists are merged where only new undiscovered Centroids are appended. The key difference here is that messages are only replicated and transmitted if, and only if, the neighbor's Centroid is smaller than its own Centroid distance to the destination. This forces forward progression of replicated messages, without creating a replicated message

pandemic [11].

2.10 DTN Protocols of Study

As discussed previously, this thesis is focused on four geographic routing protocols that will each have PNT positional error artificially applied through simulation. Each simulation with introduced positional error will then be compared to a baseline of itself, (i.e., no introduced error), and then against the other four protocols. Through a repetitive process of re-engineering each protocol and rerunning the simulation, we can determine the protocol best capable of handling PNT error effects, provide the highest probability of message delivery, MDRs, the lowest latencies, and the best bandwidth performances metric.

2.10.1 Vector

The Vector Routing protocol, as proposed by Kang and Kim [10], attempts to negate the negative effects of *Epidemic* message flooding incurred over a DTN. The protocols assumes that the mobility patterns are unpredictable, certain nodes have no neighbors, and that every node knows its location through positional data provided by GNSS PNT. By obtaining its coordinate location every $\triangle t$ seconds and previous coordinates from a $\triangle t$ seconds prior, the vector V_{cur} is calculated. The premise is that V_{cur} can predict the relative direction of movement by exploiting the history of the mobility of the nodes. It then compares its location with the location of a neighbor node's V_{cur}. Message routing is then calculated based on each node's direction and velocities.

$$nFwd(x) = \gamma f(|\theta_x - \theta_y|) + (1 - \gamma) g(|d_x - d_y|) nData(x) \tag{2.6}$$

Equation 2.6 describes how nodes determine the number of messages to forward to the other nodes, where $nFwd(x)$ is the number of messages tagged for replication, θ_x and θ_y are the direction of node movement, d_x and d_y are their velocities , $nData(x)$ is the number of messages node x has its buffer, $f(\theta)$ is the normalized function of θ, and $g(d)$ is the normalized function of d.

Kang and Kim's protocol provided similar MDRs compared with the *Epidemic* routing protocol with 38.5% less traffic on average. They observed an average of 0.47 hops less than

Epidemic routing protocol. Vector routing is subject to orthogonal errors [11] that are not easily overcome.

2.10.2 Centroid

Centroid also uses location information when making routing decisions. Proposed by Rohrer [11], the novel geographic routing primitive, Centroid, calculates the center of mass based off the history of a node's position instead of relying on the accuracies of GNSS PNT positional data. The protocol strictly looks at the points of the node trace itself, which is not necessarily a highly dense center of mass and is calculated as

$$C_x(t_p) = \sum_{t=1}^{t_p} \frac{C_x(t-1) \times (t-1)}{t} + \frac{x_t}{t} \tag{2.7}$$

where C is the Centroid, x is the $X, Y, or Z$ component of a node's position, and t_p is the current time [11]. The time increments is a parameter that can be modified and determines the delta between the old and new Centroid and is calculated using the Equation 2.8.

$$C_x(t_p) = \frac{x_{t_p} - C_x(t_p - 1)}{t_p} \tag{2.8}$$

Because location history is averaged out over time, positional data errata is assumed to be negligible in Centroid [11] routing and is observed in our findings in Chapter 4.

2.10.3 Geolocation Assisted Predictive Routing (GAPR)

Proposed by Rohrer et al. [14], GAPR inherits the benefits of existing geolocation routing protocols while leveraging the ability to unlearn encounter probabilities when they are flawed. The fundamental difference in GAPR compared to other geolocation protocols is that it utilizes positional data to determine state changes and not to make routing decisions. Further, GAPR implements P_{direct}, the delivery probability of each store message in its buffer. To that end, when two nodes encounter each other, they then exchange their geolocation data along with a time stamp. Each node then records this metadata into a table that is then exchanged each time it encounters a new node. This allows for the detection of network changes when a neighbor has moved from its last position. Node A records the historical location of node i as $l_{old}(i)$ along with the timestamp of $T_{old}(i)$. When node A learns that

i has moved, it updates its table with $i, l_{new}(i)$ and the time stamp $T_{new}(i)$ from another neighboring node. Node A is now capable of detecting a movement of i and resets P_{direct} for i based on the following:

$$Distance(l_{new}(i), l_{old}(i)) > \alpha \times R \tag{2.9}$$

$$T_{new}(i) - T_{old}(i) \leq T_0 \tag{2.10}$$

where R is the transmission range, α is a tune-able parameter set to 1 by default, and T_0 is another tuneable parameter. Nodes not only maintain a location table containing i nodes geolocations and time stamps, they also maintain a transitive location table containing the same metadata learned from its neighbors about the i nodes [14]. Rohrer et al. observed similar results when compared to MaxProp, PRoPHET, and *Epidemic*.

2.10.4 Geolocation Assisted Predictive Routing (GAPR)2

A hybrid of Vector and GAPR methodologies, GAPR2, was proposed by Killeen [12] as a way to improve the efficiency of GAPR by removing as much overhead as possible. This was achieved by rearranging message and priority delivery mechanisms through the use of Vector's message limiting calculation as defined in Equation 2.4 above. By combining the geolocation effectiveness of GAPR and the limits on replications Vector brings, Killeen observed a five-fold drop in replication overhead when compared to GAPR in an urban mobility simulation. Note, that although overhead was decreased, the message delivery ratio was 10% lower than that of GAPR [12].

2.11 DTN Security

DTNs must incorporate security into their architectures. Attacks, even from a single attack vector, can drastically degrade performance in mobile networks. Because DTNs fundamentally rely on participating nodes acting legitimately on the network, the ability for adversaries to take advantage of potential DTN vulnerabilities is heightened. Nodes that act as "black holes" advertising optimal metrics to collect all messages on the network, or inundate the network with messages as a "flooder," can quickly cause a Denial of Service (DOS) on the network. Security mechanisms like distributed certificate authorities and message encryption are some implementations currently being fielded into DTN protocols [67]. Although

not specifically related to this thesis, it is essential to understand the importance of securing DTNs.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methodology, Design, and Simulation

In this Chapter, we introduce our methodology, design, and simulation parameters. Specifically, we discuss simulation options currently available that have been used in past DTN research and follow with our reasoning as to why we choose ns-3 simulator for this work. We then lay out our simulation design by introducing different mobility scenarios and provide specifics on how we incorporated PNT positional error into the four protocols previously discussed in Section 2.10. We close the Chapter by discussing the parameters that we coded and implemented during each simulation and the metrics we used to analyze the results.

3.1 Simulation Options

The ability to capture all the intricacies of network characteristics through a simulator is challenging and typically requires massive amounts of computing power to run complex simulations. To truly capture network characteristics that reflect network operations in the wild, researchers would need to set up a real-world DTN that incorporates all layers of the network stack. To do so, especially in the initial phases of research, is not realistic and would be too costly. However, it is often not necessary to capture every aspect of the network stack, as most simulations conducted are looking at specific characteristics and layers of networking operations. Therefore, simulation provides insight into the characteristics of specified problems and phenomena that researchers are studying. In the following subsections, we discuss network simulation tools and the pros and cons of each.

3.1.1 Opportunistic Network Environment (ONE)

The ONE is an agent-based discrete-event network simulator coded in Java that uses a step engine to increment traffic over a DTN. Introduced by Keränen et al. [13] in 2009, the ONE specifically evaluates DTN routing and application protocols. It allows researchers to create real-world scenarios based off several mobility models and procedures and provides a visualization interface that tracks node movements in real time. As depicted in Figure 3.1, the ONE operates using four distinct functions. The *movement models* provide synthetic models or the import of existing trace files that can be simulated. The *event generator* creates the

messages that are used inside the simulation. The *Routing* function is implemented by specified mobility models and makes routing decisions within the DTN. Last, the *visualization and results* function aggregates data points into a visual representation that is presented to the user in a Graphical User Interface (GUI).

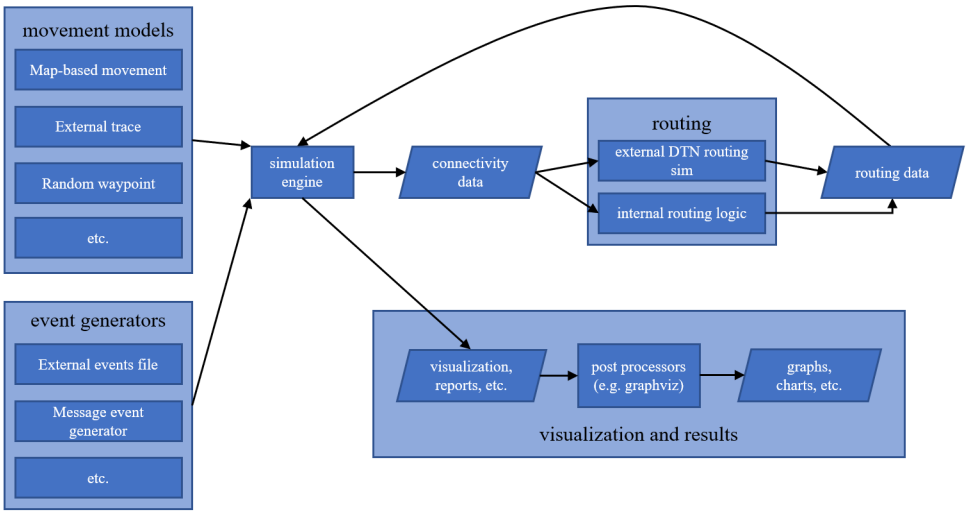


Figure 3.1. Overview of the ONE Environment. Adapted from [13].

The ONE implements movement capabilities through mobility models. Popular mobility models include `RandomWalk` and `RandomWaypoint` but also include map-based models that are used to produce the visualization output in the GUI. The ONE also comes preloaded with map data of the Helsinki downtown area that the map-based models use to present node movement. Other maps can be uploaded into the ONE but input of Geographic Information System (GIS) data must be manually configured [13].

The ability to incorporate social relationships through node communities is also available inside the ONE. The grouping of nodes that generally stay together for a more extended period, (e.g., cellphones of individual members of a family), can be modeled. Further, the random nature and variety of nodes participating in DTNs can range from a person walking on the sidewalk, to a high-speed vehicle on a freeway, and are captured in the ONE mobility models [13] providing for a more realistic modeling experience for researchers using the tool.

Because the focus of the ONE simulator is to model the behavior of DTN Store-Carry-Forward paradigms, the creators have deliberately abstracted away the lower levels of the network stack [13]. Researchers strictly interested in looking at DTN and application protocols may find the ONE simulator useful in their studies; however, if simulations require a complete look at the characteristics of protocols at all layers of the network stack, other simulator options may be better. Note, a paper published in 2014, discussed ways to implement an *EnhancedONE* architecture that expands simulation into the layers that the ONE abstracts away [68], thus increasing the simulators ability to capture more network characteristics.

3.1.2 Network Simulator 3 (ns-3)

ns-3 is a discrete-event network simulator implemented in C++ and Python that allows scripting, making multiple simulations easier to run. It is built to enable the static or dynamic calling of libraries and user-defined modules, allowing for increased flexibility. Because ns-3 simulates the entire network stack, its use in the research of DTN has been effective in understanding the underlying functionality of traditional network protocols and prototypes created by researchers. Further, ns-3 programs may access all of the Application Programming Interfaces (APIs) directly or make use of the helper API module that encapsulates lower-level API calls. Because ns-3 utilizes a Command Line Interface (CLI), additional modules must be incorporated and called to provide an interactive GUI. Simulation of Urban MObility (SUMO) and NetAnim are two such modules that help researchers visualize their simulations [15], [69].

Figure 3.2 depicts the software organization of ns-3. The core of the ns-3 organizational stack is the base of the simulator housing elements of the simulator itself, schedulers, time, and events. The network layer incorporates elements such as packets, sockets, address types, nodes, and network devices. The next two layers of the stack are the elastic elements where both traditional and prototype protocols can be inserted and executed. The mobility model is particularly important because it serves to add realistic implementations of nodes and devices on the move in the environment. Additionally, elements such as the number of source and sink nodes, antenna transmit and receive gains, and bandwidth capacities can be modified in these two layers allowing for researchers to modify parameters as required. The helper layer encapsulates the lower levels in order to script unique scenarios and protocols.

Last, the test layer aids researchers in debugging both ns-3 installation and implementations without inadvertently corrupting the core and network layers [70].

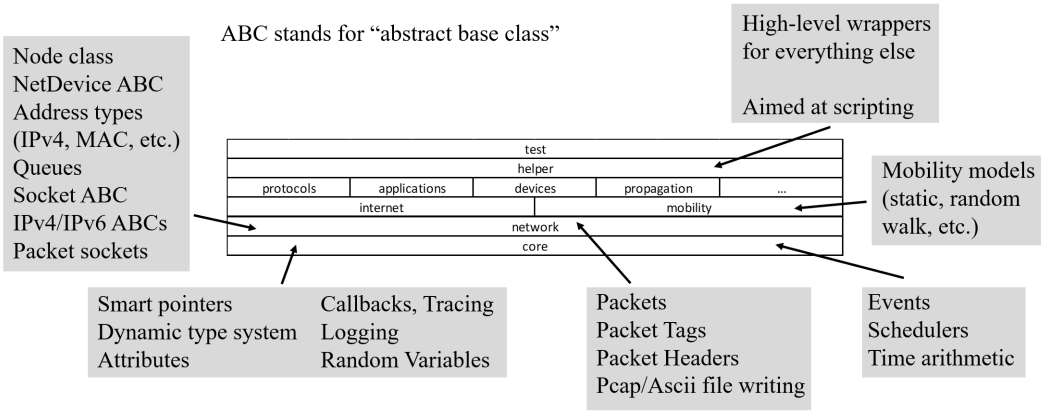


Figure 3.2. Software Organization of *ns-3*. Adapted from [70].

Similar to the ONE simulator, ns-3 contains a multitude of modules that allow the incorporation of location information. Modules, such as the `geographic-position.cc`, can ingest simulated Cartesian coordinates and overlay them into a 2-d or a 3-d Mercator map. Additionally, the `random-variable-stream.cc` module provides numerous distribution models that provide realism of node movement when complimented with different mobility models. The mobility models inside ns-3 consist of `ConstantPosition`, `RandomWaypoint`, `RandomWalk2d`, `RandomWalk3d`, and `RandomDirection` (among others) each providing for varying characteristics of nodes on the move [70].

ns-3 produces a variety of outputs, and each can be coded into whatever simulation a researcher is running. Of particular use is a tracing capability that creates `.csv` and `.tr` traces files that can then be used to plot graphs. ns-3 also allows for American Standard Code for Information Interchange (ASCII) `.mob` trace files capturing node-by-node information exchanges [70].

Since ns-3 is a discrete-event simulator and executes all layers of the network stack, it takes significantly more time to run simulations compared to the ONE simulator. However, ns-3 provides more fidelity and information on the protocols being run and is useful if performance limitations of the lower levels of the network stack are a significant factor in the overall simulation performance. Further, since the ONE simulator abstracts the lower

levels of the network stack, more experimentation is involved in fielding realistic protocols before they can be deployed.

Last, for ease of parsing output files of ns-3 simulations, the marrying of C++ and Python allows researchers to easily run post-processing scripts to aggregate the potentially massive files that are created. This was useful in this thesis, as is apparent from the data snippets in the Appendices. The ability to use scripts significantly reduces the time spent on data input and analysis and is well suited for research in the field of DTNs.

Even though it lacks organic visualization interfaces, the ability to add additional modules, and the flexibility to modify the source code to meet the needs of our research, we felt that ns-3 was the preferred option over the ONE. Further, by not abstracting away layers of the network stack, ns-3 provides a more realistic simulation experience over other simulation tools available.

Simulation of Urban MObility (SUMO)
SUMO is an open-source package that can be incorporated and run from ns-3 through the use of the `Ns3MobilityHelper` [69]. It works both in an offline and online mode that simulates traffic simulation overlay on top of a map. The intent is to simulate new traffic strategies in simulation before they are used in real-world applications. It is mostly used in traffic forecasting, traffic light evaluations, and route selections in vehicular communication systems [69].

NetAnim
NetAnim is a software package that is part of the ns-3 baseline package. It provides a visualization interface that allows researchers to see the interconnections between participating nodes. It is best used in mobile scenarios to understand node contact opportunities. The package enables the user to freeze the simulation to inspect elements of data exchanges at a given moment. Further, it adds range bubbles that represent the radius of node signal penetration [15].

3.1.3 Comparison of ns-3 and the ONE

Recent research conducted at NPS observed up to 33% of network data transmitted is due to control messaging when implemented in ns-3, suggesting that it provides a more realistic

portrayal of network characteristics over the ONE [71]. Conversely, simulations run with ns-3 often take up to 50 times longer to complete because they do not abstract away the lower levels of the network stack [71]. The ability to export output data files and aggregate them using Python, compensates for the lengthy simulation run time. Considering these facts, ns-3 was the preferred simulator for our research.

3.2 Mobility Scenarios

Several mobility scenarios that have been incorporated into ns-3 simulation and testing. Killeen implemented both the Helsinki and Bold Alligator scenarios (described at length below) in the ONE simulator [12]. Mauldin implemented the Omaha scenario in the ONE and subsequently implemented Killeen's Helsinki and Bold Alligator trace files, along with the trace file of the Omaha scenario, into ns-3 [71]. These scenarios are characteristically different and provided diversity within our study. The Helsinki scenario is of particular interest because it has been used to study DTNs for some time and has been cited in the literature since Keränen et al. released their seminal paper on the ONE simulator [13]. Both the Bold Alligator and Omaha scenarios provide insight into military scenarios that represent potential deployment environments for DTN routing.

