Evaluation and Innovation in Opportunistic Networks

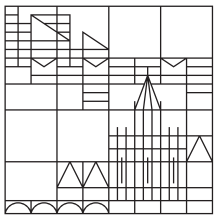
A Doctoral Dissertation submitted in partial satisfaction of the requirements for the degree of

Doctor of Engineering Science (Dr.-Ing.)
Doktor der Ingenieurwissenschaften

submitted by

Muhammad Arshad Islam
at

Universität Konstanz



Faculty of Sciences

Faculty of Computer and Information Science

Referees

Prof. Dr. Marcel Waldvogel
Prof. Dr. Ulrik Brandes

Defense date

30th of March 2011

Abstract

The presence of an increasing number of mobile devices has prompted the demand to use them for information dispersion through opportunistic networks, which they form coincidently due to their geographic location. Opportunistic networks pose several new challenges to the current transmission protocols as they are not only capable of *store and forward* routing, but also lack the offline routing capability, i.e. source and destination must be connected to the network simultaneously. We can find several opportunistic network protocols in literature, but neither a solid comparison nor a trusted baseline has been presented.
In this study, we have analyzed and evaluated eight existing routing algorithms on a common basis in an effort to grasp the strong and weak points of each of them and to see whether it is possible to design a hybrid technique that may take advantage of the strengths of several other techniques. We propose three important criticisms regarding the evaluation of existing routing algorithms.

1. Most evaluations restrict themselves to comparing against the two extremes, direct-contact forwarding and flooding.

2. Each attempt uses a completely different choice of scenario and simulation parameters.

3. Most attempts concentrate on methods to find a path to destination but the reliability of the path cannot be ensured in this paradigm.

The findings have revealed that almost all the techniques fail to perform under variable conditions, i.e. bandwidth hungry techniques failed to deliver when bottlenecks existed although, they outclassed every other technique where network had sufficient capacity. In contrast, techniques that required good network connectivity failed to perform in sparse network. As a result of our comparison of selected networks under a wide variety of realistic scenarios, we have not only been able to identify and describe favorable traits of protocols, but also necessary relationships of successful mobile opportunistic network protocols with QoS routing. This study defines a very light weight metric, which not only encapsulates the path bandwidth but also maintains a dynamic path ranking by degrading the path efficiency as it suffers from data load. Moreover, this study focuses on the routing algorithm Nile[1] that has an adapting capability with the underlying network. It thereby maintains acceptable performance without exhausting the network resources keeping a check on the network "pulse", i.e. bandwidth. Nile

[1]Motivation comes from the notion of controlling floods from the river Nile by constructing dams and barrages across the river banks.

is a multi-path protocol that deploys replication based on heuristic for computing disjoint path .
Flooding is considered to be a protocol that can deliver the best performance if its overhead is ignored. Therefore, it is customary to use flooding as a performance benchmark for opportunistic networks. We identify and describe the current simulation practices that do not expose the shortcomings of flooding as an upper bound. We provide a step towards a routing benchmark, which is flexible, provides results close to an upper bound, is simple to implement, and thus might be a candidate for a common benchmark. This new method called EPO[2], does not suffer from bottlenecks that limit the performance of epidemic flooding, even when bandwidth is scarce. Our analysis shows that networks are not suffering congestion as suggested by flooding, giving a better insight in the underlying network.
Since most of the practical routing protocols rely on history to profile devices, these profiles are used to compute the routes in the network. We found that history is either not able to predict device behavior accurately or history looses the details about the device behavior due to aggregation of metrics. We have therefore analyzed opportunistic network with max-flow to see the throughput of the network. We afterwards compared the outcome of this max-flow with a modified max-flow that uses history for its throughput computations. We have found that it is not possible to obtain an accurate history based max-flow, since it is not easy to find the link in path that can be considered to be responsible if history gives an unreliable path.

[2]Inspired by use of Erythropoietin as performance enhancement drug.

Zusammenfassung

Die ständige Zunahme der Anzahl mobiler Geräte führt zur Notwendigkeit diese auch im Rahmen *opportunistischer Netze* auch durch ihre geographische Lage und Mobilität zur Informationsverteilung zu nutzen. Opportunistische Netze stellen neue Herausforderungen, welche von aktuellen Protokollen nicht erfüllt werden. So benötigen diese eine ständige Netzverbindung zwischen Quelle und Ziel und sind nicht auf die in opportunistischen Netzen notwendigen *Store-and-Forward-Architektur*.
In dieser Dissertation werden erstmals acht existierende Algorithmen auf Basis von gemeinsamen Parametern analysiert und verglichen. Dadurch können Stärken und Schwächen der einzelnen Verfahren herausgearbeitet werden. Diese fliessen dann in einen neuen Algorithmus ein, welcher die Stärken vieler dieser Techniken vereint.Obwohl in der Literatur Protokolle für opportunistische Netze beschrieben sind, fehlt ein solider Vergleich oder auch nur eine geeignete Basis für einen solchen. Insbesondere liegen drei Probleme vor:

1. Die meisten Vergleiche beschränken sich auf die zwei Extreme Direktverbindung und Flooding.

2. Jeder Vergleich basiert auf einer einmaligen Wahl von Umgebung und Simulationsparemetern.

3. Darüberhinaus beschränken sich die bekannten Methoden darauf, einen Pfad zum Ziel zu finden, unabhängig ob über diesen Pfad auch genügend Daten übertragen werden können.

Die Resultate zeigen deutlich, dass die betrachteten Techniken sich variablen Umständen nicht anpassen können. So versagen bandbreitenhungrige Protokolle in der Präsenz von Flaschenhälsen, während sie alle anderen deklassieren, sobald das Netz genügend Kapazität aufweist. Eine ähnliches Bild zeigt sich auch zwischen den beiden Polen hohe Vermaschung und spärlicher Verbindungen. Als Ergebnis dieses breit angelegten Vergleichs unter realitätsnahen Bedingungen können wir nun nicht nur die vorteilhaften Aspekte eines Protokolls zeigen und beschreiben, sondern auch die Notwendigkeit der Beziehung zwischen mobilen opportunistischen Protokollen und QoS-Routing aufzeigen. Diese Erkenntnis führt zu einer leichtgewichtige Metrik, welche die verfügbare Bandbreite und eine dynamische Pfadqualität verschmilzt. Diese Studie beleuchtet auch den neuen Routingalgorithmus *Nile*[3], welcher sich dem unterliegenden Netz anpasst und über ein breites

[3]Benannt nach dem Fluss Nil bei welchem die Flut durch zusätzliche Baumassnahmen kontrolliert wird.