3.2.1 Helsinki

The Helsinki scenario is modeled after the city of Helsiki, Finland. It has been used as the base model for simulations in the ONE since the tool has been available. This is mostly because it takes significant time to build new city models and requires additional software to translate the coordinate systems that can be ingested by the ONE [13]. In this thesis, we used the standard Helsinki scenario that Mauldin transcribed into ns-3 [71].

The Helsinki scenario models real-world mobility of DTN nodes as they traverse city sidewalks, roads, and tramway systems. Figure 3.3 depicts the paths nodes in the model follow. The model defines restrictions on where a particular node can travel to ensure consistency and realism. Specifically, trams only operate on tramways, vehicles on roads, and pedestrians follow sidewalks.

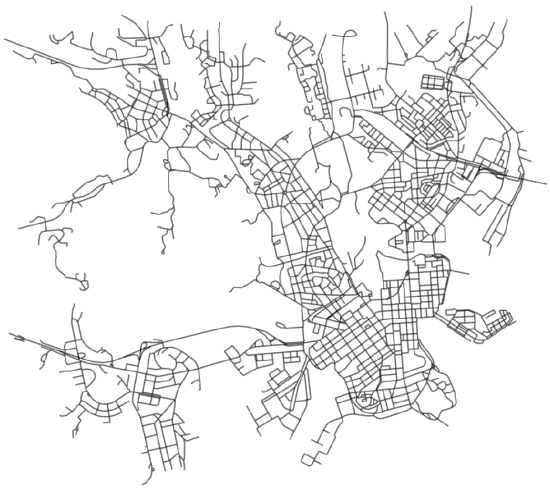


Figure 3.3. Helsinki Simulation Area Map. Source [13].

Of particular interest is the warmup time of 1000 s that allows routing primitives to converge before traffic is sent out across the network. This is important because if not enough time is given to allow routing to settle, messages sent early in the simulation will likely be dropped, thus changing the results. The model employs two types of radios. The first, a base radio, for pedestrians and vehicles that utilize the 802.11g at 2.4GHz wireless protocol. The second is the tram radio that is about ten times faster than the base radio while implementing 802.11ac at 5GHz.

The Helsinki scenario uses a simple range propagation loss model. The loss model simulates the ranges of each radio, with greater signal closer to the transmitter and weaker the further away. If two radios come in contact with each other's signal "bubble," and the signal has not degraded to the point of being unusable, the nodes can communicate and exchange data. The transmission and receive gain levels can be modified to simulate higher-gain antennas, extending the range of the radios.

The Helsinki model also has a variable message buffer parameter, which is important in DTN research. Because of the Store-Carry-Forward paradigm, participating nodes have finite buffer space and must have mechanisms in place to determine how to manage them. Various techniques, such as First In First Out (FIFO), are often implemented in buffer management algorithms.

Because we are interested in protocol performance when artificial positional error is introduced but otherwise want our results to be comparable to prior work using these same protocols and mobility scenarios, we implemented the same testing parameters used in previous studies [12], [71], [72]. Table 3.1 outlines the default parameters used in the Helsinki mobility scenario.

Table 3.1. Parameters of the Helsinki Mobility Scenario. Adapted from [12], [71], [72].

Parameter	Values
Simulation Duration	12 hrs
Simulator Seed	717,718
Warmup Time	1000 s
Timestamp Resolution	0.1 s
Beacon Interval	5 s
Number of Pedestrians	80
Number of Cars	40
Number of Trams	6
Base Radio Bandwidth (Mbps)	54
Tram Radio Bandwidth (Mbps)	526.5
Base Radio Transmit Range	10 m
Tram Radio Transmit Range	1000 m
Base Buffer Size (MB)	5
Tram Buffer Size (MB)	50
Message Rate	1/25-35 s
Message Size	0.5-1.0 MB
Message TTL	5 hrs
Hop Limit	50
Protocols	Epidemic, Centroid, GAPR, GAPR2, Vector

3.2.2 Omaha

The Omaha scenario, programmed by Mauldin [71], is modeled after Operation Neptune, the amphibious landing portion of World War II (WWII)'s overarching operation, Operation OVERLORD, commonly known as "D-Day" that took place on the beaches of Normandy, France on June 6th, 1944. The scenario itself breaks down the participating nodes into three unique radio types. The first are unmounted soldiers traversing the paths ashore. The second are the ships at sea moving inside their patrol boxes. The third are the troop transports that

move along the beach between the ships and the soldiers ashore. Note, both the soldiers ashore and the transports have the same speed movements to account for the troops when they are both on the transports and after they disembark. Further, all the node types use the same radio parameters [72]. Figure 3.4 depicts the Omaha scenario map and associated paths of each node type.

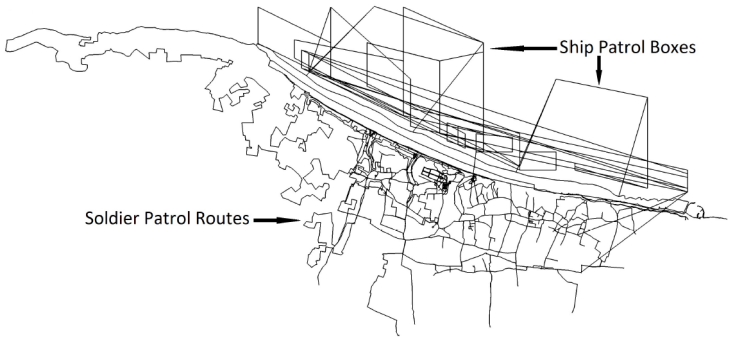


Figure 3.4. Omaha Simulation Area Map. Source [71].

Originally coded in ONE, the Omaha scenario required several third party software applications to convert map data into an ingestable format that ns-3 could then run. Part of Mauldin's research took the trace files outputted by the simulation in the ONE and imported them into a running ns-3 mobility model [71]. The scenario parameters, outlined in Table 3.2, consist of 61 nodes with various configurable parameters. Because of the scale and size of the real invasion, the scenario drastically scales back the number of participating nodes by grouping the same set of nodes together. Further, the speeds of the nodes are never the same, (i.e., they adjust speed as they randomly move about the map). These parameters determine when and where nodes contact one another and exchange messages [71].

The Omaha radio configurations are similar to those of the Helsinki scenario. The radios implement an 802.11g at 2.4GHz wireless protocol for signal propagation. Further, it implements the same range propagation loss model to simulate realistic radio transmitted ranges. The scenario allows for configurable buffer sizes.

As with the Helsinki scenario, the parameters are static for the purpose of this thesis since we were only interested in how the protocols we studied perform when the positional signal error is simulated.

Table 3.2. Parameters of the Omaha Mobility Scenario. Adapted from [12], [71], [72].

Parameter	Values
Simulation Duration	12 hrs
Simulator Seed	717,718
Warmup Time	1000 s
Timestamp Resolution	0.1 s
Beacon Interval	5 s
Number of Soldiers	44
Number of Ships	17
Radio Bandwidth	12 Mbps
Radio Transmit Range)	550 m
Solider Node Buffer Size (MB)	5
Ship Node Buffer Size (MB)	50
Message Rate	1/25-35 s
Message Size	0.5-1.0 MB
Message TTL	5 hrs
Protocols	Epidemic, Centroid, GAPR, GAPR2, Vector

3.2.3 Bold Alligator

The need for an effective ah-hoc network that is capable of providing reliable communications and IP services in a humanitarian support mission is something both the United States Navy (USN) and the United States Marine Corp (USMC) have been interested in for some time. The Bold Alligator exercise was initially envisioned to revitalize the USN and USMC capability to conduct major amphibious operations from the sea while conducting non-combatant evacuation exercise [73]. As such, it has since been used by researchers as an opportunity to study how effective DTN strategies perform in constrained military operations.

The exercise consists of a multi-national coalition postured against a fictitious rogue nation, Garnet, who has begun to invade the ally nation of Amberland [12]. The primary objective of the scenario is to conduct Noncombatant Evacuation Operations (NEO) while maintaining a robust security posture that allows Amberland to reestablish its border sovereignty. The entire exercise is backed by three USN ships and a Marine Expeditionary Unit (MEU) built for about 10,000 personnel [73].

Originally coded and simulated in the ONE by Killen [12] and exported into ns-3 by Mauldin [71], the scenario used a compliment of node types ranging from the unmounted Marine on the shore, Landing Craft Air Cushions (LCACs) moving from ship to shore, Humvees on the roads, and ships at sea. Although similar to the Omaha because it is also an amphibious operation, it differs in its size and scale, and the increased amount of node mobility built into the simulation. Figure 3.5 depicts the scenario map as participating nodes move in.

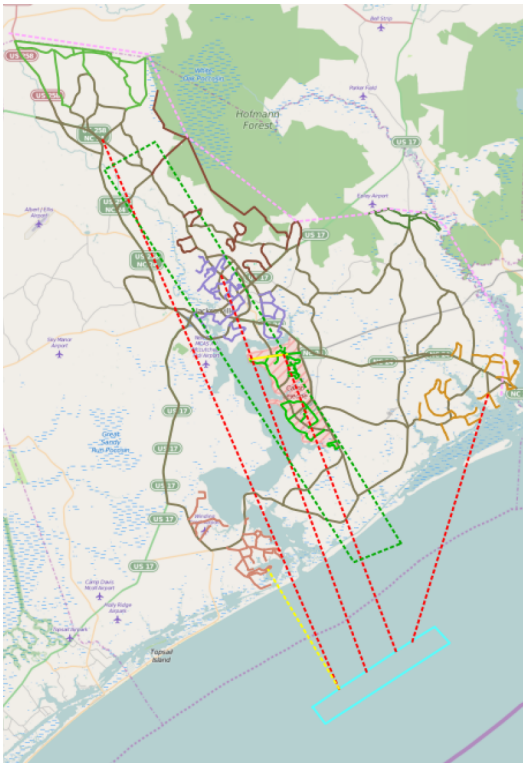


Figure 3.5. Bold Alligator Simulation Area Map. Source [12].

The ground element of the Bold Alligator scenario consists of Marine platoons and Humvees, that traverse the roads and the inner urban portions of the fictitious city. The paths themselves are depicted as the brown, green, and dark green solid lines, as shown in Figure 3.5. The drones follow the dotted pink line in the upper Northeast portion of the map, and the helicopters follow the red dotted lines from the ships at sea to evacuation points where Marines on the ground are staged with evacuees. The yellow dotted line reflects the path LCACs traverse to and from the ships at sea [12]. The scenario provides a real-world

complex DTN that can be simulated using various protocols.

Table 3.3. Parameters of the Bold Alligator Mobility Scenario. Adapted from [12], [71], [72].

Parameter	Values
Simulation Duration	24 hrs
Simulator Seed	717,718
Warmup Time	10800 s
Timestamp Resolution	0.1 s
Beacon Interval	5 s
Number of Marines	70
Number of Humvees	20
Number of Drones	2
Number of Helicopters	8
Number of LCACs	2
Number of Ships	3
Base Radio Bandwidth	12, 24, 36, 54 Mbps
Humvee Radio Bandwidth	54 Mbps
Drone Radio Bandwidth	6 Mbps
Ship Radio Bandwidth	54 Mbps
Base Radio Transmit Range	100 m
Humvee Radio Transmit Range	3000 m
Drone Radio Transmit Range	3000 m
Ship Radio Transmit Range	10000 m
Marine Node Buffer Size (MB)	5
Drone Node Buffer Size (MB)	5
Humvee Node Buffer Size (MB)	50
LCAC Node Buffer Size (MB)	50
Helo Node Buffer Size (MB)	50
Ship Node Buffer Size (MB)	500
Marine Message Size	250-500 KB
Humvee Message Size	0.5-1.0 MB
Ship Message Size	0.5-1.0 MB
Marine Message Rate	1/5-10 s
Humvee Message Rate	1/10-20 s
Ship Message Rate	1/25-35 s
Message TTL	5 hrs
Protocols	Epidemic, Centroid, GAPR, GAPR2, Vector

Again, even though there are many configurable parameters in this scenario, and because we were only interested in the performance of the protocols specific to this thesis when

artificially introducing positional error, we used the default parameters outlined in Table 3.3 above. Because geographic routing protocols that are less dependent on the terrestrial aspects of networking take their inputs from space to aid in their message routing decisions, capturing performance metrics in a scenario such as this, provides the potential to strengthen military communication capabilities when contested or constrained and are directly pertinent to this thesis.

3.3 PNT Positional Error Introduction

Based on Table 2.4 of Chapter 2 and applying a normal distribution as described in Section 2.7 of Chapter 2 inside the ns-3 architecture to artificially adjust positional inputs, we were able to observe geographic routing protocols in real-time. To do this, we fuzzed the `DoGetPosition` function that returns a position to the `MobilityModel::GetPosition` function call. The fuzzing aspect is coded into the `MobilityModel::GetPosition` function as depicted in Algorithm 1 and is then subsequently used by the protocol calling the function. It is important to make the distinction between the actual position of the node versus the positional input received by the routing protocol. We are not adjusting the nodes' simulated positions. In the real world a node's actual position is not the same as the positional data it receives. Instead, geographic routing protocols use GNSS positional data to approximate their real position and aid in making routing decisions based on the three fundamental assumptions listed in Section 2.3.

The code behind the Fuzzing Algorithm was built around the the `mobility-model.cc(.h)` of the core mobility model library inside ns-3. By modifying the variance for each GNSS system and then recompiling the ns-3 code, we artificially introduce error for each protocol run against the three unique scenarios. In this way, regardless of which protocol we run, the position of the node in question is fuzzed with artificially introduced error without actually modifying the underlying structure of the protocols themselves.

Algorithm 1 `mobility-model.cc` Fuzzing Code

```
.AddAttribute ("PNTNormalDistribution", "Configurable PNT parameters.",
       // Defaults to zero mean, and std dev = 2, and bounded to +-2000 of the mean
       StringValue ("ns3::NormalRandomVariable[Mean=0.0|Variance=1.0|Bound=2000.0]"),
       MakePointerAccessor (&MobilityModel::m_PNT_NormDist),
       MakePointerChecker<NormalRandomVariable> ())

Vector
MobilityModel::GetPosition (void) const
{
       Vector FuzzPosition = DoGetPosition():
       FuzzPosition[0] += m_PNT_NormDist->GetValue();
       FuzzPosition[1] += m_PNT_NormDist->GetValue();
       return FuzzPosition;
}
```

The parameter `Mean` is set to 0 by default. `Bound` denotes the max range of the error we want to introduce assuming that any accuracy error greater than 2000 m would occur with negligible frequency given a normal distribution. Furthermore, setting the bound this high is an order of magnitude above the expected max error, so it does not artificially limit the distribution. Table 3.4 depicts the parameters for each GNSS system used in our simulations. Note that for a baseline for our thesis, we ran each of our scenarios with no error.

Table 3.4. Variance Parameters used during Simulation.

GNSS	Min Accuracy	Variance
No Error (Best Accuracy)	$0m$	N/A
QZSS	$1m$	0.707107
GPS	$2m$	1
GLONASS	$6m$	1.73205
Max Error	$200m$	10

Algorithm 2 is shown to illustrate the placement of the function call inside the `mobility-model.h` that is used by other files inside ns-3. Note that the variable used

in our fuzzing code is `m_PNT_NormDist` and can be modified to accept various GNSS accuracy measurements.

Algorithm 2 `mobility-model.h` Fuzzing Code

```
1: double Ptr<NormalRandomVariable> m_PNT_NormDist;
```

3.4 Protocol Implementation

The structure of ns-3 makes implementing protocols simple. The structure follows common naming conventions and is built around five distinct directories. Each protocol's folder contains a `/doc`, `/examples`, `/helper`, `/model`, and `/test` sub-directories. `/doc` contains documentation supporting reference material specific to each of the models. The `/examples` folder contains tutorials and helpful documentation to assist newcomers learn the functionality of the simulation suite. The `/helper` folder contains files that can be used in cross simulations. The `/model` folder contains all the standard protocols distributed with the ns-3 installation package and is where researchers can add their own prototypes for execution. Last, the `/test` folder contains installation validation scripts that test the integrity of the original install and subsequent protocol injections [15].

Within each protocol `/model` folder, there are four critical files for implementation and simulation. The `packet.cc` file implements aspects of the packet headers and serialization. The `queue.cc` folder manages the queue management scheme and calls the `packet.cc` to access header information. Two critical elements of this file are created: the `PacketQueue`, the protocol's buffer, and the `QueueEntry` inside the buffer itself. The `protocol.cc` implements the exchange of messages between participating nodes using both the `packet.cc` and the `protocol.cc` modules to grab the necessary header and queue management information to function properly. Last, the `tag.cc` implements specific modules related to the simulation meta-data [15]. Note that each of these files have a unique identifier that is part of a specific naming convention within ns-3. Protocol names should proceed the first dash of each file depicted (i.e., `gapr-packet.cc`).

In order to implement the four protocols we studied in this thesis, we had to make sure the folders and files associated with each were placed in the correct directories inside ns-3. We then ran a simple test to confirm that the default configurations were present before

making the modifications to the `MobilityModel::GetPosition` function call described in Section 3.3. This ensured that previous work was validated prior to injecting artificial error and running our own simulations.

The parameters used in the simulation stem from previous work on each of the four protocols of study and the mobility scenarios described in Section 3.2 of this Chapter. The configurable parameters are based on Table 2.4 of Chapter 2. GNSSs unique min accuracy measurements (best accuracy) are implemented in Algorithm 1 as depicted in Table 3.4.

ns-3 allows several mobility models that change the movement characteristics of the participating nodes. `ConstantPosition` forces a static position of nodes in the DTN. `RandomWaypoint` creates random waypoints at set intervals that each node moves to. A node will take the most direct path from its current position to the next, crossing other nodes along the way. `RandomDirection` is similar to `RandomWaypoint`, but instead of nodes moving to a new waypoint, they travel in a new random direction for a set period. `RandomWalk2d` and `RandomWalk3d` were of interest to us because they provided the most realistic model to simulate real-world node movement. The key difference between the two is the addition of a third dimension, altitude (z) that is a useful model if researchers are interested in observing the performance of aerial nodes. For the purpose of this thesis, we implement the `RandomWalk2d` model and scoped our research to a two-dimensional plane.

3.5 Observed Metrics

Metrics allow us to characterize how our simulated network (or a real network) performs. Although there are many ways researchers can characterize results, and in turn, interpret and visualize them, the set of metrics chosen depends on the type of research being conducted. Because DTNs operate in highly mobile and constrained environments, and the end goal is ultimately maximizing the amount of successful message delivery, many metrics would not provide tangible results that would make sense. Further, ns-3 data outputs are raw, and large data dumps that require additional post-scripting to parse and make sense of the data. Post-scripting is easily done through the use of a Python script and packages like `Pandas`, `NumPy`, and `Matplotlib`.

3.5.1 Message Delivery Ratio (MDR)

MDR is a performance metric used in ad-hoc networks. Specifically, it is a unitless calculation that describes how well a protocol delivers packets from source, through relays, and ultimately to the destination, regardless of the mobility model in use. It is calculated by dividing the total number of packets delivered by the total number of packets transmitted. This is depicted by Equation 3.1.

$$\text{MDR} = \frac{\text{\# Messages Delivered}}{\text{\# Messages Generated}} \tag{3.1}$$

3.5.2 Goodput

Goodput is similar to Throughput with one distinct difference, it defines the actual data received by removing redundant copies of the same message that are received through retransmissions or duplicates caused by message flooding strategies. It is defined in units of Bytes per second (B/s). This is depicted by Equation 3.2.

$$\text{Average Goodput} = \frac{\text{Average Bytes Received Per Node}}{\text{Average Delivery Time Per Message}} \tag{3.2}$$

3.5.3 Latency

Latency can be calculated in a multitude of ways and is dependent on the network being studied. It is the measure of determining the delay of messages being exchanged between a source and destination. Latency is particularly important in DTNs because applications rely on efficient delivery of generated messages to operate. Latency is calculated by using the following equation and has units of time (typically ms or s). This is depicted by Equation 3.3.

$$\text{Latency} = \text{Time Of Message Delivery} - \text{Time Of Message Transmission} \tag{3.3}$$

3.5.4 Hop Count

Hop Count is an important metric to observe because we can determine how well a protocol can make efficient routing decisions to lower the number of times a message must be forwarded. The fewer relays a message traverses through, theoretically, the faster the message makes it to the destination. A lower hop count may also correlated with a more energy efficient data exchange since each radio transmission consumes power. Hop Count is incremented as a message is forwarded by each node along its path.

3.6 Simulation Parameters

Based on what we have discussed, we decided that mobility scenario default parameters would be used against each of the geographic routing protocols of study. Only the `m_PNT_NormDist` variable would be adjusted to meet the simulation requirements needed for our research. Table 3.5 illustrates the default parameters used for the Helsinki mobility scenario. The key difference between the parameters used in previous simulations and our research was the *simulator seed*. We used four sets of mobility scenario maps each with their own unique seeds of 717, 718, 719, and 720. This was done for two reasons. First, we were only concerned about the impact of positional error on geographic routing protocols. Modifying other parameters would change the baseline simulation performance without contributing insight into the effects of positional error. Second, the time required to run the simulations using the parameters used in previous research would increase the simulation run times exponentially.

The default parameters used for the Omaha mobility scenario, as depicted in Table 3.6, utilize the same default parameters discussed as the Helsinki model for the same reasons. Although Omaha and Helsinki use several of the same parameters, the simulation implements a completely different set of nodes and bandwidth characteristics. *simulator seed*, *message rate*, and *message size* are the only parameters that are shared between both mobility scenarios.

The Bold Alligator scenario has significantly more parameters than Helsinki and Omaha. The *simulation seed* is the only parameter that is shared between all three. The *base radio bandwidth* was fixed to 24 Mbps, and a unique seed was set to each scenario map. The rest of the parameters were fixed. Table 3.7 depicts the default parameters used in our research.

The ability to change the variance of our error distribution code was necessary in order to account for GPS, GLONASS, and QZSS accuracies that were introduced into each geographic routing protocol. The default parameters used in four separate sets of simulations against each mobility scenario and protocol are the same as the accuracy (min) as depicted in Table 2.4 of Chapter 2. Note that we used the GPS II (series) GNSS constellation, over the GPS III (series) since it is not yet FOC. Because GLONASS has a range of accuracies, we used the mean of this range, 6 m, for our simulations.

Table 3.5. Default Simulation Parameters of the Helsinki Mobility Scenario.

Parameter	Values
Simulation Duration	12 hrs
Simulator Seed	717, 718, 719, 720
Warmup Time	1000 s
Timestamp Resolution	0.1 s
Beacon Interval	5 s
Number of Pedestrians	80
Number of Cars	40
Number of Trams	6
Base Radio Bandwidth (Mbps)	54
Tram Radio Bandwidth (Mbps)	526.5
Base Radio Transmit Range	10 m
Tram Radio Transmit Range	1000 m
Base Buffer Size (MB)	5
Tram Buffer Size (MB)	50
Message Rate	1/25-35 s
Message Size	0.5-1.0 MB
Message TTL	5 hrs
Hop Limit	50
Protocols	Centroid, GAPR, GAPR2, Vector

Table 3.6. Default Simulation Parameters of the Omaha Mobility Scenario.

Parameter	Values
Simulation Duration	12 hrs
Simulator Seed	717, 718, 719, 720
Warmup Time	1000 s
Timestamp Resolution	0.1 s
Beacon Interval	5 s
Number of Soldiers	44
Number of Ships	17
Radio Bandwidth	12 Mbps
Radio Transmit Range)	550 m
Solider Node Buffer Size (MB)	5
Ship Node Buffer Size (MB)	50
Message Rate	1/25-35 s
Message Size	0.5-1.0 MB
Message TTL	5 hrs
Protocols	Centroid, GAPR, GAPR2, Vector

Table 3.7. Default Simulation Parameters of the Bold Alligator Mobility Scenario.

Parameter	Values
Simulation Duration	24 hrs
Simulator Seed	717, 718
Warmup Time	1000 s
Timestamp Resolution	0.1 s
Beacon Interval	5 s
Number of Marines	70
Number of Humvees	20
Number of Drones	2
Number of Helicopters	8
Number of LCACs	2
Number of Ships	3
Base Radio Bandwidth	12 Mbps
Humvee Radio Bandwidth	54 Mbps
Drone Radio Bandwidth	6 Mbps
Ship Radio Bandwidth	54 Mbps
Base Radio Transmit Range	100 m
Humvee Radio Transmit Range	3000 m
Drone Radio Transmit Range	3000 m
Ship Radio Transmit Range	10000 m
Marine Node Buffer Size (MB)	5
Drone Node Buffer Size (MB)	5
Humvee Node Buffer Size (MB)	50
LCAC Node Buffer Size (MB)	50
Helo Node Buffer Size (MB)	50
Ship Node Buffer Size (MB)	500
Marine Message Size	250-500 KB
Humvee Message Size	0.5-1.0 MB
Ship Message Size	0.5-1.0 MB
Marine Message Rate	1/5-10 s
Humvee Message Rate	1/10-20 s
Ship Message Rate	1/25-35 s
Message TTL	5 hrs
Protocols	Centroid, GAPR, GAPR2, Vector

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Data Collection and Analysis

In this chapter, we discuss our simulation results. We begin with our observations of the simulations as they were running, followed by how we collected the raw data. We then provide an analysis of each metric against each mobility scenario and protocol, as outlined in Section 3.5 of Chapter 3. We end with a comparative analysis of each protocol's observed performance against each scenario. Because of the size of the output files, it was necessary to run post scripts to parse and aggregate the data into statistical measure, which was then presented using line and bar graphs. More detailed raw data may be found in Appendices A, B, and C.

4.1 Observations

Because ns-3 is a discrete-event simulator, simulations do not run in real-time. A single second in real-time does not necessarily correlate to a single second in the simulation. The time it takes for a specific protocol to execute the simulation to completion is entirely dependent on the scenario that is running and how each protocol computes its unique routing primitives. To run our simulations, we utilized a server farm that allowed for sixty unique simulation sets to run in parallel. On average, the Helsinki and Omaha scenarios took anywhere from three to four days to complete across the four protocols. Bold Alligator averaged about nine days of run time to complete. The protocols themselves also contributed to simulation run times. Centroid and Vector almost always ran to completion in about half the time of GAPR2, while GAPR took the longest by taking roughly three times as long to complete. This was observed to be true across each of the three scenarios.

The second observation was based on the introduced GNSS error. As expected, the simulations with no error introduced ran to completion first, whereas the simulations with a 200 m introduced error took, on average, about three to four times longer to complete. Because the Bold Alligator simulation run time was double that of Helsinki and Omaha, and there were six unique node types totaling 105 DTN nodes participating in the scenario, it is likely that the complexity and computational burden of computing routing decisions drastically

slowed the simulation time. Both Mauldin and Brown observed significantly longer run times, making the assumption that the Bold Alligator quickly reaches a point between 600 s and 800 s in real-time after the warm-up period where the scenario then drastically slows down [71], [72]. Computing the error adds many calls to the random number generator, which increases the computational cost of the simulation.

4.2 Data Collection

Each ns-3 simulation creates a `.tr` trace file once the scenario warm-up period has completed. As the simulation progresses and nodes exchanges messages, new data is appended to the trace file. These files quickly grow to between 1-2 GB in size once the simulation is complete. Because of the large file size, we ran a post-processing script that parsed each individual protocol trace file into manageable data sets that we used to populate our graphs. In order to tighten our confidence intervals, we ran four sets of Helsinki and Omaha simulations against each of the four protocols of study for a total of 160 simulations. Since Bold Alligator showed less performance variance between runs (due in part to the fact that the scenario generates 1200% more messages than the others), we only ran two sets of simulations against each protocol totaling 32 simulations. This left us with a total of 192 simulations that ran to completion.

Since we conducted four simulations (two for Bold Alligator) for each protocol, each with their own mobility scenario map, we had four (two for Bold Alligator) unique trace files for each GNSS system. We then took the average of each set and used it for our plots. Note that the x-axes on our graphs have been adjusted so that trends in the data can be easily viewed.

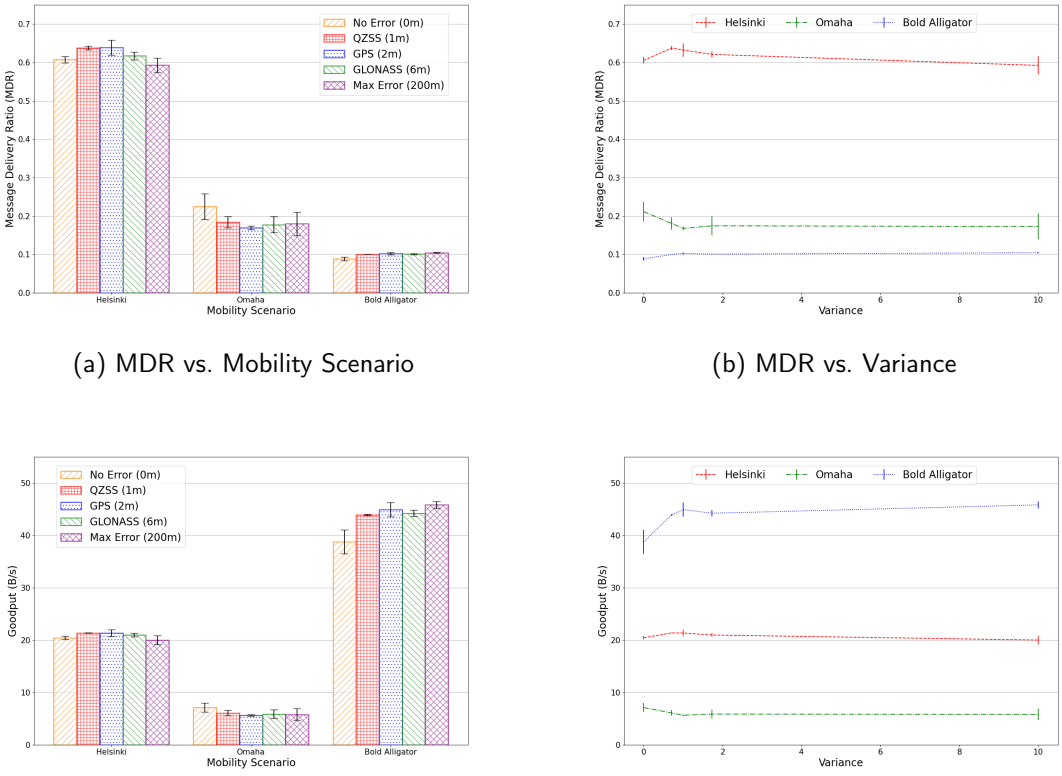
4.3 Vector Analysis

As previously stated, Vector almost always ran to completion first. This is likely due to its message replication primitive. Vector makes its routing decisions based on a neighboring node's direction and velocity. If a neighboring node is traveling in the same or opposite direction, Vector tends to exchange few or no messages with that particular neighbor. If a neighboring node is traveling in an orthogonal direction, then Vector exchanges more messages in an attempt to spread copies of the messages it is carrying to geographic areas it has not been to and will not visit. Additionally, when nodes exchange their direction

and velocity, Vector exchanges more messages with the nodes that are moving the fastest. Combining orthogonality, direction, and velocity of neighboring nodes provides a weighted average that is then used to make the routing decision [10].

4.3.1 Message Delivery Ratio (MDR) and Goodput

Figure 4.1 depicts Vector's MDR and goodput performance across each scenario. We present MDR and goodput together because successful message deliveries account for goodput's unique message calculation.



(a) MDR vs. Mobility Scenario

(b) MDR vs. Variance

(c) Goodput vs. Mobility Scenario

(d) Goodput vs. Variance

Figure 4.1. Vector: MDR/Goodput Performance (with 95% confidence intervals shown)

As a result, both MDR and Goodput should have similar trends, as we observed in both

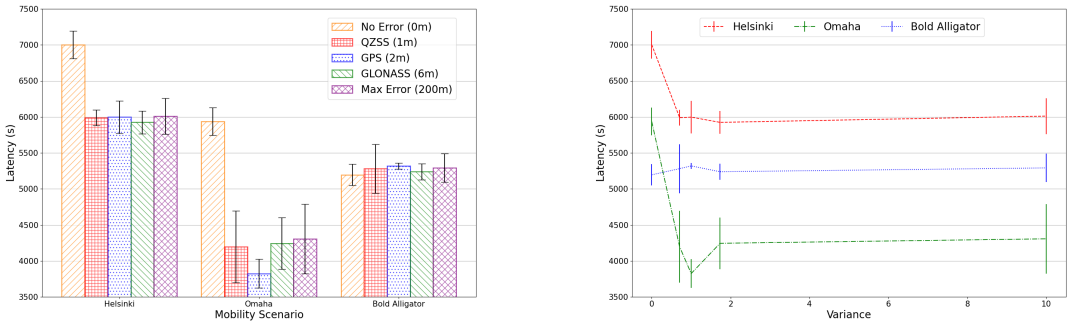
the Helsinki and Omaha mobility scenarios. The results show that Vector performs the best when run against the Helsinki scenario. Vector's MDR is doubled when compared to Bold Alligator. The results are consistent with previous research [71], [72]; additionally the results indicate that introduced GNSS positional error has little affect on Vector's MDR performance. The Helsinki scenario is diverse, with nodes traveling in many different directions. Because Vector capitalizes on node orthogonality, and because of Helsinki's diversified node movements, it is likely that more messages are exchanged between nodes, increasing the chance of successful delivery. Both Omaha and Bold Alligator mobility scenarios have nodes that revisit paths consistently and would suggest that Vector exchanges messages less frequently resulting in a lower MDR. It is significant to note that when QZSS variance is applied to the Helsinki scenario, Vector performs slightly better than with no error. This suggests that Vector's protocol primitive is over-fitting node positions to its advantage. The Omaha scenario MDR performance follows how we had initially expected that introduced error would affect protocol primitives with decreasing MDR performance as more error is introduced. If we consider the error bars, it remains consistent across variances. Bold Alligator remains consistent regardless of which variance was introduced.

Goodput is closely tied to MDR because successful message delivery directly accounts for goodput's calculation. This is because goodput is calculated on unique message deliveries. It is the average of each unique message size divided by the amount of time each message was in transit. As with MDR, there appear to be no observable effects of introduced GNSS error in either the Helsinki or Omaha mobility scenarios. Bold Alligator shows a slightly upward trend as the variance is increased with the best goodput with the 200 m error applied. Additionally, we do not see a similar MDR and goodput trends in Bold Alligator as we do in Omaha and Helsinki. This is because Bold Alligator transmits 1200% more messages than the other two scenarios. Similar to MDR, the greater number of generated messages plus the number of nodes participating in Bold Alligator create more opportunities for Vector to exchange its messages in accordance with its routing primitive.