Feld an Parametern gute Resultate liefert, in dem es sich der Entwicklung der verfügbaren Bandbreite anpasst und mehrere Pfade nutzt, von denen Unabhängigkeit erwartet werden kann.
Flooding gilt als Benchmark für opportunistische Netze, da es die besten Resultate erzielt, wenn sein massiver Overhead ignoriert wird. Wir beschreiben und erläutern ein einfaches Netzwerkprotokoll für die Simulation, welches diese Nachteile von Flooding zu vermeiden versucht und deshalb in Näherung einer oberen Schranke für die Effizienz darstellt. Es stellt damit einen Kandidaten für eine allgemeine Vergleichsbasis dar. Diese Methode, EPO[4], vermeidet die Nachteile von Flooding auch bei knapper Bandbreite und vermeidet Überlastung des Netzes. Dadurch können zusätzliche Einsichten in die Netzstruktur gewonnen werden.
Routingprotokolle in opportunistischen Netzen sind auf die Vorgeschichte dieser Netze angewiesen um Voraussagen und damit Routenentscheidungen treffen zu können. Um die grundsätzliche Qualität dieser Vorgeschichte auf der einen und den Einfluss von Aggregation auf die damit zu erzielende Routingperformance aufzuzeigen, haben wir den Durchsatz mittels eines modifizierten Maxflow-Ansatzes analysiert und verglichen. Neben den Einblicken in die Auswirkungen der Eigenschaften dieser Vorgeschichte zeigte sich auch, dass es bei Routingfehlern unmöglich wird, einen fehlbaren Link klar zu identifizieren, der den Routenplan zunichte macht.

[4] Inspiriert durch die Verwendung von Erythropoietin als leistungsverbessernde Substanz.

Acknowledgments

During the five years I have been at University of Konstanz, I have been influenced by many people around me, who have shaped the way I think and taught me a lot about how to do research.

Among those, I want to first thank my supervisor Marcel Waldvogel for providing me with guidance, support, and the encouragement to try something new and different. It would not have been possible to complete this study without his faith in my work and his excellent advice on many issues concerning this work and beyond. I greatly appreciate the many hours of discussions we have had over the past five years.

I am grateful to Prof. Ulrik Brandes for providing me with friendly and valuable advices on several occasions.

I would also like to express my thanks to Disy research group. I have learned so much from all of you – from figuring out what research is, to choosing a research agenda, to learning how to present my work. your collaboration, queries and criticism have been tremendously valuable to me throughout this and other projects.

I would like to express my appreciation and gratitude to the following people who have either contributed towards or inspired the work presented in this thesis. I was fortunate to have creative ideas from Sebastian Kay Belle, which helped me to bring a few new avenues to my project. I also thank Michael Zinsmaier who helped us to implement them. The work presented in Chapter 10 could not be completed without their help.

My thanks to Anna Dowden-Williams, who improved this text with her reviews and comments. She also helped me reviewing a few published papers that were very important milestones in the course of this project.

And of course to all my friends who have given me the strength and support and with whom I had many thought provoking conversations.

My special thanks goes to my family, who has supported me throughout this project and during my time in Konstanz.

Contents

Chapter 1

Introduction

> "Motivation is what gets you started. Habit is what keeps you going."
> Jim Rohn

This dissertation addresses the problem of finding a practical routing solution for opportunistic networks. Opportunistic networks are mostly constituted by wireless devices that happen to be in each other's radio range and, thus, become a part of a multi-hop communication network. Routing in such an environment is a twofold challenge because delivering a message is more complicated than finding a path to destination. This study investigates the characteristics of opportunistic networks and presents an implementation of a protocol that scales to any kind of network, irrespective of the network size, network throughput, traffic volume, and device mobility. This chapter introduces the area of research by presenting the motivation behind our work. It also presents scenarios from our daily life that may benefit from our investigations. At the end of this chapter, we have discussed the contributions we have brought to the field of routing in opportunistic networks.

1.1 Motivation

Although, the problem of routing in opportunistic networks receives extensive attention, we have still not seen a robust and reliable solution. The reason being that challenges involved in opportunistic networks routing differ significantly from traditional wired networks. Not only we can design and plan the structure of wired networks, but we can also adapt their functionality, since we have real time information about the route changes in the network in case part of a networks fails.

Moreover, wired networks have dedicated nodes at central locations to carry out the routing operations. They are strategically positioned taking the capacity requirement of the applications into consideration. Additionally, the inherent nature of wired networks enable us to have near-real-time information about the traffic load. Whenever new requirements emerge, structural modifications are made to the network to adapt to the changes. The adaptation process assures that there is no need to change the routing mechanism and the network continues to operate.

In contrast, opportunistic networks (as the name suggests) cannot be designed or planned. They are implicitly created and evolved due to wireless devices that happen to be in each other's communication range. These wireless devices then behave as data mules as well as routers. They make routing decisions to bring the messages to their respective destinations based on the local knowledge that they have obtained earlier from the network. In our view, routing in opportunistic networks is composed of two steps:

1. to find a path to destination. As there are no dedicated routers, devices are responsible to share routing information with each other and, then based on this shared information, one or more paths are computed.

2. to ensure that the given paths are reliable enough to deliver the message. Due to unavailability of accurate traffic information, devices have to filter out those paths that do not have the residue capacity to transmit the message in question.

Existing opportunistic network routing methods mostly concentrate on finding the path to destination and neglect the second issue of delivering the message. As we will show in the upcoming chapters, due to either selection of favorable underlying networks or impractical traffic patterns, most of the existing methods promise impressive performance just by addressing the issue of path finding.

1.2 Applications

Certain terrestrial applications must deal with various forms of disruption as well as delays. However, not on the same scale as light-trip times across the solar system. Delays and disruption in these cases will much more likely be due to the operating system turning off battery-powered devices to conserve scarce power or mobile devices leaving each others radio ranges [FCG+06]. In both cases, protocols that implement a store-and-forward approach reminiscent of how email works, can

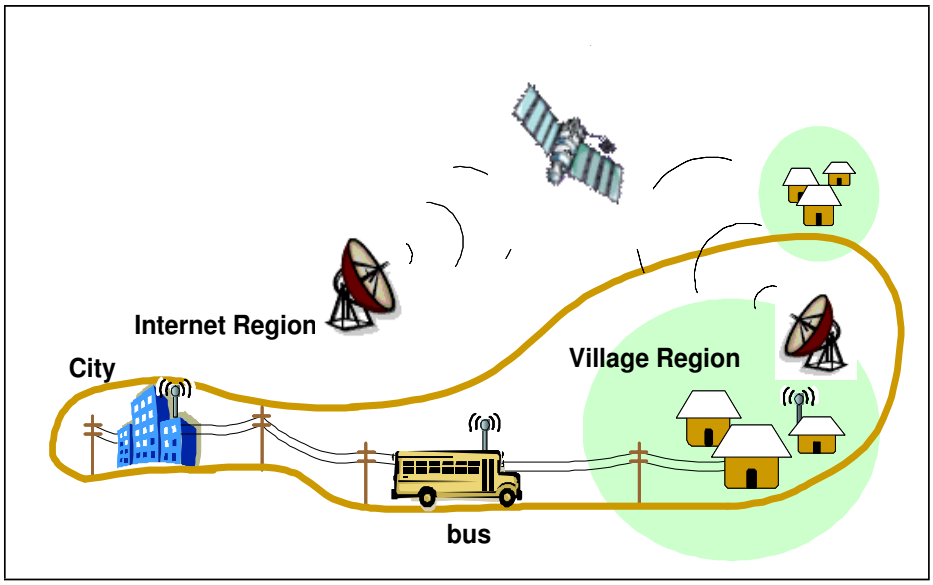


Figure 1.1: A DTN Node Mounted on bus delivering data around the villages [JOW+02]

offer significant advantages. Therefore, the evolution process started with the concept of a network that can tolerate delays, hence the first term *Delay Tolerant Network* (DTN) came into existence.

1.2.1 Remote/Indigenous Communities

Communication between remote villages and metropolitan areas suffer from lack of infrastructure such as wired network access. Other means of networking (GSM, satellite...) are either unavailable, intermittent, or too expensive to be viable [PFH04]. They also require solving the problem of power and local data distribution. Through the use of community gateways, and mobile relays, it is possible to use the existing infrastructure of vehicles and human mobility to provide connectivity between remote and more populated areas. There are some pilot projects using the opportunistic networks for offering e-government and other social services like e-mail in polar scandinavian regions [LDLE08] and in third world countries [PFH04].

1.2.2 Sensor Networking

In sensor networks, a large number of sensors is often deployed to achieve a high degree of redundancy. However, in certain sensor network scenarios it might be

possible or desirable to deploy only a smaller number of sensors in order to only have a limited subset of the sensors active at any given time to conserve energy. Depending on the application area, these sensors may be mobile and, thus, may not be connected continuously. Despite such conservation strategies, it is required that data from all sensors be collected, even though the network may not be connected at all times. Examples of such scenarios include a collection of oceanographic data from tags attached to seals or whales in the ocean, or from zebras in the African Savanna [JOW+02] where either the sensors themselves or the mobile mules that move among sensors collect data.

1.2.3 Disaster Management

Opportunistic networks plays a vital role in the case of disaster recovery management. Just like computers have become an integral part of our daily life, they have become indispensable in case of natural catastrophes and man-made disasters during which communication infrastructures have been destroyed. Furthermore, computing services are essential to plan and manage relief and rescue strategies. However, their decision making power is hampered by the fact that networking facilities are often no longer available due to the inherit nature of their jobs. Relief and rescue organizations rely heavily on satellite communication. Although, this medium comes at a relatively low cost, it can at most be used for a phone call. Moreover, it lacks multimedia facilities like video or imagery. Opportunistic networks can be deployed with comparative ease and information traveling through opportunistic network and thereby improves the efficiency of the rescue work tremendously.

1.2.4 Deep Space Communication

Communicating from earth to space craft or future bases further out in the solar system, pose similar challenges as opportunistic networks. For instance, due to the long distance and the propagation speed of radio waves, the round trip time from Earth to Mars takes between 8 and 40 minutes, depending on the orbital positions of the planets. Furthermore, when communicating with a satellite, orbiting Mars, there will also be frequent periods of disconnection when the satellite is behind the planet, which then effectively blocks all radio waves. Thus, all communication must be carefully scheduled to avoid transmitting large amounts of data, since one (very long) round trip time later it might be discovered that the receiver was in the radio shadow behind the planet.

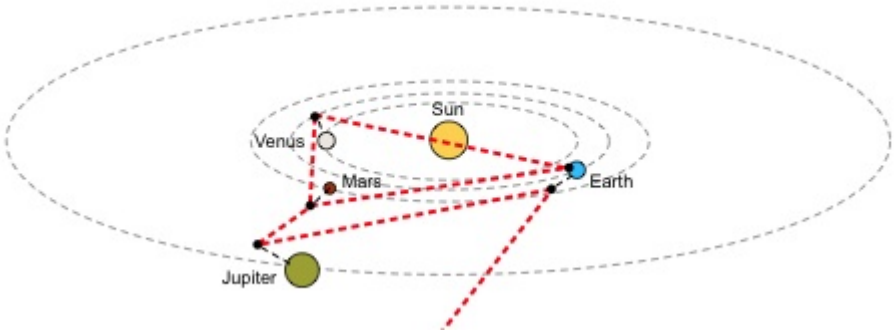


Figure 1.2: Interplanetary communication proposed with the help of delay tolerant communication with satellites as middle hops

1.3 Routing Challenges

The current trends in the field show that most of the routing protocols are tested on networks that are dense, predictable or aided by a communication infrastructure, such as publicly accessible access point and EDGE (Enhanced Date Rates for GSM Evolution). We can find several examples in the literature where opportunistic networks have been deployed in real life. In a few cases, Internet connected mules are introduced in the network to improve the network performance [KO07], while in other cases, public transportation is equipped with wireless devices. Therefore, the routing protocol implicitly gains the advantage of the predictable schedule of a public transportation system to deliver the messages [BGJL06].We can also find examples where simulations are performed mostly on artificially generated networks in which the movement of the nodes is defined by a prescribed velocity in a random direction in a designated area. In such setups, the movement of the devices is bounded by a predefined area, which the devices may never leave. The devices may become inactive, move with variable speed and interact randomly with other devices while remaining inside the area. Thus, the end result is a dense network with a high probability that devices may come into each other's range. We can also find a few examples that rely on a device based "home zone". Here the assumption is that a device spends most of the time either in an euclidean space or among a designated group of people. Device behavior, first of all is not that predictable and additionally a device may become overburdened if several external home zones end up using this device to route all traffic to its home zone.

Our observation is that the opportunistic nature of these networks brings so much unpredictability to path structure that the shortest path existing at one point in time may either not turn out to be the shortest one or in the worse case scenario, this path may even fail to exist in future. As the devices carry the message

until the contact with the next hop is established, the buffering delays tend to be considerably longer than inter-hop transmission times. The buffering delays are dependent on the movement of the device carrier and therefore the delays incurred by the message to its destination may range from a few minutes to several days. Another reason for the delays that cannot be ignored, is the congestion due to the traffic volume itself. In a cluster scenario, this can have very serious repercussions because a wireless device, overburdened with transmission will not only suffer from rapid loss of battery life, but also from lack of transmission opportunities to other devices, thereby causing extra delays for messages to reach their respective destinations.

The above given arguments show that it is not only necessary to find a path that reaches the destination but also to consider quality of the path, i.e. required residue capacity, must be ensured. The lack of capacity of any path may either be an original characteristic of the path or arise from the current traffic trends. It is imperative to mention here that the traffic measures that devices share among each other, also suffer delays leaving the rest of the network with inaccurate information to make routing decisions. We believe that in such an unpredictable environment it is not only necessary to use multi-path routing, i.e. propagating multiple replicas of a message along several paths simultaneously, but it is also necessary to keep the traffic volume in check. The replicas, on one hand increase the probability of the message delivery to the destination, while on the other hand they create congestion in the network, thereby reducing the delivery probability of other messages being propagated at the same time through common paths.