4.3.2 Latency

Figure 4.2 depicts Vector's Latency performance. Here we observed an notable phenomenon with the highest latency occurring with no GNSS error applied in both the Helsinki and Omaha mobility scenarios. This is an artifact of Vector's orthogonal mechanism in its

routing primitive and the total number of messages transmitted in both scenarios. When no error is introduced, messages are forwarded less frequently causing them to spend more time in buffers before arriving at their destination. When error is introduced, the total number of message transmissions almost triples across all error models as seen in our data tables located in Appendix A and B. As more messages are exchanged, buffers are emptied, and latency is decreased. Bold Alligator sees this same effect, but due to the total number of messages generated and the larger number of participating nodes in the scenario, latency is absorbed across the network as Vector exchanges its messages. Furthermore, we observe that latency varies across each scenario, with ranges anywhere from 100 s to 1000 s. Disregarding the differences in latencies among each unique simulation, we generally see consistent performance when considering possible results within the error bars. Appendix A and Appendix B provide raw data from both the Helsinki and Omaha scenarios if the reader is interested in how we accounted for the the latency spike.



(a) Latency vs. Mobility Scenario

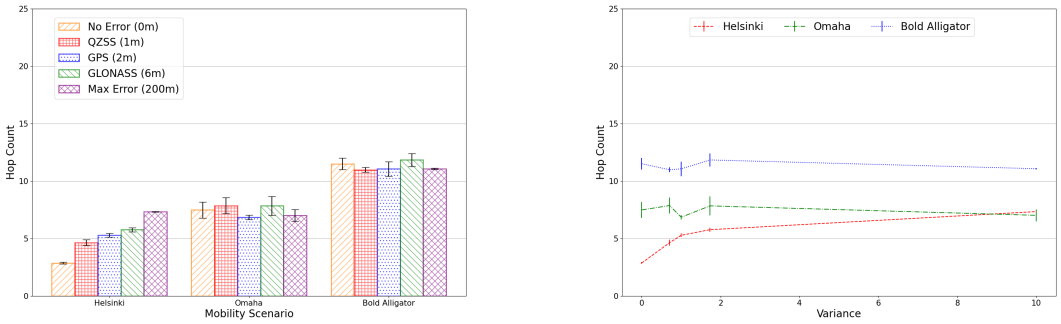
(b) Latency vs. Variance

Figure 4.2. Vector: Latency Performance (with 95% confidence intervals shown)

4.3.3 Hop Count

Figure 4.3 depicts the number of times a message hits a relay node prior to reaching its destination. GNSS positional error seems to only affect Vector's performance when run against the Helsinki scenario. We see an increase in hop count at each iteration of error introduced with the greatest observed effect occurring at the 200 m error. This increase is

likely an artifact of Vector's orthogonal routing primitive thresholds being met for frequently and exchanging messages with nodes based on inaccurate positional data. The resultant effect is the message's propagation through more nodes than usual.



(a) Hop Count vs. Mobility Scenario

(b) Hop Count vs. Variance

Figure 4.3. Vector: Hop Count Performance (with 95% confidence intervals shown)

It is a notable observation that both Omaha and Bold Alligator hop counts are somewhat immune to GNSS positional error and generally consistent across each simulation set. This is likely a result of the mobility scenarios themselves. Participating nodes in both mobility scenarios follow more direct or consistent paths. Helsinki's urban scenario provides more node movement entropy (i.e., pedestrians, vehicles, and trams). As a result, Vector is likely more affected by positional error resulting in an upward trend in Hop Count.

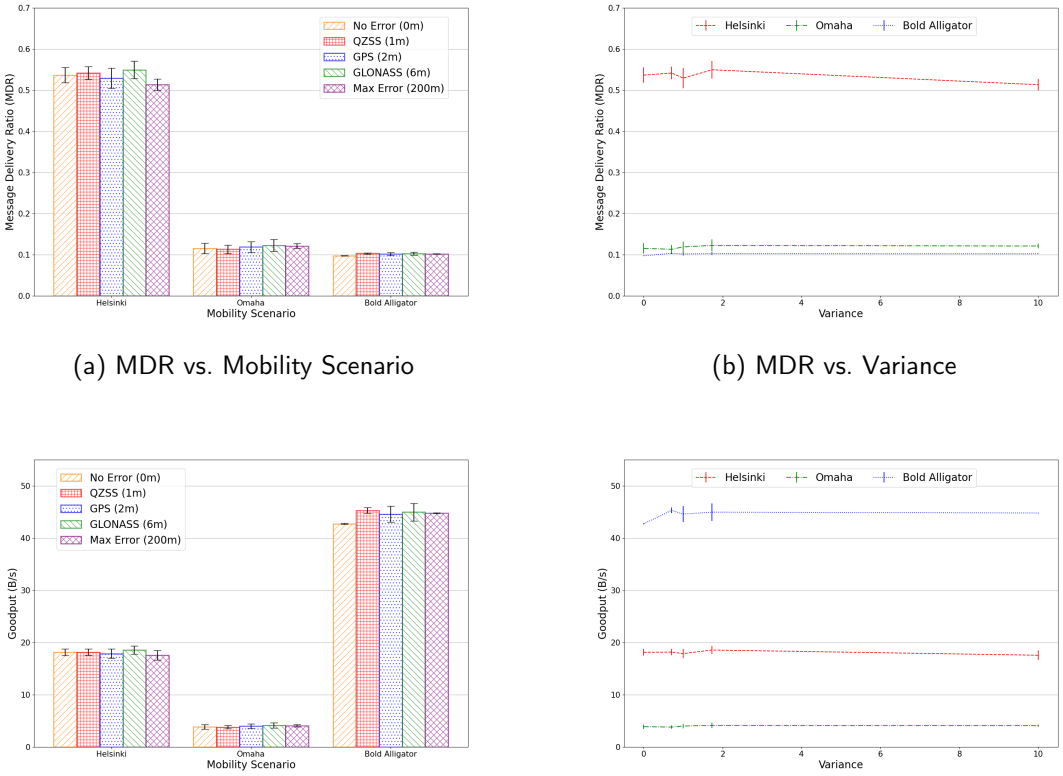
4.4 Centroid Analysis

As stated in Section 4.1, Centroid typically ran to completion within a few hours after Vector. Centroid's primitive places emphasis on the Cartesian distance between two nodes. At each node encounter, Centroid exchanges its current centroid (a balanced central point based on the history of the node's locations), a message list with all the messages that have been received by the destination, a list of all messages currently in its buffer, and a list of all its current neighbors. To reduce redundant messages and network saturation, Centroid's acknowledgment list notifies neighboring nodes of messages that have been delivered.

Neighboring nodes then remove messages from their buffers. Each node then calculates the Centroid's distance and message limit. The result is a linear relationship between a Centroid distance and messages exchanged that averages out error over time. More messages are then exchanged with more distant centroids, reducing the number of message replications across the network [11].

4.4.1 Message Delivery Ratio (MDR) and Goodput

Figure 4.4 depicts Centroid's observed MDR and goodput performance across each mobility scenario. As expected, Centroid's MDR performance appears to be immune to any introduced GNSS error across each mobility scenario.



(a) MDR vs. Mobility Scenario

(b) MDR vs. Variance

(c) Goodput vs. Mobility Scenario

(d) Goodput vs. Variance

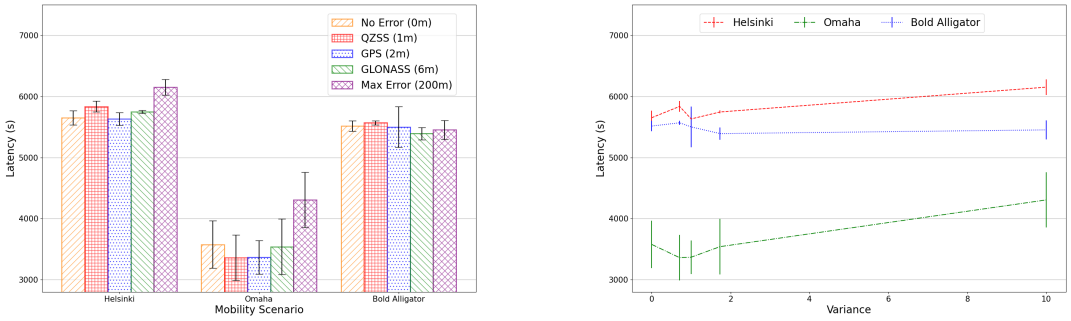
Figure 4.4. Centroid: MDR/Goodput Performance (with 95% confidence intervals shown)

This is true because Centroid's primitive averages out any error over time by calculating deltas between a node's old and new centroids at each encounter; therefore, we see a consistent MDR. This is consistent with Rohrer's findings [11] and others who have studied Centroid in their research [71], [72]. This is a desirable characteristic in DTNs, especially when participating nodes lack LOS or have GNSS signal degradation while in urban or natural canyons. As with MDR, Centroid's goodput performance with GNSS variance introduced appears to be immune to any positional errors across each mobility scenario. The differences between each mobility scenario can be explained based on the scenarios. Because Bold Alligator generates significantly more messages, its goodput is observed to be nearly double that of Helsinki's goodput and four times that of Omaha. It is essential to note that the goodput performance is observed to be better for Helsinki and Omaha if we consider the number of messages generated by each scenario. On a per-message ratio, Helsinki goodput performance is roughly 475% better than Bold Alligator when considering total number of messages generated by each scenario. Omaha goodput performance is roughly 27% better. This fits the trend relationship between MDR and goodput.

4.4.2 Latency

Figure 4.5 depicts Centroid's observed latency performance. As with Vector, we observe Centroid's performance between 5000 s and 6000 s for both Helsinki and Bold Alligator mobility scenarios. Helsinki remains consistent with a spike in latency when max error is introduced. Because the variance difference between GLONASS and max error is 194 m, Centroid is exchanging its control message (i.e., acknowledgments list, centroid lists, etc.) more frequently causing the spike seen at max error suggesting an upward trend in latency as more error is introduced. Centroid's performance in the Omaha scenario appears to be the best with GPS error introduced. It is challenging to correlate why Centroid latency appears to be the best in this case. It may be an artifact of the mobility scenario itself and should be studied further. It may be caused by over fitting node positions with no error so when error is introduced, latency decreases. Similar to Helsinki, Centroid's latency appears to spike with max error applied in the Omaha mobility scenario likely for the same reasons as Helsinki. Centroid's goodput performance in the Bold Alligator mobility scenario remains consistent with a notable observation. The error bars on GPS show a fluctuation that suggests that Centroid's routing primitive hits a convergence point that can affect its performance in either direction. Another observation is the absence of the max error latency spike seen in both

Helsinki and Omaha. It suggests that because Bold Alligator has more participating nodes and has a 1200% increase in message generation, that Centroid is able to average out error more often. This works to Centroid's advantage because there are more opportunities for successful message deliveries when each subsequent message acknowledgment allows for the removal of a message from a node's buffer.



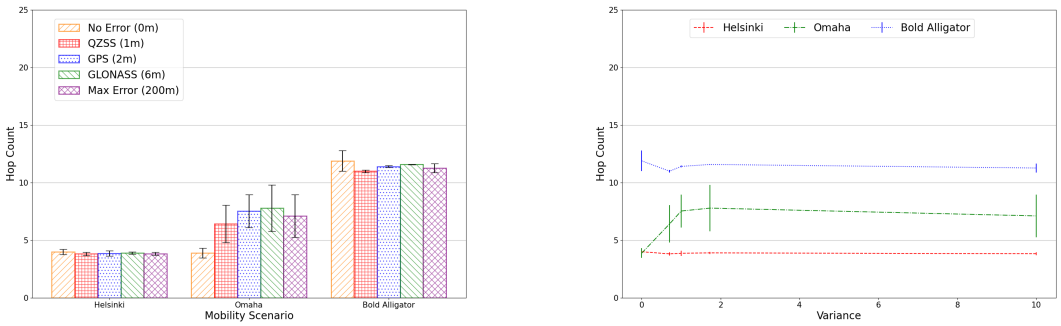
(a) Latency vs. Mobility Scenario (b) Latency vs. Variance

Figure 4.5. Centroid: Latency Performance (with 95% confidence intervals shown)

4.4.3 Hop Count

Figure 4.6 depicts message hop counts for Centroid. The Helsinki mobility scenario shows a consistent message hop count suggesting that Centroid is immune to introduced error in the Helsinki scenario. This is also seen in Bold Alligator and is likely an artifact of the scenarios. Helsinki's high node entropy provides the potential for more node encounters leading to Centroid exchanging routing primitive lists more frequently. Because of this increase in encounters, Centroid is better able to calculate the distance delta between centroids and forwards more messages to centroids with a greater delta. This pushes messages closer to the destination. Bold Alligator's increased node count and messages generated likely adds to Centroid's increased neighboring node encounters, providing more opportunities to update its centroid lists. The effect is a consistent hop count. Omaha appears to have an upward hop count trend. This is likely an artifact of the mobility scenario itself because of the low participating node count, the number of messages generated, and the consistent node

movement of the troop along the shore and ships at sea.



(a) Hop Count vs. Mobility Scenario

(b) Hop Count vs. Variance

Figure 4.6. Centroid: Hop Count Performance (with 95% confidence intervals shown)

As nodes build a history of movement and Centroid builds its centroid lists, it remains consistent with no error introduced. When error is introduced at each node encounter, Centroid likely exchanges messages with nodes assuming that their positions are accurate. As a result, each message has to hit more relays en-route to the destination to compensate. This was consistent, regardless of which error model was introduced.

4.5 GAPR Analysis

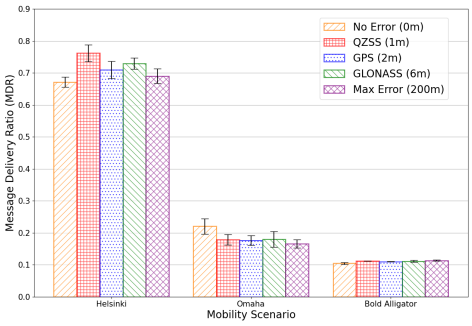
GAPR takes a different approach to routing messages in a DTN. It implements an acknowledgment mechanism that broadcasts out to the network that a message has been delivered and copies are therefore deleted from other node's buffers freeing up buffer space in the process. Second, GAPR estimates a delivery probability for each message in its buffer based on a history of node encounters. GAPR then forwards messages with the highest delivery probabilities first, then steps-down its message list to messages with lower probabilities. If GAPR's buffer is full, and a new message with a higher delivery probability is received, then the primitive drops the message with the lowest delivery probability to make room for the new message. Last, GAPR implements a two-hop optimization primitive that checks whether a neighboring node is in contact with the message's destination. If the node is in

contact, then the message is given priority is first regardless of previously estimated delivery probability [14].

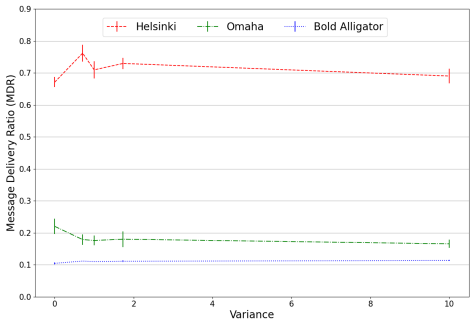
4.5.1 Message Delivery Ratio (MDR) and Goodput

Figure 4.7 depicts GAPR's observed performance across each mobility scenario. Of note, Helsinki appears to have the most significant differences against each variance introduced. We observed that GAPR's MDR increases when 1 m of error is introduced then slightly dips, remaining consistent from then onward (within the error bars). This suggests that Helsinki's increased node entropy has a positive effect on GAPR's MDR performance. This is likely because the delivery probabilities are artificially adjusted to increase a message's successful delivery. Furthermore, GAPR's buffer management primitive drops the messages with the lowest delivery probabilities. This increases performance because as the delivery probabilities are calculated, more churn in the buffer occurs with the addition of new messages and the removal of old ones. GAPR's performance in Bold Alligator appears to remain consistent across each variance. This is likely an artifact of the scenario based on the number of nodes and the consistency of node paths. GAPR is able to calculate delivery probabilities more accurately because the history of a node's location remains consistent over the 24 hr simulation period. Omaha illustrates how we would expect introduced error would affect a protocol's performance. The slightly downward trend in MDR suggests that Omaha's node entropy and node count affects GAPR's routing primitive. If we consider the error bars, we notice that MDR remains somewhat consistent once error is applied to the mobility scenario. Across each mobility scenario, it appears that GAPR's goodput remains consistent regardless of which variance is introduced. For Helsinki, there is a slight increase in performance from no error to QZSS introduced error, and then performance remains consistent out to the 200 m error. GAPR's performance in the Omaha mobility scenario suggests that GAPR is immune to any error. This could be an artifact of the mobility scenario. If we correlate GAPR's MDR performance to its goodput performance, we notice that when no error is introduced, MDR is slightly increased, yet goodput remains consistent across each variance. Because Omaha's node movement is consistent, GAPR is able to more accurately calculate the delivery probabilities of messages increasing its MDR. Since goodput is based on an average of unique messages over an average of each unique message delivery time, each individual flow's goodput metric is averaged together, correlating to the leveled trend we observe for Omaha. Bold Alligator trends upwards, which is likely

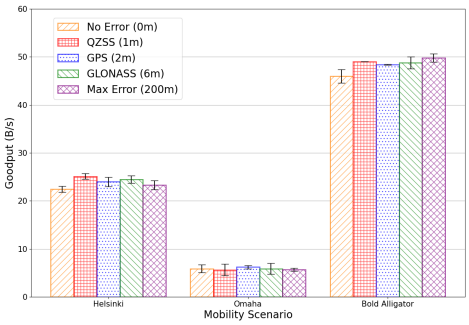
caused by the total number of messages generated and GAPR's ability to calculate delivery probabilities.