1.3.1 Our Approach

The reliability question is not new to the network community and there is a solution available to control network traffic. Quality of Service (QoS) methods are usually used to ensure real-time multimedia traffic to the destination because of the obvious needs of streaming applications. Traditionally, QoS is employed to provide better service to selected network traffic over various technologies. Here it is imperative to mention that ensuring QoS in wired networks is relatively easy, traffic capacity and demand can be estimated to an accurate level because the capabilities and specification of routers are known and the traffic generated by the applications can be estimated accurately. We argue that we must couple QoS routing practices similar to those of wired networks with the routing problem in opportunistic networks because storage and transmission capabilities of wireless devices in opportunistic networks can severely degrade if QoS aspects are fully ignored. In opportunistic networks, not only individual devices have highly variable

capacities but also the traffic is less predictable. It is well known that wireless communication suffers highly from traffic congestion and in case of a bottleneck, path recalculation can be a resource expensive process. As already stated, a traffic congestion in opportunistic networks, not only creates problems for those messages that are directly involved but it also reduces the delivery probability of those messages that are sharing the same path.

We can identify a strong desire to have path capacity metrics that can predict the path traffic as accurate as possible. For better prediction based routing decisions, more information must be gathered about network that may adversely effect the delivery process. It is therefore, necessary for opportunistic networks to have path metrics that are accurate as well as concise. It is easy to understand that in order to have accuracy and conciseness simultaneity is a contradictory goal. Thus, we have to find a sound compromise between the two. One question in the choice of path metric is, how to couple speed with reliability. As already discussed, we need the shortest, most reliable path.

1.4 Contribution

Opportunistic networks are still going through their teething period. If interested in investigating opportunistic networks, one has to start with a particular problem of personal interest and then builds upon that basis. As already stated earlier, there are several offshoots of opportunistic network (we will be discussing the evolution process in the coming section), including vehicular networks and delay/disruption tolerant network. It is difficult to find one precise infrastructure definition that suits every variant of opportunistic network. A routing protocol designed under the assumptions of delay tolerant network may assume considerably different underlying condition as compared to vehicular network routing protocol. We believe that all of these platforms do have a common premise and we should not limit any one of them for a specific type of traffic or behavior.

1. Given these issues, we have brought several techniques to a common testing ground and investigated the scenarios in which one technique may work better than the others. We have explained the reasons behind it and during this process, we have identified the issues of common simulation practices that bring undue advantage to proposed solutions.

2. We have implemented an opportunistic network simulator that is designed to work with real life movement traces. We have also simulated several routing protocols that show their versatility. Anyone interested in simulating a new

protocol can do it by using our simulator after becoming acquainted with its process.

3. Whenever a DTN protocol is analyzed, it is benchmarked against flooding, however, a comparison to other solutions is not really available. In our opinion, the variance in the testing environment and underlying networks is so high that it is difficult for researchers to test two different techniques on one common ground. To our knowledge, we are first to bring different opportunistic routing protocols under one testing environment that is based on real life movement data.

4. We have analyzed the behavior of several protocols in depth and provided a thorough investigation of their strong and weak points.

5. We have identified the differences one must consider between wireless ad hoc networks and opportunistic networks. The primary difference being that delivering a message to a destination is a far more complicated than just computing the path.

6. To our knowledge, we are the first to propose inclusion of QoS like metrics for opportunistic network routing. Employing QoS methods to opportunistic network is different from employing QoS in ad hoc wireless networks. Other then the challenge of gathering QoS information from a network, the delay involved makes the available information outdated and unreliable. This in turn adds on additional complexity for the routing decisions in opportunistic networks.

7. Based on the strength identified, we have gathered the requirements of a novel protocol that is flexible enough so that it is capable of working successfully in any environment. We have also tested variants of this protocol for uni-cast as well as multi-cast routing.

8. We have also identified the issues that must be considered to obtain reliable simulation results. We have described new definitions that help us to obtain the near optimum baseline for simulations.

1.5 Publications

The following texts were published as a result of this research project:

1. Islam, M.A.; Waldvogel, M.; Nuntifix-Modeling of Delay Tolerant Networks : A Technical Report, University of Konstanz, Germany, March 2008

2. Islam, M.A.; Waldvogel, M.; Reality-Check for DTN Routing Algorithms :Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems Workshops, pages 204–209, 2008

3. Belle, S.K.; Islam, M.A.; Waldvogel, M.; I seek for knowledge: Exploiting social properties in Mobile Ad Hoc Networks, pages 1–5, Wireless Days, 2008 '08 1st IFIP

4. Islam, M.A.; Waldvogel, M.; Optimizing Message Delivery in Mobile-Opportunistic Networks, To appear in BCFIC 2011

5. Islam, M.A.; Waldvogel, M.; Questioning Flooding as a Routing Benchmark in Opportunistic Networks, To appear in BCFIC 2011

6. Islam, M.A.; Waldvogel, M.; Analyzing prediction quality of history in opportunistic networks, submitted to the Fifth IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications.

1.6 Outline

Chapter 2 describes the problems related to mobile communication in general. Additioanlly, we present the evolutionary process of opportunistic networks and its offshoots in this chapter.

Chapter 3 introduces the routing problem in wireless ad hoc networks. It establishes a definition and the aim of an ad hoc routing protocol and discusses the related issues. It also presents a self made classification of existing routing protocols and identifies their pros and cons.

In Chapter 4 we describe the relationship between an opportunistic network and a routing protocol. Advantages or disadvantages that may be caused by the characteristics of the networks are identified, and discussion is presented to how the research community dealing with these aspects. Following we describe the reason behind our choice of traces and discuss how we have cleaned up the selected traces for our routing simulations. A brief synthesis of selected traces is also presented in this chapter.

We have discussed the detailed simulation methodology in Chapter 5. We have presented the process of transformation of all the traces to a common format for our simulator. We have compared all the network traces with respect to those issues that play a vital role in message routing. Moreover, we have presented

descriptions of several protocols that we have selected for our study. At the end of this chapter, we have identified evaluation criteria that we have used to analyze the performance of all the protocols.

Chapter 6 identifies the strengths of all the protocols identified in Chapter 5. We then present, in detail, the technical specifications of our novel routing protocol. Following, we show how we have incorporated the strong points we have identified to take advantage of the network structure in all the cases.