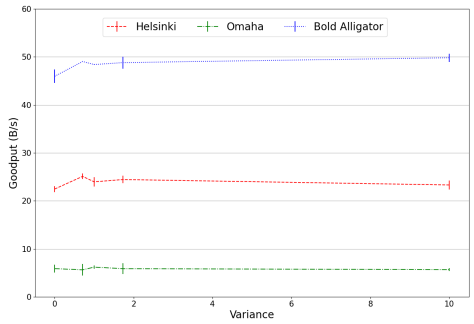
(a) MDR vs. Mobility Scenario



(b) MDR vs. Variance



(c) Goodput vs. Mobility Scenario



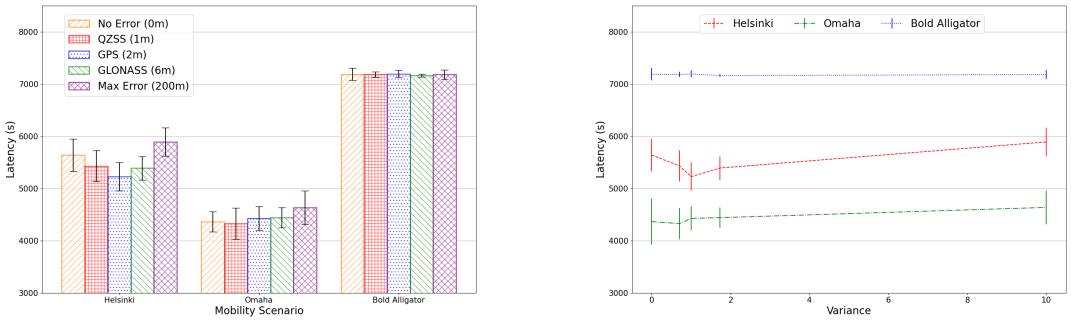
(d) Goodput vs. Variance

Figure 4.7. GAPR: MDR/Goodput Performance (with 95% confidence intervals shown)

As more error is introduced, node encounters become more diverse, lowering a message's delivery probability. This leads to the message eventually being dropped in place of a new message with a higher delivery probability. Although a minor increase, this upward trend is an important observation because it shows that introduced error has a positive affect on GAPR's routing primitive.

4.5.2 Latency

As with Vector and Centroid latency observations, GAPR's performance provides some notable results. Figure 4.8 depicts GAPR's latency performance across each mobility scenario. Both Omaha and Bold Alligator's results show a consistent trend, with Omaha only slightly increasing as variance is introduced.



(a) Latency vs. Mobility Scenario (b) Latency vs. Variance

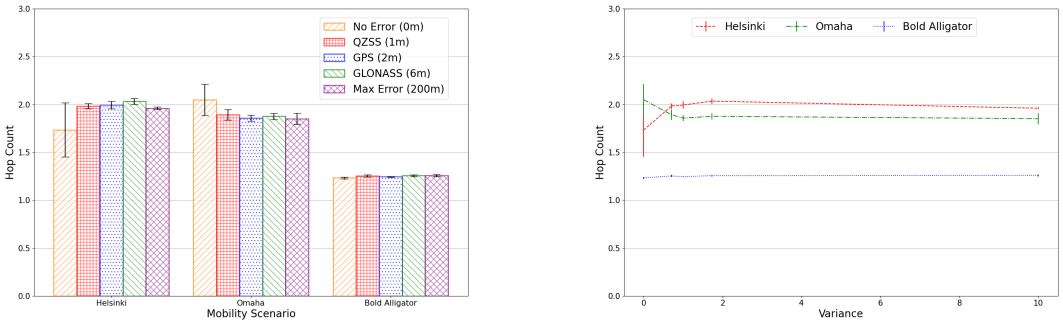
Figure 4.8. GAPR: Latency Performance (with 95% confidence intervals shown)

If we consider data points falling within the error bars, we observe GAPR's latency performance, when run against the Omaha mobility scenario, remains consistent. The Helsinki results are significant because they show that when GPS error is introduced, GAPR performs the best. The results suggest that 2 m of error creates a convergence point that GAPR's primitives are able to use to calculate better delivery probabilities. Considering node entropy, GAPR is likely able to utilize node encounter history to its advantage at this convergence point. If we simply consider the data points within the error bars, this drop in latency with GPS error could be a cause of a smaller sample size. Increasing the number of runs may produce a better average across each variance.

4.5.3 Hop Count

Figure 4.9 depicts the hop count performance of GAPR. Here we observe two significant data points. In both Helsinki and Omaha, no error hop counts vary from the rest of the variance runs. The difference we observe could be a result of the mobility scenario and

not GAPR's routing primitive, especially if we consider the potential of data points falling within the error bars. In both scenarios, more simulations could potentially shift the hop count in-line with the rest of the error models.



(a) Hop Count vs. Mobility Scenario

(b) Hop Count vs. Variance

Figure 4.9. GAPR: Hop Count Performance (with 95% confidence intervals shown)

For Helsinki specifically, we would expect a lower message hop count if no error was introduced because the delivery probabilities would be more accurate. Furthermore, the two-hop optimization likely adds to lower the hop count. Conversely, Omaha's increase in hop count with no error introduced is notable because is counters our conclusions with respect to GAPR's performance in Helsinki. The lower hop counts observed with introduced error suggests that the difference is an artifact of the mobility scenario and not the variance. Omaha's node movements are generally consistent, and when error is introduced, GAPR calculates its delivery probabilities while dropping messages in the process, reducing the total number of individual hops by each unique message. The Bold Alligator results show no noticeable effect on GAPR's performance across all variances. Considering that the distribution of hop counts is so small, we surmise that GAPR is immune to introduced error across each scenario and variance.

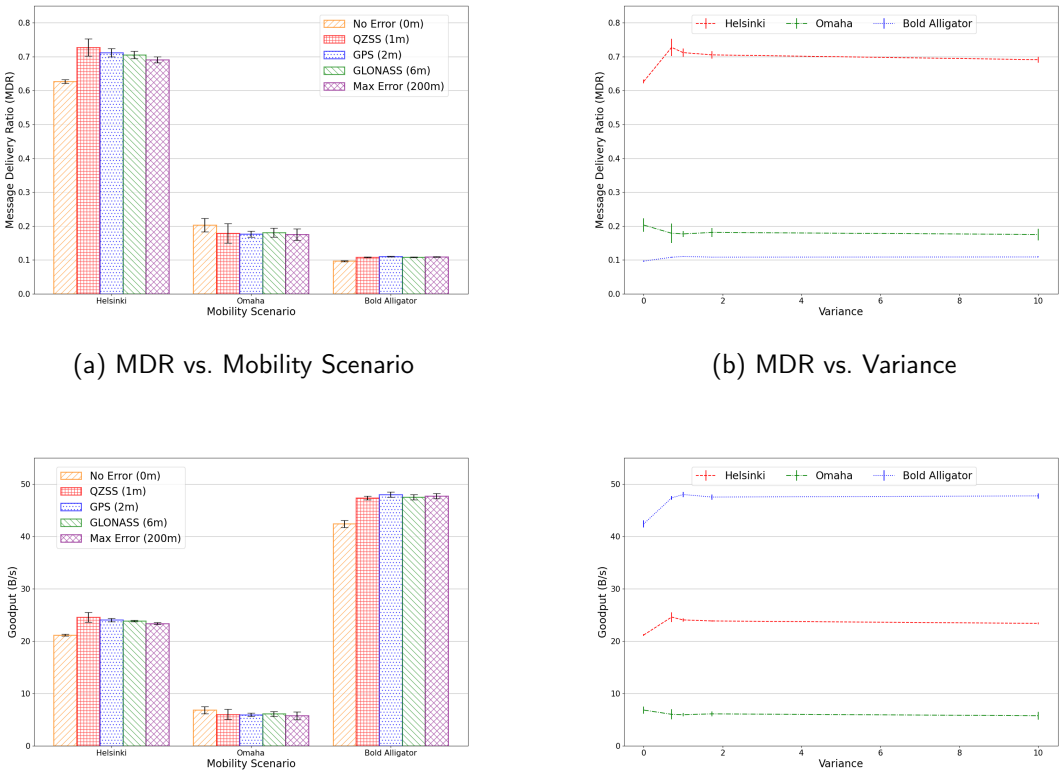
4.6 GAPR2 Analysis

GAPR2 capitalizes on Vector's routing primitive of orthogonality, direction, and velocity, in addition to GAPR's delivery probabilities and buffer management mechanisms. GAPR2

is able to exchange only the messages with the highest probability of successful delivery to nodes traveling the fastest and orthogonal to the source node's direction. This effect creates a wider message replication spread across the DTN, significantly increasing the chance of message delivery [12].

4.6.1 Message Delivery Ratio (MDR) and Goodput

Figure 4.10 depicts GAPR2's observed MDR and goodput performance against each mobility scenario. Both Omaha and Bold Alligator MDRs appear to remain consistent across each of the variances if we consider possible data points within the error bars.



(a) MDR vs. Mobility Scenario

(b) MDR vs. Variance

(c) Goodput vs. Mobility Scenario

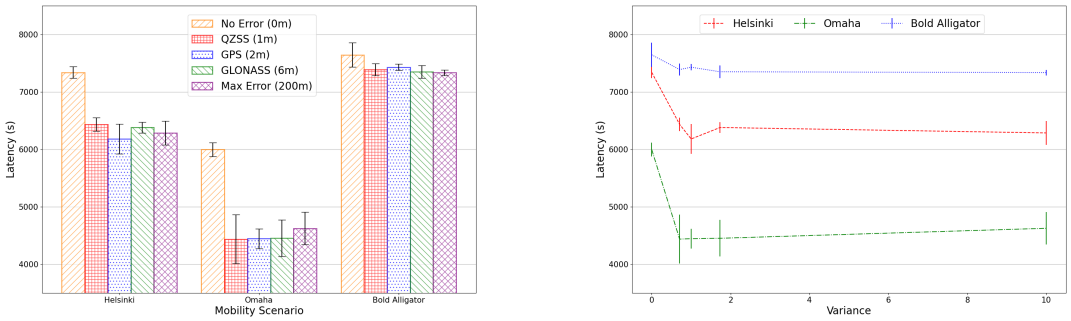
(d) Goodput vs. Variance

Figure 4.10. GAPR2: MDR/Goodput Performance (with 95% confidence intervals shown)

Similar to GAPR, GAPR2's performance in the Helsinki mobility scenario is likely an artifact of the scenario. Participating nodes in Helsinki have considerably more entropy forcing GAPR2's delivery probabilities to update more frequently at each node encounter. Since GAPR2 implements Vector's primitive of increased message exchange to orthogonal nodes traveling the fastest, the introduction of error likely forces messages with lower delivery probabilities to drop and increases the exchange of messages with higher probabilities to neighboring nodes. As messages are delivered, the acknowledgment broadcast reduces the burden on a node's buffer by allowing it to drop delivered messages in their individual buffers. Both these characteristics account for the increased MDR with each variance. For Omaha and Bold Alligator, we notice a consistent MDR trend with a slightly downward trend in the Omaha mobility scenario. The scenario's node paths likely cause this, and therefore GAPR2 is able to accurately calculate individual message probabilities consistently across each variance. Goodput is observed to be nearly identical to the performance of GAPR. This is likely because of the shared delivery probability primitive and two-hop optimization. One noticeable difference is the increased goodput performance within the Helsinki and Bold Alligator scenarios, from no error to QZSS error. This large increase is likely caused by the inclusion of Vector's orthogonal and velocity routing decision primitive. As individual messages are delivered based on their delivery probability to a node orthogonal to the source node, the message is dispersed more broadly across the DTN, increasing its chances of successful delivery.

4.6.2 Latency

Figure 4.11 depicts latency performance for GAPR2 across each mobility scenario. We once again notice a significant phenomenon when no error is introduced in both the Helsinki and Omaha mobility scenarios. This latency spike is caused by the inclusion of Vector's routing primitive. When no error is introduced, GAPR2 transmits nearly half the total number of messages as it does with error introduced. This causes messages to remain in a node's buffer which directly correlates to an increase in latency. As error is introduced, messages are transmitted freeing a node's buffer decreasing latency. This is observed in both Helsinki and Omaha mobility scenarios. Appendix A and B provide raw data for each scenario that we based our conclusions on with respect to the latency spike. Bold Alligator sees this phenomenon only slightly. The smaller increase in latency, when no error is introduced, suggests that GAPR2 acts much in the same way as it does in Helsinki and Omaha.



(a) Latency vs. Mobility Scenario

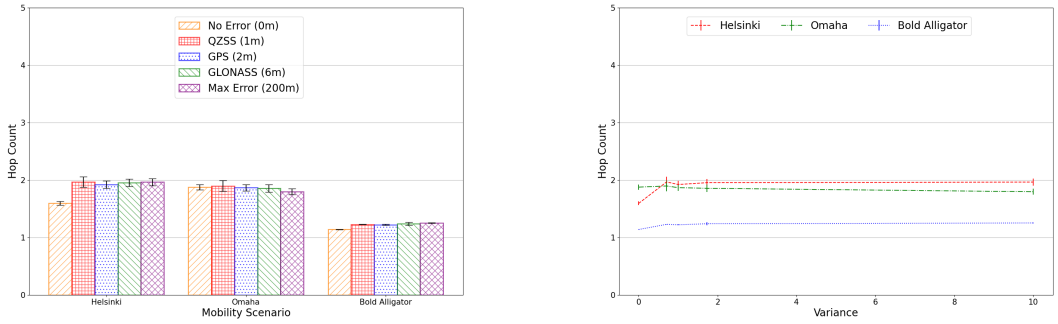
(b) Latency vs. Variance

Figure 4.11. GAPR2: Latency Performance (with 95% confidence intervals shown)

Even though the delivery probabilities are calculated on a per-message basis inside a node's buffer, the Vector mechanism likely prevents messages from being exchanged unless a neighboring node is orthogonal to a source node's direction of travel. More messages in buffers correlate to increased network latency. Because there are more nodes in Bold Alligator and the nodes consistently revisit the same paths, the latency spike, when no error is introduced, is not as significant.

4.6.3 Hop Count

Figure 4.12 depicts GAPR2's hop count performance across each mobility scenario. GAPR2's performance remains generally consistent across each scenario and variance. Of note is Helsinki, with no error introduced. It shows a small dip in hop count, suggesting that GAPR2 is able to accurately calculate in delivery probabilities and exchange messages with orthogonal neighboring nodes more efficiently. It suggests that GAPR2's two-hop optimization is able to send messages more efficiently to a destination than it does when error is introduced. GAPR2 achieves this by clearing its node's probability lists if there are enough sudden node movements at each encounter. This primitive ensures delivery probabilities are as accurate as possible before exchanging any messages. Overall, we observed relative immunity with respect to hop count when variance is introduced.



(a) Hop Count vs. Mobility Scenario

(b) Hop Count vs. Variance

Figure 4.12. GAPR2: Hop Count Performance (with 95% confidence intervals shown)

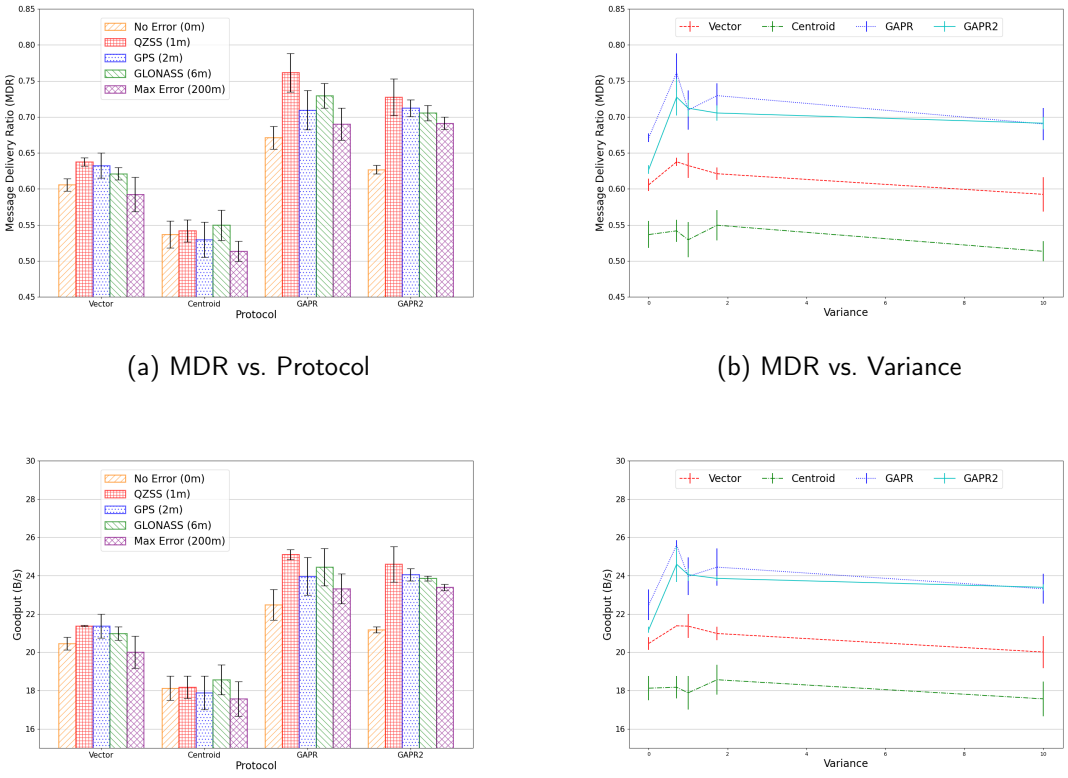
4.7 Comparative Analysis

In this section, we provide a side-by-side comparison of each protocol against each mobility scenario and metric. In this way, we are able to provide a comparative analysis of each protocol's performance when GNSS error is introduced. It is important to understand how a protocol's performance varies. The effect of GNSS PNT error on individual protocols was discussed in previous sections, so it will not be repeated here. Instead, we will provide our interpretations as to why a particular protocol appears to perform better in a specific mobility scenario.

4.7.1 Helsinki MDR and Goodput Comparison

Figure 4.13 depicts each protocol's MDR and goodput performance while running simulations using the Helsinki mobility scenario. Based on our observations, GAPR appears to perform the best but is not completely immune to GNSS error effects. Each protocol's MDRs are observed to perform better when QZSS error is introduced and then has a slight downward trend out to 200 m error suggesting that error does have an effect. It is likely that each of the protocols are over fitting error to account for ns-3's exact position when the `DoGetPosition` function is called from the `mobility-model.cc` module. By artificially introducing error, some protocols actually perform better in two scenarios, which is the

opposite of what we initially expected. As expected, goodput follows a very similar trend of MDR with both GAPR and GAPR2 performing the best across all variance models. As with MDR, we see a positive effect on goodput when QZSS error is introduced followed by a slightly downward trend out to 200 m. This is expected as goodput is directly related to MDR and should show a similar trend.



(a) MDR vs. Protocol

(b) MDR vs. Variance

(c) Goodput vs. Protocol

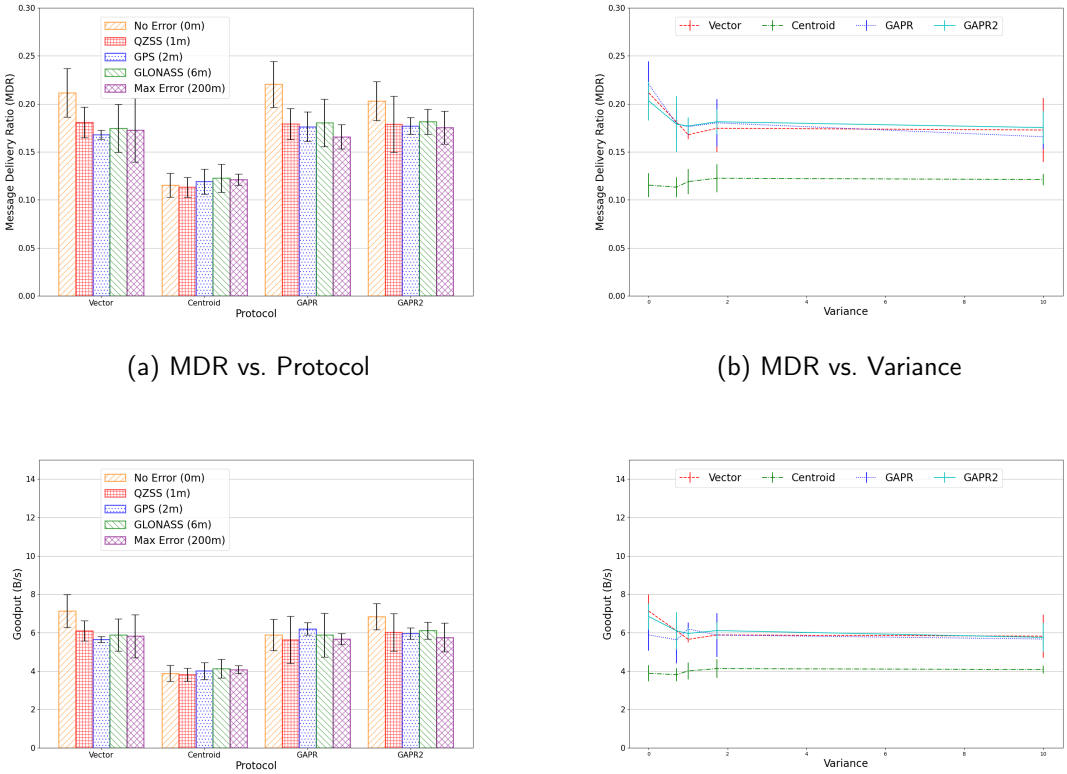
(d) Goodput vs. Variance

Figure 4.13. Helsinki MDR/Goodput: Protocol Performance Comparison (with 95% confidence intervals shown)

4.7.2 Omaha MDR and Goodput Comparison

Figure 4.14 depicts the performance of each protocol while running simulations using the Omaha mobility scenario. When no error is introduced, Vector, GAPR, and GAPR2 have the highest MDR across their simulation sets. As error is introduced, MDR begins to drop;

however, it remains within the error bars. The fact that the results between these three protocols seem close to each other suggests that the Omaha mobility scenario does not provide enough opportunities to send messages between nodes.



(a) MDR vs. Protocol

(b) MDR vs. Variance

(c) Goodput vs. Protocol

(d) Goodput vs. Variance

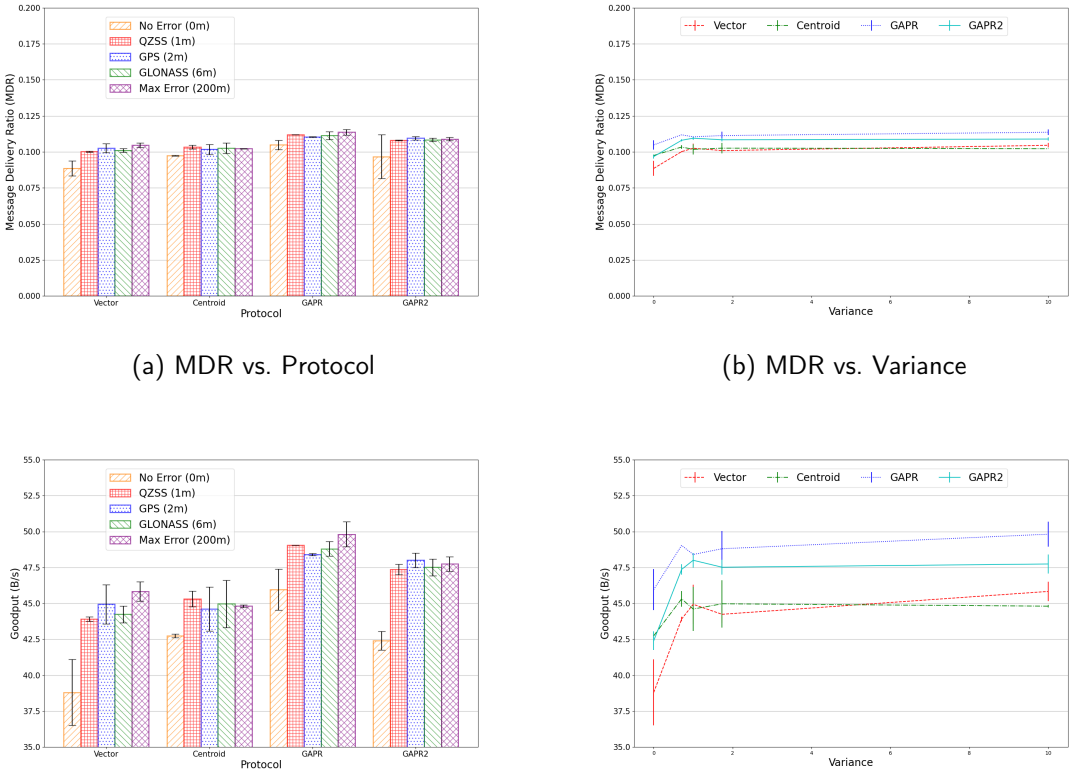
Figure 4.14. Omaha MDR/Goodput: Protocol Performance Comparison (with 95% confidence intervals shown)

Additionally, the nodes themselves likely play a critical part in how each protocol performs. Ships at sea remain static, and the amphibious troop transports parallel the shoreline. Most node entropy comes from the disembarked soldiers on the beach, but even they follow a predictable path. Combined with these aspects, the total number of messages generated and the participating node count likely contributes to the decrease in a protocol's ability to exchange a message. Centroid remains consistent throughout each simulation likely because it averages out error over time. If we focus on the trend-lines themselves, we see a steady

trend from QZSS out to 200 m error. Based on this, we surmise that the Omaha scenario itself has more of an effect on each protocol's MDR and goodput performance than an introduced variance. Note that even with Omaha's limitations in this respect, goodput's downward trend follows a similar trend as MDR as goodput is calculated based on unique message deliveries.

4.7.3 Bold Alligator MDR and Goodput Comparison

Figure 4.15 depicts MDR and goodput performance for each protocol while running the Bold Alligator mobility scenario. MDR appears to remain consistent across each protocol and variance with the exception of Vector and GAPR2 when no error is introduced.



(a) MDR vs. Protocol

(b) MDR vs. Variance

(c) Goodput vs. Protocol

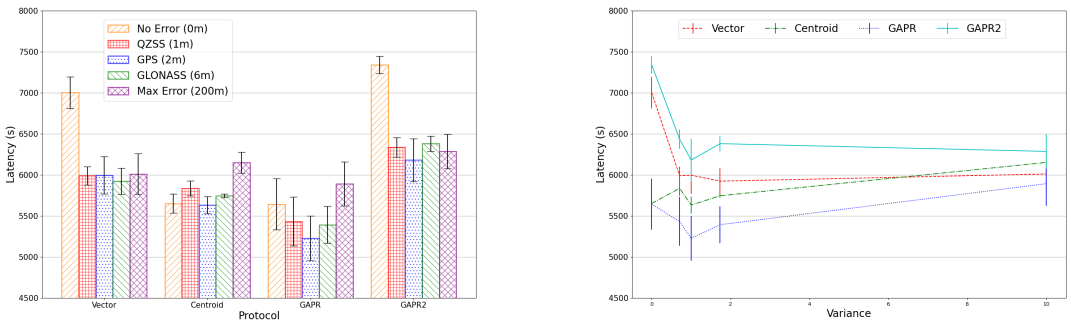
(d) Goodput vs. Variance

Figure 4.15. Bold Alligator MDR/Goodput: Protocol Performance Comparison (with 95% confidence intervals shown)

This is likely an artifact of Vector's and GAPR's shared latency phenomenon, which we discussed in Sections 4.3.2 and 4.6.2. Because the total number of messages transmitted when no error is introduced is significantly less, successful message deliveries are reduced affecting MDR. The goodput results are significant because each protocol shows an upward trend as more error is introduced. This is likely an artifact of the Bold Alligator mobility scenario because of its consistent node movement patterns combined with its higher participating node count. Disregarding the trends and looking strictly at protocol performance, GAPR appears to have the best performance across all variances. Each protocol shows an upward trend in goodput performance as error is introduced.

4.7.4 Helsinki Latency Comparison

Figure 4.16 depicts each protocol's latency performance while running the Helsinki mobility scenario. As discussed in Sections 4.3.2 and 4.6.2, we see the no error latency spike caused by fewer total message transmissions correlating to more messages staying in a node's buffer. Latency by nature, is an inconsistent metric because as more entropy is introduced to a network, latency tends to fluctuate. This is why we see larger error bars with latency than we do with the other observed metrics. The results indicate that GAPR performs the best with the lowest latency across each each variance.

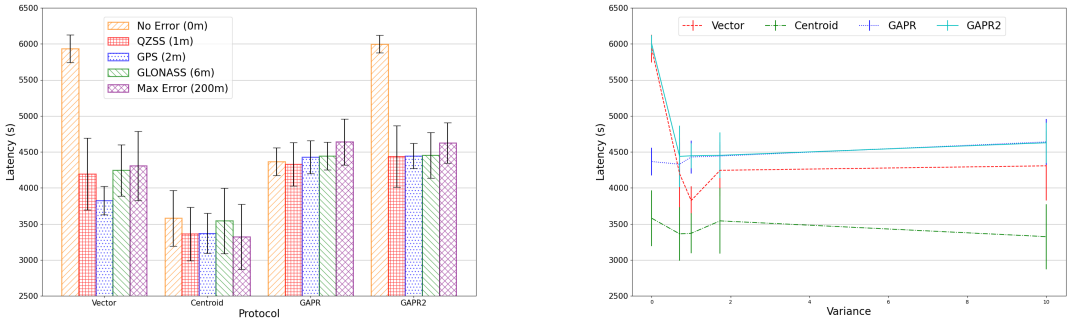


(a) Latency vs. Protocol (b) Latency vs. Variance

Figure 4.16. Helsinki Latency: Protocol Performance Comparison (with 95% confidence intervals shown)

4.7.5 Omaha Latency Comparison

Figure 4.5 depicts each protocol's latency performance while running the Omaha mobility scenario. Similar to Helsinki, we see the no error spike with Vector and GAPR2 for the same reason as discussed in Sections 4.3.2 and 4.6.2. We observe consistent latency across each variance above no error, across each protocol, with Centroid having the best latency performance. Unfortunately Centroid has the best latency because it has the lowest MDR, i.e. messages that remain in the network longer tend to be dropped before arriving at their destination. GAPR is the most consistent in the Omaha scenario with each of the variance data points within the error bars. If we consider Centroid and GAPR's MDRs, we see that Centroid is about two thirds of GAPR. Therefore, we consider GAPR as the best performing protocol with respect to latency in the Omaha mobility scenario.



(a) Latency vs. Protocol

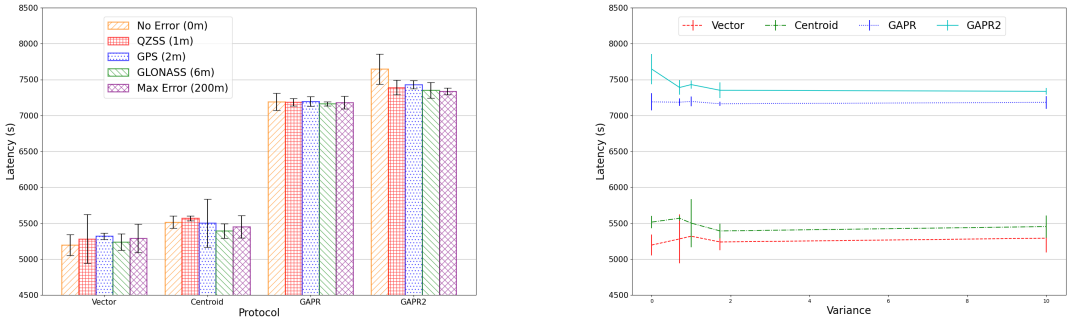
(b) Latency vs. Variance

Figure 4.17. Omaha Latency: Protocol Performance Comparison (with 95% confidence intervals shown)

4.7.6 Bold Alligator Latency Comparison

Figure 4.18 depicts each protocol's latency performance while running the Bold Alligator mobility scenario. Here we see similarities in both GAPR and GAPR2 performance. This is likely caused by more control messages being exchanged during short node encounters with fewer messages being exchanged. Both GAPR and GAPR2 send out a series of control messages that are necessary in determining their message delivery probabilities. If the routing primitive threshold is not met, the messages are not exchanged, accounting for the

larger latencies we see in both protocols. Vector and Centroid both deliver more messages, freeing up their buffers. Therefore we see the correlation to a smaller latency metric. Because of this, Vector performs the best with respect to latency in the Bold Alligator mobility scenario.



(a) Latency vs. Protocol

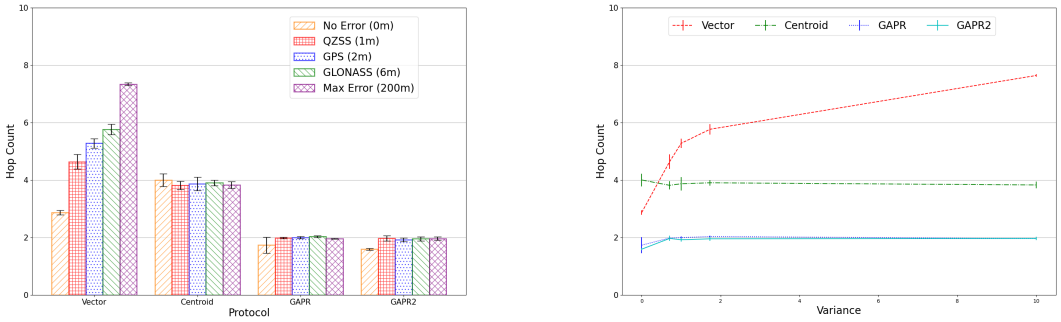
(b) Latency vs. Variance

Figure 4.18. Bold Alligator Latency: Protocol Performance Comparison (with 95% confidence intervals shown)

4.7.7 Helsinki Hop Count Comparison

Figure 4.19 depicts each protocol's hop count performance while running the Helsinki mobility scenario. Traditional network paradigms tend to correlate a higher hop count with increased latency. We observe that not to be true in this case. Latency across each protocol is consistent across all variances, with the exception of Vector and GAPR's no error latency spike. The results tell us that latency and hop count are not necessarily dependent on each other in this mobility scenario. Vector, for example, has consistent latency with QZSS error and beyond, yet we see an upward trend in hop count. This suggests that even the smallest of errors in Vector drastically effects its hop count performance. We observed this in Section 4.3.3 and correlated the upward trend to Helsinki's increased node entropy. As Vector encounters more orthogonal nodes, it exchanges more messages, increasing the message hop count in the process. Centroid, GAPR, and GAPR2 each remain consistent regardless of which variance is introduced. Both GAPR and GAPR2 perform the best with hop counts consistently hitting, at most, two hops between a message source and

destination. This is likely an artifact of combining the two-hop optimization mechanism and the delivery probability reset mechanism that resets probabilities once a three-hop limit has been reached. Considering these observations, we consider both GAPR and GAPR2 as the best performing protocols with respect to hop count in the Helsinki mobility scenario.