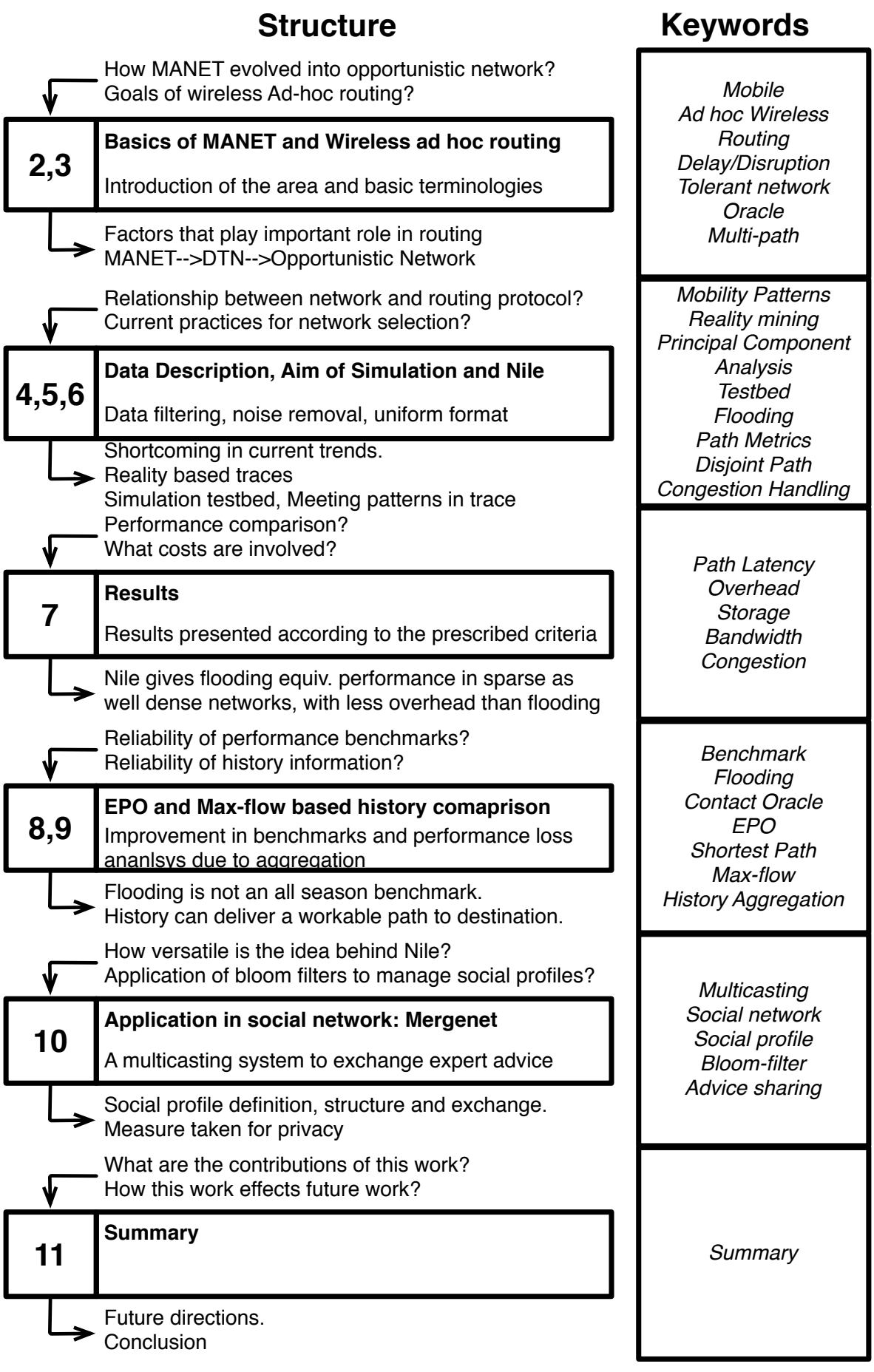
Chapter 7 presents the performance results of all the studies that we have performed. It analyzes the behavior of all the routing protocols from several different angles and explains the reasons behind those behaviors. It provides a comprehensive summary of the aspects that should be considered during performance analysis of any opportunistic network routing protocol.

Chapter 8 builds upon the results presented in Chapter 7 by identifying the shortcomings of flooding. It presents the reason behind the selection of flooding as a benchmark and then challenges the logic behind those reasons. This chapter also sheds light on those aspects that must be considered to establish a universal and scalable benchmark.

Chapter 9 discusses the effects of aggregation of information on routing. We will show that opportunistic networks suffer from lack of information on which suitable routing decision can be made. If this information is available, a significant question is accuracy. We created several hypothetical scenarios to identify the effects of routing decisions based on potentially imprecise and inaccurate information on the performance.

Chapter 10 introduces and mobile social networking architecture that is motivated by those missing aspects in the current online social networking facilities that can help users to engage in near real-time with their social environment. We have also analyzed the performance of a multicasting routing protocol, which we have tested in the selected traces..

Chapter 11 discusses briefly those issues that we have not been able to explore in depth and that will be followed up in the future. It highlights the other issues that the research community has currently been exploring and it presents the lessons we have learnt from this project.


Structure
Keywords
How MANET evolved into opportunistic network?
Goals of wireless Ad-hoc routing?
2,3
Basics of MANET and Wireless ad hoc routing
Introduction of the area and basic terminologies
Factors that play important role in routing
MANET-->DTN-->Opportunistic Network
Mobile
Ad hoc Wireless
Routing
Delay/Disruption
Tolerant network
Oracle
Multi-path
Relationship between network and routing protocol?
Current practices for network selection?
4,5,6
Data Description, Aim of Simulation and Nile
Data filtering, noise removal, uniform format
Shortcoming in current trends.
Reality based traces
Simulation testbed, Meeting patterns in trace
Mobility Patterns
Reality mining
Principal Component
Analysis
Testbed
Flooding
Path Metrics
Disjoint Path
Congestion Handling
Performance comparison?
What costs are involved?
7
Results
Results presented according to the prescribed criteria
Nile gives flooding equiv. performance in sparse as
well dense networks, with less overhead than flooding
Path Latency
Overhead
Storage
Bandwidth
Congestion
Reliability of performance benchmarks?
Reliability of history information?
8,9
EPO and Max-flow based history comaprison
Improvement in benchmarks and performance loss
ananlsys due to aggregation
Flooding is not an all season benchmark.
History can deliver a workable path to destination.
Benchmark
Flooding
Contact Oracle
EPO
Shortest Path
Max-flow
History Aggregation
How versatile is the idea behind Nile?
Application of bloom filters to manage social profiles?
10
Application in social network: Mergenet
A multicasting system to exchange expert advice
Social profile definition, structure and exchange.
Measure taken for privacy
Multicasting
Social network
Social profile
Bloom-filter
Advice sharing
What are the contributions of this work?
How this work effects future work?
11
Summary
Future directions.
Conclusion
Summary

Chapter 2

Evolution of opportunistic networks

> "A good short-story writer has an instinct for sketching in just enough background to ground the specific story." Lynn Abbey

This chapter presents an overview of the recent activities in the field of mobile communication. The applications of ad hoc networks is established by the fact that mobile communication is revolutionizing human life, and the industry is moving forward to meet increasing demands and expectations of the masses, establishes the application of ad hoc networks. In the last part, we establish a connection between DTN in space and opportunistic network on earth, redefining the concept of ad hoc networks and bringing it a step closer to the P2P paradigm. This chapter presents the evolution process of *Delay/Disruption Tolerant Networks* into *Opportunistic Networks* and presents a brief summary of the work that is being carried out by the research community.