(a) Hop Count vs. Protocol

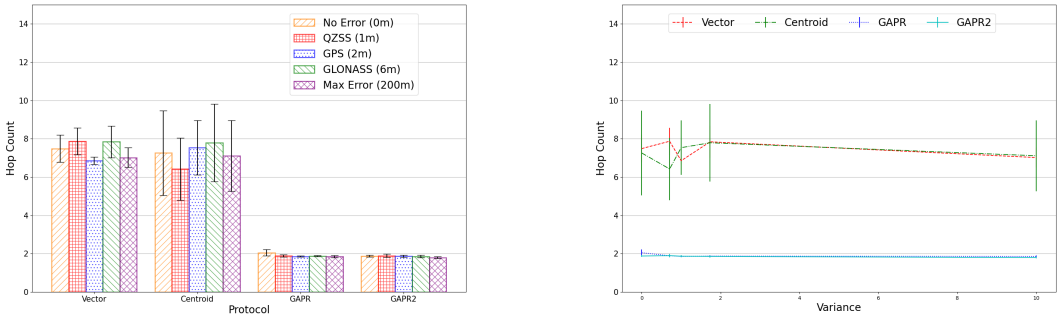
(b) Hop Count vs. Variance

Figure 4.19. Helsinki Hop Count: Protocol Performance Comparison (with 95% confidence intervals shown)

4.7.8 Omaha Hop Count Comparison

Figure 4.20 depicts each protocol's hop count performance while running the Omaha mobility scenario. Again, we observe both GAPR and GAPR2 hop count performance consistently hitting two hops across each variance for the same reasons as discussed in Section 4.7.7. A key difference between Vector's performance in Helsinki and Omaha, is we do not see the upward trend in hop count as more error is introduced. This is likely an artifact of the Omaha scenario and the consistent paths that participating nodes take during the simulation. Nodes moving in the scenario often do not meet the orthogonality threshold, therefore, fewer messages are exchanged between nodes. As a result, Vector does not propagate small errors throughout the network, as it does in the Helsinki scenario, and remains consistent across each error variance. Centroid performs slightly worse in Omaha compared to Helsinki and has large error bars. This can be correlated to the participating nodes in the scenario. Because ships are static, and amphibious transports move along the shoreline, Centroid's centroid list averages out error with each node encounter over time but

immediately gets the error back, causing a larger fluctuation of hop count in the process. Based on these observations, GAPR and GAPR2 perform the best in the Omaha mobility scenario.



(a) Hop Count vs. Protocol

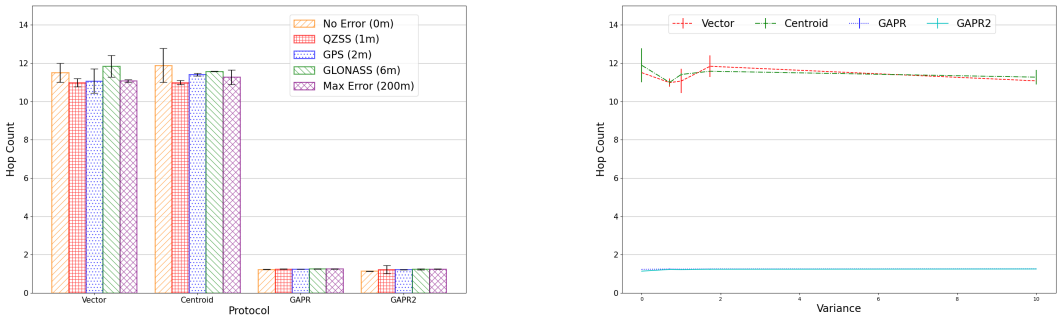
(b) Hop Count vs. Variance

Figure 4.20. Omaha Hop Count: Protocol Performance Comparison (with 95% confidence intervals shown)

4.7.9 Bold Alligator Hop Count Comparison

Figure 4.21 depicts each protocol's hop count performance while running the Bold Alligator mobility scenario. Here we immediately observe the inverse relationship of Bold Alligator's latency to hop count. Both Vector and Centroid have small latencies, yet both have significantly higher hop counts. This suggests that both protocols exchanged more messages across all variances, clearing out the node's buffer, decreasing latency in the process. We see the largest observed hop counts across all scenarios with Vector and Centroid, suggesting that they are acting more like *Epidemic*, flooding messages across the network. This is likely an artifact of the Bold Alligator scenario because of the consistent paths that participating nodes traverse. The Bold Alligator scenario consists of six different node types, each with varying buffer sizes and bandwidths, adding the likelihood of an increasing number of relays a message must traverse before reaching its destination. The same inverse relationship is observed with both GAPR and GAPR2. Both protocols have higher latencies and significantly lower hop counts. The two hop optimization is dominating the performance of the protocol with respect to hop count acting almost like direct contact routing, where there is exactly one relay, the destination. Because of this increase in latency, it suggests that both GAPR

and GAPR2 retain their messages until they are either in direct contact with the destination or are within one hop of the destination. Based on these observations, GAPR and GAPR2 perform the best in the Bold Alligator mobility scenario with respect to hop count.



(a) Hop Count vs. Protocol

(b) Hop Count vs. Variance

Figure 4.21. Bold Alligator Hop Count: Protocol Performance Comparison (with 95% confidence intervals shown)

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusion and Future Work

In this final chapter, we provide our conclusions. No additional data is provided in this chapter, as conclusions are drawn from Chapter 4. Section 5.1 presents conclusions based on our observations of introduced GNSS PNT positional error on each of the four geographic routing protocols of study. We discuss observed unique performance characteristics across each metric of study, including which protocols performed the best within each mobility scenario. We then conclude with potential topics for future study.

5.1 Conclusions

It was readily apparent from our results that introduced GNSS PNT positional error affects geographic routing protocols, however, the observed effect did not completely align with what we had initially hypothesized. Across each of the mobility scenarios and metrics, the protocols of study were *either* positively or negatively affected by introduced error. This left us with more questions than answers. It suggests that no single protocol tested is a good fit for all mobility scenarios tested. Specifically, in some scenarios it seems a routing protocol would making bad forwarding decisions, and actually performed better when those decisions were randomized by introducing PNT error. This was true with respect to both GAPR and GAPR2's MDR and goodput performance when running the Helsinki and Bold Alligator mobility scenarios. In both cases, the addition of error increased both metrics by nearly 15%. In the case of the Omaha mobility scenario, introduced error decreased both MDR and goodput for Vector, GAPR, and GAPR2.

When looking at latency within the Helsinki and Omaha mobility scenarios, we observed a characteristic shared between Vector and GAPR2. When no error was introduced, latency was observed to spike by nearly 1000 s. It was determined to be an artifact of Vector's orthogonal routing primitive where messages are not forwarded, causing the associated spike in latency. When error was introduced, Vector and GAPR2 assumed a neighboring node's direction and velocity were orthogonal to its own, increasing the probability of exchanging messages for a given encounter, and decreasing latency in the process. We

saw this in the aggregated data with a tripled message transmission count once error was applied. The addition of error positively affected both protocol's latency performance within the Helsinki and Omaha mobility scenarios. We observed that Vector's hop count was negatively affected by introduced error, suggesting that even the smallest of errors affect the protocol's hop count performance significantly.

In the Bold Alligator mobility scenario, we observed that both Vector and Centroid performed like *Epidemic* with respect to hop count. Conversely, both GAPR and GAPR2 performed like direct contact with a single hop between source and destination. Considering these observations, we conclude that introducing error on geographic routing protocols has an effect, however, it is likely that the mobility models themselves have more drastic effects on performance. Furthermore, protocols perform better in different mobility scenarios and suggest that protocols should be tailored to a specific mobility scenario.

We also observed a difference in performance between simulators. Previous GNSS PNT positional error studies were conducted in the ONE simulator, and when compared to the results of our data from ns-3, we observed decreasing performance across each protocol. Because ns-3 simulates the entire network stack, we concluded that our choice of simulator provided a more realistic environment for testing our positional error model. Furthermore, we concluded that the choice of simulator is dependent on the scenario, the protocol or study, and the focus of the research.

5.2 Future Work

As the field DTN research continues to mature, researchers are left with more questions than answers. That was the case with our research. We sought to answer the question of whether or not geographic routing protocols performance degraded due to error in their positional inputs from GNSS but found the answer to be both yes and no. Based on the varying positive and negative results we observed in our study, we recommend evaluating protocols against individual mobility scenarios, and determining which geographic routing protocol fits the scenario the best. Doing so will likely provide more consistent results across each error model when they are introduced. We are not the first to point out this option as an area of future work [35], so there is potential in this approach. Second, a multi-protocol adaptation within the Bold Alligator scenario, although complicated, would provide a unique perspec-

tive for the observation of introducing positional error. The model would consist of different node clusters implementing different geographic routing protocols. Third, simulating the effects of GNSS PNT error caused by atmospherics on geographic routing protocols. This would require knowledge of weather conditions and signal propagation schemes that would then be coded into ns-3. This would be a more realistic approach to the introduction of positional error on geographic routing protocols in the real-world. Collaboration with a student from the Meteorology and Oceanography (METOC) community would provide critical insight into meteorology that would significantly aid in this line of research. Fourth, comparing and contrasting geographic routing protocols to other DTN protocol strategies when positional error is introduced. Alternative protocols that do not rely on positional data to make routing decisions, and, instead, use other mechanisms available in the environment they are operating, should be considered and researched further. Last, studying the effects of GNSS PNT positional error on other geographic routing protocols like CenterMass. This would require the translation of CenterMass ONE's implementation from Java into ns-3's implementation of C++. Each of these suggestions for future work were outside the scope of this study but are certainly areas that would broaden the work conducted in this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A:
Helsinki Scenario Data

A.1 Aggregate Scenario Data

Table A.1. Vector Data

Parameter	No Error (0m)	QZSS (1m)	GPS (2m)	GLONASS (6m)	Max Error (200m)
Average Messages Generated	1431.5	1430	1431.5	1431.5	1431.5
Average Messages Transmitted	1268.5	1272.75	1292.75	1298.75	1310.25
Average Unique Messages Delivered	869.75	908.75	914.75	883.5	848.5
Average Messages Dropped	2.25	1.25	4	4.25	7
Average Total Transmissions	12662	34192	42171.25	43375.5	57239.25
Average Total Receptions	10025.5	29002.75	36466.75	36834	47769.75
Average MDR	0.607574829	0.63755592	0.639008175	0.617167212	0.592713491
Average Latency Mean (s)	7066.107303	5991.99006	5960.366753	5909.011033	6011.25732
Average Goodput Mean (B/s)	20.51563079	21.43556134	21.57708912	20.83996528	20.01438657
Average Hop Count Mean	2.869762673	4.660274289	5.18327212	5.72304062	7.359016992

Table A.2. Centroid Data

Parameter	No Error (0m)	QZSS (1m)	GPS (2m)	GLONASS (6m)	Max Error (200m)
Average Messages Generated	1431.5	1430	1431.5	1431.5	1431.5
Average Messages Transmitted	1051.75	1035.75	1046.5	1047.5	1033.25
Average Unique Messages Delivered	774.75	775.75	771.5	792	737
Average Messages Dropped	4.75	4.25	4.5	5	4
Average Total Transmissions	27384	26775.5	28363.5	27604.25	25051.5
Average Total Receptions	23137.75	23177.75	23879	23172	20907
Average MDR	0.541222589	0.544736195	0.53891925	0.553235949	0.509799675
Average Latency Mean (s)	5582.354343	5777.589239	5668.500598	5740.247202	6191.08615
Average Goodput Mean (B/s)	18.27477431	18.29836227	18.19811343	18.68166667	17.3843287
Average Hop Count Mean	3.967243676	3.787538153	3.874281405	3.902286099	3.878056255

Table A.3. GAPR Data

Parameter	No Error (0m)	QZSS (1m)	GPS (2m)	GLONASS (6m)	Max Error (200m)
Average Messages Generated	1432.25	1430.25	1431.5	1432.25	1431.5
Average Messages Transmitted	1344	1372	1355.5	1263	1357.5
Average Unique Messages Delivered	991.75	1061.5	1045.75	960.25	989.25
Average Messages Dropped	1.25	3	3	2.75	3
Average Total Transmissions	117206.75	107750.25	110489.5	103578.75	98133
Average Total Receptions	99886.75	292454	93048.75	87574.75	81613.5
Average MDR	0.685638672	0.749097707	0.730513685	0.727567947	0.691049441
Average Latency Mean (s)	5694.420169	5481.713421	5209.495458	5376.229147	5827.151345
Average Goodput Mean (B/s)	23.39336227	25.03862269	24.66711227	22.65034144	23.33439236
Average Hop Count Mean	1.813432291	1.974623368	1.998873461	2.00680413	1.961182324

Table A.4. GAPR2 Data

Parameter	No Error (0m)	QZSS (1m)	GPS (2m)	GLONASS (6m)	Max Error (200m)
Average Messages Generated	1431.5	1430	1433.25	1430	1430
Average Messages Transmitted	1271.25	1335	1324.5	1343.5	1339.75
Average Unique Messages Delivered	892.75	1039.75	1015.75	1023.25	995.25
Average Messages Dropped	1.5	3	2.5	1.25	1.75
Average Total Transmissions	13032.25	30322	33991	38379	47478
Average Total Receptions	10682.5	25969.25	29183.75	32801.5	40533.5
Average MDR	0.629565346	0.727081554	0.706012403	0.712429496	0.694289388
Average Latency Mean (s)	7326.938408	6367.173644	6178.273869	6288.699079	6254.766031
Average Goodput Mean (B/s)	21.05815394	24.52558449	23.95947338	24.1363831	23.47592014
Average Hop Count Mean	1.596499941	1.956556448	1.820352477	1.941865265	1.964810871

APPENDIX B:
Omaha Scenario Data

B.1 Aggregate Scenario Data

Table B.1. Vector Data

Parameter	No Error (0m)	QZSS (1m)	GPS (2m)	GLONASS (6m)	Max Error (200m)
Average Messages Generated	1427.25	1427.25	1427.25	1427.25	1427.25
Average Messages Transmitted	1202.25	1243.75	1224.75	1228.75	1206.75
Average Unique Messages Delivered	320.5	262.5	241.75	253.5	256.75
Average Messages Dropped	79.5	53.5	70.5	68	78.25
Average Total Transmissions	76518.5	110337.25	111338.5	108273.25	113068.75
Average Total Receptions	55517.5	64155	64905.5	63905.75	65744.5
Average MDR	0.224575205	0.183933227	0.169384858	0.17762959	0.179919977
Average Latency Mean (s)	5912.954115	4193.314256	3867.489842	4155.902108	4298.909001
Average Goodput Mean (B/s)	7.55994213	6.191840278	5.702390046	5.979548611	6.056209491
Average Hop Count Mean	7.49992664	7.565445944	6.75897675	7.805190211	7.125740028

Table B.2. Centroid Data

Parameter	No Error (0m)	QZSS (1m)	GPS (2m)	GLONASS (6m)	Max Error (200m)
Average Messages Generated	1427.25	1427.456	1427.25	1427.25	1427.25
Average Messages Transmitted	868.5	885.75	889.5	890.75	900.75
Average Unique Messages Delivered	168	161.25	170.5	173.75	175.25
Average Messages Dropped	3.25	3.75	4	3.5	4.25
Average Total Transmissions	43070	48608	51005	50666.75	51122.75
Average Total Receptions	24760.75	29723	30707	30581	30958.5
Average MDR	0.117708951	0.112984613	0.119467489	0.121744134	0.122790891
Average Latency Mean (s)	3576.943312	3270.609701	3320.389189	3506.320949	3410.766294
Average Goodput Mean (B/s)	3.962777778	3.803559028	4.021747685	4.098408565	4.133790509
Average Hop Count Mean	7.340095194	6.443146387	7.468596933	7.986791456	7.937247602

Table B.3. GAPR Data

Parameter	No Error (0m)	QZSS (1m)	GPS (2m)	GLONASS (6m)	Max Error (200m)
Average Messages Generated	1427.25	1427.25	1427.25	1427.25	1427.25
Average Messages Transmitted	964.75	983.25	962.5	975.5	967
Average Unique Messages Delivered	325.25	260	254.75	259.75	243.5
Average Messages Dropped	265.5	211.25	219	200.75	195
Average Total Transmissions	150852.75	155099.25	156478	157109.75	155850.5
Average Total Receptions	139494.25	142936	144003.75	145500.5	144077.5
Average MDR	0.227891844	0.182183013	0.178501018	0.182009653	0.170622492
Average Latency Mean (s)	4595.580663	4248.259062	4318.632356	4335.117084	4436.279543
Average Goodput Mean (B/s)	7.671984954	6.13287037	6.009033565	6.12697338	5.743668981
Average Hop Count Mean	2.075077363	1.891673244	1.856092249	1.861523521	1.852503179

Table B.4. GAPR2 Data

Parameter	No Error (0m)	QZSS (1m)	GPS (2m)	GLONASS (6m)	Max Error (200m)
Average Messages Generated	1427.25	1427.25	1427.25	1427.25	1427.25
Average Messages Transmitted	928	974	975.75	980	970.5
Average Unique Messages Delivered	299.5	260.5	261	265.25	258.75
Average Messages Dropped	68.75	119.5	168	165.25	154.75
Average Total Transmissions	87555	120938.75	123542	123486	130069
Average Total Receptions	83483.75	112337.25	114475.5	114271.5	120234.25
Average MDR	0.209856771	0.182540017	0.182878527	0.185859252	0.181308284
Average Latency Mean (s)	5990.355025	4310.910358	4530.722518	4397.608598	4614.890634
Average Goodput Mean (B/s)	7.064594907	6.144664352	6.156458333	6.256707176	6.103385417
Average Hop Count Mean	1.905939987	1.883376927	1.856706263	1.857560841	1.821328262

APPENDIX C:
Bold Alligator Scenario Data

C.1 Aggregate Scenario Data

Table C.1. Vector Data

Parameter	No Error (0m)	QZSS (1m)	GPS (2m)	GLONASS (6m)	Max Error (200m)
Average Messages Generated	18584	18584	18584	18584	18584
Average Messages Transmitted	10096	12679.5	12759	12717	13096
Average Unique Messages Delivered	1645	1861.5	1905	1875.5	1943
Average Messages Dropped	67791	52770	41688	61610	60888
Average Total Transmissions	317133	637365	655231.5	6667337.5	738858
Average Total Receptions	302375	610686.5	727368	633571	702313
Average MDR	0.088516209	0.100166763	0.102507063	0.100919948	0.10455254
Average Latency Mean (s)	5195.224142	5281.321548	5318.778093	5238.017551	5291.173213
Average Goodput Mean (B/s)	38.8022	43.90899306	44.93506944	44.23922454	45.83141204
Average Hop Count Mean	11.50745651	10.98587339	11.06753702	11.83952932	11.07822114

Table C.2. Centroid Data

Parameter	No Error (0m)	QZSS (1m)	GPS (2m)	GLONASS (6m)	Max Error (200m)
Average Messages Generated	18584	18584	18584	18584	18584
Average Messages Transmitted	65190	11187.5	11007	11247	11283.5
Average Unique Messages Delivered	1812	1921	1891	1906.5	1899.5
Average Messages Dropped	13250.5	67791	51631.5	98286	70274
Average Total Transmissions	577669	662013.5	635874	698148	702015
Average Total Receptions	556794.5	637725	612180	672759	676283
Average MDR	0.097503187	0.103368309	0.101753669	0.102587679	0.102211556
Average Latency Mean (s)	5515.165952	5568.775419	5500.851887	5391.707386	5452.348937
Average Goodput Mean (B/s)	42.74138889	45.31247685	44.60483796	44.97045139	44.80533565
Average Hop Count Mean	11.89074013	10.9967134	11.41181849	11.58097573	11.27174555

Table C.3. GAPR Data

Parameter	No Error (0m)	QZSS (1m)	GPS (2m)	GLONASS (6m)	Max Error (200m)
Average Messages Generated	18584	18584	18584	18584	18584
Average Messages Transmitted	11054.5	10807.5	11077	11052.5	11086.5
Average Unique Messages Delivered	1948.5	2038	2052	2069	2112
Average Messages Dropped	28610.5	35400.5	54230.5	78893	73825
Average Total Transmissions	841463.5	789645	920088	962665	963412
Average Total Receptions	828890.5	775835.5	903158	943602.5	943857
Average MDR	0.104847769	0.109664707	0.110417592	0.111331906	0.113645851
Average Latency Mean (s)	7189.260162	7305.122324	7195.114826	7162.724522	7182.644363
Average Goodput Mean (B/s)	45.96114583	48.07226852	48.4025	48.80349537	49.81777778
Average Hop Count Mean	1.233606449	1.244242386	1.246341226	1.257241216	1.259702081

Table C.4. GAPR2 Data

Parameter	No Error (0m)	QZSS (1m)	GPS (2m)	GLONASS (6m)	Max Error (200m)
Average Messages Generated	18584	18580	18584	18584	18584
Average Messages Transmitted	9612.5	10548.5	10705.5	10819	10979
Average Unique Messages Delivered	1797.5	2008	2035	2014.5	2024
Average Messages Dropped	1896	32132.5	23961.5	60961.5	31503.5
Average Total Transmissions	368682	663608.5	673658	688472.5	728840.5
Average Total Receptions	364309.5	653563	663638.5	677181	717550
Average MDR	0.096723218	0.108050068	0.109502629	0.108399524	0.108910722
Average Latency Mean (s)	7645.588465	7389.661671	7430.16444	7350.767935	7336.552457
Average Goodput Mean (B/s)	42.39936343	47.36462963	48.00150463	47.51795139	47.74203704
Average Hop Count Mean	1.140181036	1.229811657	1.222168558	1.240104289	1.253666373

List of References

[1] V. Cerf, S. Burleigh, A. Hooke, J. Torgerson, R. Durst, K. Scott, E. Travis, and H. Weiss. (2001, January). Interplanetary internet (IPN): Architectural Definition. *IPN Research Group*. [Online]. *1*(1). Available: https://tools.ietf.org/pdf/draft-irtf-ipnrg-arch-00.pdf

[2] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2003, p. 27–34.

[3] J. Postel, "Transmission control protocol," Internet Requests for Comments, RFC Editor, RFC 793, September 1981. Available: http://www.rfc-editor.org/rfc/rfc793.txt

[4] J. Postel, "User datagram protocol," Internet Requests for Comments, RFC Editor, RFC 768, August 1980. Available: http://www.rfc-editor.org/rfc/rfc768.txt

[5] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," *Technical Report*, June 2000.

[6] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-tolerant networking architecture," Internet Requests for Comments, RFC Editor, RFC 4838, April 2007. Available: http://www.rfc-editor.org/rfc/rfc4838.txt

[7] K. Scott and S. Burleigh, "Bundle protocol specification," Internet Requests for Comments, RFC Editor, RFC 5050, November 2007. Available: http://www.rfc-editor.org/rfc/rfc5050.txt

[8] S. Ruhrup, "Theory and practice of geographic routing," University of Freiburg, Breisgau, Germany, Tech. Rep., February 2009.

[9] J. Rohrer, "Effects of gps error on geographic routing," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–2.

[10] H. Kang and D. Kim, "Vector routing for delay tolerant networks," in *2008 IEEE 68th Vehicular Technology Conference*, 2008, pp. 1–5.

[11] J. Rohrer, "Geographic centroid routing for vehicular networks," in *Proceedings of the Seventh International Conference on Advances in Vehicular Systems, Technologies and Applications (VEHICULAR)*, 2018.

[12] K. Killeen, "Gapr2: A dtn routing protocol for communications in challenged, degraded, and denied environments," master's thesis, Naval Postgraduate School, 2015.

[13] A. Keranen, J. Ott, and T. Karkkainen, "The one simulator for dtn protocol evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, 2009, pp. 1–10.

[14] J. Rohrer, R. Beverly, and G. Xie, "Gelocation assisted predictive routing (gapr) protocol for heterogeneous dtn mobility patterns," *2013 International Conference on Connected Vehicles and Expo (ICCVE)*, December 2013.

[15] G. Riley and T. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*. Springer, 2010, pp. 15–34.

[16] J. Cohenour, "Global positioning system clock and orbit statistics and precise point positioning," Ph.D. dissertation, Russ College of Eng., Ohio Univ., Athens, OH, 2009.

[17] M. Bellis. (1980, July). ARPAnet: The World's First Internet. *Thought.Co.* [Online]. *1*(3). Available: https://www.thoughtco.com/arpanet-the-worlds-first-internet-4072558

[18] S. Gallagher. (2019, October). 50 Years Ago Today the Internet was Born. Sort of. [Online]. Available: https://arstechnica.com/information-technology/2019/10/50-years-ago-today-the-internet-was-born-sort-of/

[19] S. Lukasik, "Why the arpanet was built," *IEEE Annals*, vol. 33, no. 3, pp. 4–21, March 2011.

[20] D. B. Johnson and D. A. Maltz, "Truly seamless wireless and mobile host networking. protocols for adaptive wireless and mobile networking," *IEEE Personal Communications*, vol. 3, no. 1, pp. 34–42, February 1996.

[21] C.-K. Toh, *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Upper Saddle River, NJ: Prentice Hall, P.T.R., 2007.

[22] Interplanetary Networking Special Interest Group (IPNSIG). (2019). [Online]. Available: http://www.ipnsig.org/. Accessed Jan. 13, 2020.

[23] Delay-Tolerant Networking Research Group (DTNRG). (2019). [Online]. Available: https://irtf.org/concluded/dtnrg. Accessed Jan. 13, 2020.

[24] A. Turner. (2020, April). How Many Smartphones are in the World? [Online]. Available: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

[25] A. Galati, T. Bourchas, S. Siby, S. Frey, M. Olivares, and S. Mangold, "Mobile-enabled delay tolerant networking in rural developing regions," in *IEEE Global Humanitarian Technology Conference (GHTC 2014)*, 2014, pp. 699–705.

[26] A. Pentland, R. Fletcher, and A. Hasson, "Daknet: Rethinking connectivity in developing nations," *IEEE Computer Society*, vol. 37, no. 1, pp. 78–83, August 2004.

[27] A. K. Gupta, J. K. Mandal, and I. Bhattacharya, "Sensor node assisted dtn for a post-disaster scenario," in *Proceedings of the 20th International Conference on Distributed Computing and Networking*, 2019, p. 401–404.

[28] E. Krotkov and J. Blitch, "The defense advanced research projects agency (darpa) tactical mobile robotics program," *The International Journal of Robotiocs Research*, vol. 18, no. 7, pp. 769–776, July 1999.

[29] J. Beauquier, P. Blanchard, J. Burman, and S. Delaet, "Tight complexity analysis of population protocols with cover times — the zebranet example," *Theoretical Computer Science*, vol. 512, no. 1, pp. 15–27, November 2013.

[30] A. Hodges. (2000). *Alan Turing: The Enigma*. [Ebrary version]. [Online]. Available: https://www.ebook.de/de/product/22531952/andrew_hodges_alan_turing_the_enigma.html

[31] S. Misra, *Network Routing: Fundamentals, Applications and Emerging Technologies*. West Sussex, UK: John Wiley & Sons, Inc., 2017.

[32] J. Pelkey. (2007). Entrepreneurial Capitalism and Innovation: A History of Computer Communications 1968-1988. [Online]. Available: http://www.historyofcomputercommunications.info/Book/2/2.1-IntergalacticNetwork_1962-1964.html

[33] I. Akyildiz, J. Jornet, and M. Pierobon, "Nanonetworks: A new frontier in communications," *Communications of the ACM*, vol. 54, no. 11, pp. 84–89, November 2011.

[34] V. Cerf and R. Kahn, "A protocol for packet network intercommunication," *Institute of Electrical and Electronics Engineers (IEEE*, vol. 22, no. 5, pp. 637–648, May 1974.

[35] S. Misra, D. Saha, and S. Pal, *Opportunistic Mobile Networks*. West Bengal, India: Springer International Publishing, 2016.

[36] L. Gao, S. Yu, T. H. Luan, and W. Zhou, *Delay Tolerant Networks*. Burwood, VIC, Australia: Springer International Publishing, 2015.

[37] E. Jones and P. Ward, "Routing strategies for delay-tolerant networks," *Submitted to ACM Computer Communication Review (CCR)*, 2006.

[38] 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*.

[39] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, July 2004.

[40] J. Burgess, B. Gallagher, D. Jensen, and B. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, 2006, pp. 1–11.

[41] A. Balasubramanian, B. Levine, and A. Venkataramani, "Dtn routing as a resource allocation problem," *Communications of the ACM*, vol. 37, no. 4, pp. 373–384, July 2007.

[42] P. Mishra, C. Gandhi, and B. Singh, "An improved greedy forwarding scheme in manets," *Journal of Telecommunications and Information Technology*, vol. 1, no. 1, pp. 50–55, 2017.

[43] J. Vincent. (2015, April). The Most Accurate Clock Ever Built Only Loses One Second Every 15 Billion Years. [Online]. Available: https://www.theverge.com/2015/4/22/8466681/most-accurate-atomic-clock-optical-lattice-strontium

[44] How Does GPS Network Time Synchronization Work? (n.d.). Masterclock. [Online]. Available: https://www.masterclock.com/support/library/gps-network-time-synchronization. Accessed Feb. 12, 2020.

[45] What is Positioning, Navigation, and Timing (PNT). (n.d.). [Online]. Available: https://www.transportation.gov/pnt/what-positioning-navigation-and-timing-pnt. Accessed Jan. 15, 2020.

[46] J. J. Sellers, *Understanding Space: An Introduction to Astronautics*, 3rd ed. New York, NY: McGraw-Hill Companies, 2005.

[47] B. Ben-Moshe, E. Elkin, H. Levi, and A. Weissman, "Improving accuracy of gnss devices in urban canyons," in *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry, CCCG 2011*, 2011, pp. 429–442.

[48] Global Positioning System (GPS). (n.d.). [Online]. Available: https://www.gps.gov/systems/gps/performance/accuracy/. Accessed Jan. 15, 2020.

[49] Globalnaya Navigatsionnaya Sputnikovaya Sistema (GLONASS). (n.d.). [Online]. Available: https://www.glonass-iac.ru/en/GLONASS/index.php. Accessed Jan. 15, 2020.

[50] Galileo European Satellite Based Navigation System. (n.d.). [Online]. Available: https://www.gsa.europa.eu/european-gnss/galileo/galileo-european-global-satellite-based-navigation-system. Accessed Jan. 15, 2020.

[51] BeiDou Navigation Satellite System (BDS). (n.d.). [Online]. Available: http://en.beidou.gov.cn/. Accessed Jan. 15, 2020.

[52] Navigation Indian Constellation (NAVIC). (n.d.). [Online]. Available: https://gssc.esa.int/navipedia/index.php/NAVIC. Accessed Jan. 15, 2020.

[53] Quasi-Zenith Satellite System (QZSS). (n.d.). [Online]. Available: https://qzss.go.jp/en/overview/services/sv01_what.html. Accessed Jan. 15, 2020.

[54] N. R. Council, *The Global Positioning System: A Shared National Asset*. Washington, DC: The National Academies Press, 1997.

[55] Statement by the Press Secretary of President Bush. [Online]. Available: https://georgewbush-whitehouse.archives.gov/news/releases/2007/09/20070918-2.html. Accessed Jan. 15, 2020.

[56] C. Albon, "Global positioning system: Updated schedule assessment cloud help decision makers address likely delays related to new ground control system," United States Government Accountability Office, San Francisco, CA, Tech. Rep. GAO-19-250, May 2015.

[57] R. Constantine. (2008). *GPS and Galileo Friendly Foes?* [Ebrary version]. [Online]. Available: https://media.defense.gov/2017/Nov/22/2001847932/-1/-1/0/WP_0012_CONSTANTINE_GPS_AND_GALILEO.PDF

[58] J. Postel, "Internet protocol," Internet Requests for Comments, RFC Editor, RFC 791, September 1981. Available: http://www.rfc-editor.org/rfc/rfc791.txt

[59] N. Meghanathan, "Performance comparison of link, node and zone disjoint multipath routing strategies and minimum hop single path routing for mobile ad hoc networks," *International Journal of Wireless & Mobile Networks*, vol. 2, October 2010.

[60] R. Poor, "Gradient routing in ad hoc networks," *MIT Media Laboratory*.

[61] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 145–158, September 2004.

[62] E. Kranakis, H. Singh, and J. Urrutia, "Compass routing on geometric networks," in *11th Canadian Conference on Computational Geometry*, 1999, pp. 51–54.

[63] Z. J. Haas and T. Small, "A new networking model for biological applications of ad-hoc sensor networks," *IEEE/ACM Transactions on Networking*, vol. 14, no. 1, pp. 27–40, Febryary 2006.

[64] P. Mundur, M. Seligman, and G. Lee, "Epidemic routing with immunity in delay tolerant networks," in *MILCOM 2008 - 2008 IEEE Military Communications Conference*, 2008, pp. 1–7.

[65] S. Grasic, E. Davies, A. Lindgren, and A. Doria, "The evolution of a dtn routing protocol - prophetv2," in *Proceedings of the 6th ACM Workshop on Challenged Networks*, 2011, pp. 27–30.

[66] S. Nelson, M. Bakht, and R. Kravets, "Encounter-based routing in dtns," in *IEEE INFOCOM 2009*, 2009, pp. 846–854.

[67] D. Bucur, G. Iacca, G. Squillero, and A. Tonda, "Black holes and revelations: Using evolutionary algorithms to uncover vulnerabilities in disruption-tolerant networks," in *Applications of Evolutionary Computation*, 2015, pp. 29–41.

[68] S. Saha, R. Verma, S. Saika, P. Paul, and S. Nandi, "e-one:enhanced one for simulating challenged network scenarios," *Journal of Networks*, vol. 9, December 2014.

[69] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 2575–2582.

[70] *ns-3 Manual*, accessed Jan. 30, 2020. Available: https://www.nsnam.org/docs/release/3.26/manual/ns-3-manual.pdf

[71] A. Mauldin, "Comparative analysis of disruption tolerant network routing simulations, and denied environments," master's thesis, Naval Postgraduate School, 2017.

[72] J. Brown, "An in-depth analysis of machine-learning-based network routing protocol for networking in a highly mobile topology," master's thesis, Naval Postgraduate School, 2018.

[73] Expeditionary Strike Group Two. (2011, December). Navy, Marine Corps Begin Bold Alligator 2011. [Online]. Available: http://www.army.mil/article/70358/Nano_technology_marches_on/

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California