2.1 Current trends in mobile communication

During the past decade, communication technology has revolutionized the way the world used to communicate. The spread of such technology is bringing more and more people into this communication web, which is becoming increasingly global in nature. The internet, in particular, is global in design as it jumps territorial boundaries. It is becoming an e-world that includes important sectors, such as e-commerce, e-friendship, e-government and e-mail [Sch05]. The number of internet

users has increased from nearly half a billion to slightly over one billion between 2001 and 2005. In 2004, 38% of all internet subscribers worldwide had access to broadband [UNC06]. The recent trend is going a leap further than just sitting in front of a PC that is connected to a network. Today, we can also use mobile phones, portable laptops and other mobile communication devices (that paved the way to be the next logical step in this technological revolution, connecting people anytime, anywhere), to connect inanimate objects in a communication network. Due to the continuing advancement in nano technology and with the advent of solid state storage devices, the size of mobile and handheld devices, including mobile phones, palm tops and other USB devices is decreasing, while their computational and storage capabilities are remarkably increasing[1]. All of these factors are fueling the explosive growth of the mobile computing equipment market, known to us today. The motivation behind such a surge is the ever increasing demand of users to have small, efficient, and multi-functionals devices capable of professional as well as personal entertainment. These devices are now playing a significant role in our lives. Moreover, with decreasing costs, more and more people across the planet are able to stay in touch by phone. Mobile phone subscribers more than doubled from just under a billion in 2001 to 2.1 billion in 2005. If we include the numbers of laptop and PDA users, it is predicted that the number of mobile and Internet terminals will grow by yet another 20 – 50%.

Irrespective of the economical condition of the users, new requirements and software products are appearing that make use of such ubiquitous technology in all walks of life, and these demands further drastically increase. Users can now rely on their mobile devices to check their email and browse the Internet. Travelers with portable devices can surf the internet from airports, railway stations, cafe and other public locations Tourist can use GPS terminals installed inside mobile phones to view driving maps and locate touristic attractions, and files and other information can be exchanged by connecting portable computers via wireless transmission. This trend is motivating the manufacturing industry to embed handheld devices with all kinds of luxuries like cameras, multimedia modules and speech recognition add ons, thus, increasing the amount of potential information to be exchanged among the users. For this reason, it is imperative to continue decorating wireless networks with both, hardware (efficient long lasting batteries, error free communication tools etc.) and software (robust routing protocols). These tools are not only helping people to access the information they are interested in, but additionally they are restructuring their social interaction. The social side of online communities is growing in importance where traditional communities in

[1]Currently 128GB storage are available on a (large) fingernail sized chip, http://www.heise.de/newsticker/meldung/SDXC-Speicherkarte-mit-128-GByte-1162681.html

the internet are built around interests, e.g. people form friendships with network users who are interested in the same multimedia content. Moreover, the ability to carry one's entire music collection enables listeners to introduce new tracks to their friends. in turn prompts users to compile special collections of tunes, or playlists.

Given these devlopments there are no conflicting opinions about the extent to which wireless networks have made their impact in every walk of human life, irrespective of personal or business communication, exchange of information between two individuals or dispersion of information to public, small scale businesses or huge metropolitan operations. Today's expensive wireless infrastructure depends on centrally deployed hub and spoke networks, while mobile ad hoc networks consist of devices that are autonomously self-organizing in networks. Although, infrastructure based networks provide a great way for mobile devices, setting up the network infrastructure is time consuming and the cost associated with installation can be quite high. Furthermore, there are situations where user required infrastructure is not available, cannot be installed or cannot be installed in time in a given geographic area. Providing the needed connectivity and network services in these situations requires a mobile ad hoc network. Ad hoc networks do not require any assistance from any kind of infrastructure allowing seamless communication at low cost, in a self-organized fashion, and with easy deployment. The large degree of freedom and self organizing capabilities make mobile ad hoc networks completely different from any other networking solution.

2.2 Mobile ad hoc network *MANET*

In general mobile ad hoc networks are formed dynamically by an autonomous system of mobile nodes that are connected via wireless links without using centralized administration or pre-existing network infrastructure such as base stations. The nodes are free to move randomly and organize themselves arbitrarily forcing the network topology to change unpredictably and rapidly. Such a network may operate in standalone fashion, or may be connected to a larger Internet. In general, routes between nodes in ad hoc networks may include multiple hops therefore it is reasonable to call such networks "multi-hop wireless ad hoc networks" [BCGS04]. Although, MANETs enable users to create their own networks that can be deployed easily and cheaply. The price of all those features is paid in terms of complex technical solutions, which are needed across all layers. Given all the reasons, mobile ad hoc networking is one of the more innovative and challenging areas of wireless networking, and is promising to become increasingly important in everybody's life.

Ad hoc networks are the key step in the evolution of wireless networks as they

inherit traditional problems of wireless and mobile communication such as bandwidth optimization, power control, and transmission quality. In addition, the multi-hop nature and the lack of a fixed infrastructure poses new research problems, such as network configuration, device discovery and topology maintenance as well as ad hoc addressing and self-routing. MANETs inherit common characteristics from wireless networks in general and add complications specifically to ad hoc networking, which are listed below:

- **Wireless:** Nodes communicate wirelessly and share same media (radio, infrared etc.)
- **Ad hoc based:** A mobile ad hoc network is temporary network formed dynamically in an arbitrary manner by collection of nodes when need arises.
- **Autonomous and infrastructure-less:** MANET does not depend on any established infrastructure or centralized administration. Each device operates in distributed peer to peer mode, acts as a independent router and generates independent data.
- **Multi-hop routing:** No dedicated routers are necessary; every node acts as a router and forwards each other message to enable information sharing between mobile hosts.
- **Mobility:** Each node is free to move about while communicating with other nodes. The topology of such an ad hoc network is dynamic in nature due to the constant movement of the participating nodes, causing the intercommunication patterns among the nodes to change continuously.

The infrastructure-less nature of MANETs makes them very attractive for tactical network related applications to improve battlefield communications and survivability. Nodes in such networks have the inherent requirement to create and join networks on the fly, i.e. any time and any where for virtually any application. The dynamics nature of military, rescue and emergency operations makes it impossible to rely on fixed pre-placed communication infrastructure in the operation area. As successful as these networks have been, they still cannot reach everywhere, and for some applications their cost is prohibitive. The reason for these limitations is that the current networking technology relies on a set of fundamental assumptions that are not true in all environments. The first and most important assumption is that an end-to-end connection exists from the source to the destination, possibly via multiple intermediaries. This assumption can be easily violated due to mobility, power saving, or unreliable networks, e.g., if a wireless device is out of range of the

network, it cannot use any application that requires network communication. In ad hoc wireless networks, where simultaneous links in the network are not possible, the growing number of studies are exploring techniques for moving network traffic over asynchronous paths. This intermittency gives birth to Disruption/Delay tolerant network(DTN) [Fal, BBL05].

.

2.3 Delay tolerant network *DTN*

Delay tolerant networking (DTN) is an attempt to extend the reach of MANETs such that they promise communication between the nodes even when they are not connected to the network. It can be argued that delays or disruption do not require any new networking technology and the standard Internet protocols are sufficiently capable. However, in certain networking scenarios, important Internet protocols are just not usable, and it is for these cases that a number of research groups are developing the delay and disruption tolerant networking approach. In such situations, the TCP often does not work, although, it is universal to most of the applications we use every day such as email and the Web. Since TCP requires the sender and receiver to be connected to the network at the same time, and this condition cannot be guaranteed in DTN scenarios, we do have a compelling need for new protocols.

2.3.1 Evolution of DTN through interplanetary communication

Much of the DTN work described here has its roots in a NASA research project developing an interplanetary Internet or interplanetary network (IPN). The basic idea is to try to make data communications between Earth and (very) remote spacecraft seem almost as easy as that between two people on different sides of the world. However, before a network node can send any application data using TCP, a three-way handshake is required that consumes 1.5 round-trip times (RTTs). There is also a generic, two-minute timeout implemented in most TCP stacks. Combining these facts, we can see that once a spacecraft is more than a minute away (in terms of light-trip time), every attempt to establish a TCP connection will fail, and no application data will ever be transmitted. In the case of Mars, for example, at its closest approach to Earth, the RTT is roughly eight minutes, with

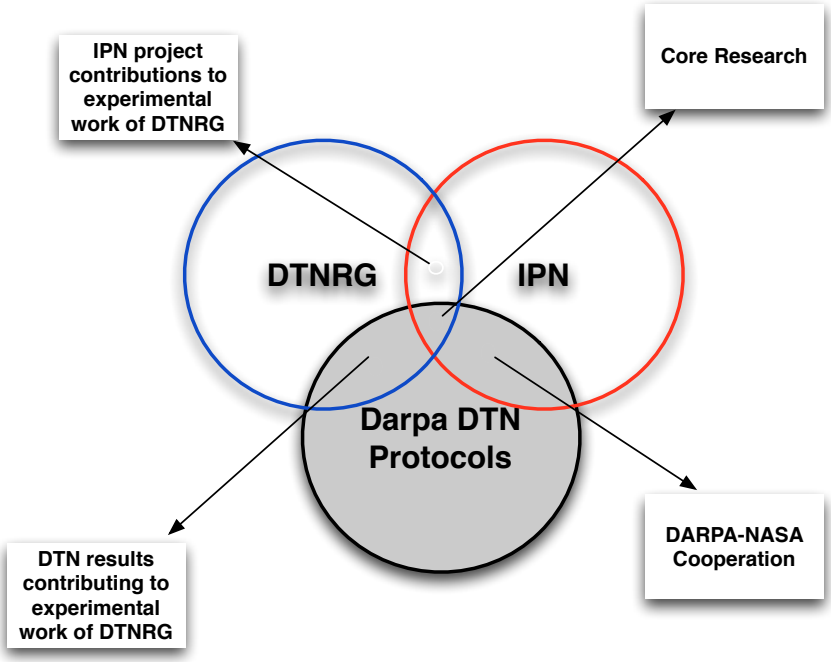


Figure 2.1: Delay tolerant network (DTN) diagram. Several organizations, including the Delay-Tolerant Networking Research Group (DTNRG), the interplanetary networking (IPN) group, and DARPA are trying to solve DTN and disruption-tolerant networking issues [FCG+06].

a worst-case RTT of approximately 40 minutes. Thus, normal TCP cannot work at all for Earth-to-Mars communications. There are also other issues involved that must be overcome, for example, the radio antenna is frequently on the wrong side of the planet. Nevertheless, at least in terms of networking, we can make good progress compared to how spacecraft data communications currently occur, which must essentially be scheduled manually on a mission-by-mission basis.

This notion of TCP being infeasible for delay-prone communication has been promoted by NASA engineers at Jet Propulsion Laboratory (JPL), who started working on IPN in 1998. Since then a couple of fairly substantial protocol development groups have achieved several noteworthy milestones. These working groups evolved into the IPN special interest group (IPNSIG), which is responsible for the progress toward developing protocol for such an architecture. At the same time, other researchers were investigating how IPN concepts might apply to terrestrial applications, particularly sensor networks, which turn out to have a lot in common with a putative IPN. Consequently, the Internet Research Task Force (IRTF) created a new research group to examine the more general area of DTN, that group is called the DTNRG, and it is currently the main open venue for work on the DTN

architecture and protocols.

The DTNRG is developing two main protocols, the Bundle Protocol and the Licklider Transmission Protocol (LTP).

- **The bundle protocol:** The protocol packages a unit of application data along with any required control information into a "bundle" and then forward this bundle along a route consisting of several intermediate devices that can each store it for significant periods. Thus, the bundle protocol is an overlay network store-and-forward protocol. A DTN node is an entity that runs an instance of the bundle protocol and can thus, in principle, send and receive bundles. However, some exceptional nodes can only transmit (such as a simple sensor), and, more commonly, some nodes might not be able to both, transmit and receive simultaneously.

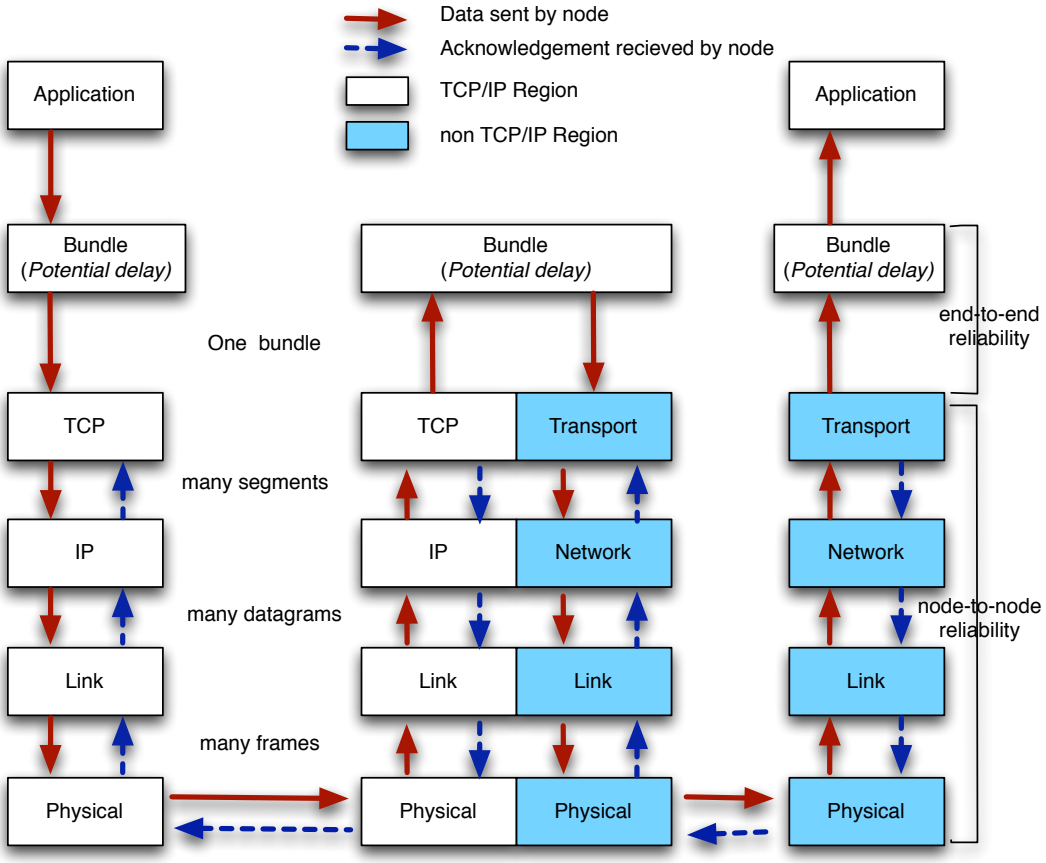


Figure 2.2: Modified structure of TCP/IP stack with Bundle Layer

- **Licklider transmission protocol.** LTP tackles delay tolerance and disconnection in a point-to-point environment with emphasis on operation over single, but typically very long-delay-links. Such links can suffer from long light-trip times and occultations[2]. From this description, it might seem that LTP is useful only for space communications, however, it can also be useful with terrestrial applications for which disruption is very likely. Applications dealing with disruptive environments can either be conventionally structured so that the application handles the expected errors, or, using a protocol such as LTP, the application can essentially be isolated from all of this complexity by having a communications daemon that handles all disruptive events, such as retransmissions required after a host shuts itself down.

To confuse matters more, the US Department of Defense, under DARPA, issued a call for proposals in early 2004 for what it called "Disruption-Tolerant Networking" (also called DTN), which is yet another generalization of the same concept. The difference is that up to the DARPA call, the main focus of DTN work was on high-delay cases such as IPN activities or sparse sensor networking (in which sensor readings are not needed in real time). However, other types of disruption can occur, such as radio shadowing or frequent passage in and out of base station range, a fact that the phrase "delay tolerance" does not properly reflect. Whether the D in DTN will come to mean "disruption" or continue to mean "delay" is not yet clear, but, in any case, the same architecture and protocols can hopefully serve in both contexts.

2.3.2 Routing in DTNs

We need on one hand a protocol that can easily be implemented on a hand held device, i.e. that does not challenges the computational power of a mobile device, on the other hand, we need a protocol that is able to differentiate between fast/congestion-free and slow/bottleneck links. The DTN paradigm has been proposed to deal with message delivery in scenarios where network partition is a normal network dynamics rather than abnormality. In the DTN paradigm, messages are sent from the sender without knowing any path leading to the receiver. Instead, the sender delivers the message to other nodes that may have a chance to deliver the message to the receiver at a later time. These intermediate nodes store the messages in its storage space and forward the messages when it encounters other nodes.

This is a different paradigm compared to traditional path-based routing, i.e. in

[2] the passage of one celestial body in front of another, thus hiding the other from view

path-based routing paradigm messages are delivered only in space domain during a short time interval, while DTN messages are disseminated through node mobility and pair-wise encounters and message exchanges between nodes during a longer time frame. Time elapsed between two successive encounters can be very long since intermediate nodes store the messages for future transmissions. As stated in Section 3, the routing information cannot be ensured to be 100% accurate. Even if a path information is accurate, the message utilizing this information can only be probabilistically guaranteed to be delivered as that path may be experiencing congestion in the middle. Thus, message replication is typically used to enhance delivery probability. Due to the richness and apparent novelty of the DTN routing problem, this has been a very active area of research.

2.4 Opportunistic networks and P2P paradigm

Lilien, Kamal and Gupta, from the P2P point of view, have developed a similar paradigm as DTNs with the name Opportunistic Networks or *oppnets* [LKG06]. According to their approach, Each oppnet grows from a pre-designed seed oppnet, or simply a seed, which is a set of nodes employed simultaneously at the time of the initial oppnet deployment. It can have just a few (possibly powerful) nodes, but in the extreme event that a single node (seed) oppnet grows into an expanded oppnet by taking in foreign nodes that become its helpers in realizing the oppnet's goals.According to Lilien, et.al. [LKG06], a seed can be wireless and ad hoc, with nodes not carefully pre-positioned but, for example, thrown out of a plane or a car in the general disaster area.

Once the seed becomes operational, its first task is to detect a set of "foreign" entities, i.e., devices, clusters, networks, or other systems, which it deems useful. The detected entities are candidates for becoming helpers for the oppnet. The size of the expanded network and locations of all but a subset of its "seed" nodes cannot even be approximately predicted. Moreover, there is no notion of either fixed addresses or fixed connectivity and the network is primarily owned and managed by the peers contributing to the network. Oppnets can be powerful, autonomous, able to self-organize, adapt to changing environments, and self-healing when faced with component failures or malicious attacks.

Each such candidate helper (or simply candidate) has a potential to provide oppnet with communication, computing, sensing, or other capabilities or resources. It can search for systems in the disaster area using the range of Internet addresses, known as IP addresses, assigned to its own geographical area.

Given such a paradigm of oppnets, it can be argued that oppnets satisfy criteria of being a P2P system with no concept of a central server. Oppnets do differ from traditional ad hoc networks where the devices of a single network are all deployed together in ad hoc networks, with the size of the network and locations of its nodes pre-designed (either in a fully "deterministic" fashion, or with a certain degree of randomness, as is the case with ad hoc networks).

Chapter 3

Routing in ad hoc networks

> "A route of many roads leading from nowhere to nothing." Ambrose Bierce

This chapter introduces in depth, the routing challenges involved in wireless ad hoc networks. The issues that have been implicitly or explicitly identified by the research community, are discussed with our support or objections regarding specific aspects. We have highlighted also those routing aspects that are pivotal according to our findings, and we have discussed the basis of several decisions that appear in the later part of this work.

3.1 Functions of an ad hoc routing protocol

Routing protocols are effectively distributed database systems, which have the purpose to dynamically communicate information about all network paths and to select the best path to reach a destination network. Routing protocols propagate information about the topology of the network among the routers within the network. Each router in the network uses this distributed database to determine the best loop free path through the network to reach any given destination. The primary aim of an ad hoc network routing protocol is the correct and efficient route establishment between a pair of nodes to ensure that messages can be delivered reliably and in a timely manner. The highly dynamic nature of mobile ad hoc networks results in frequent changes and unpredictability in network topologies, adding difficulty and complexity to routing among the mobile nodes within a network. The dynamic nature poses another challenge of building the route con-

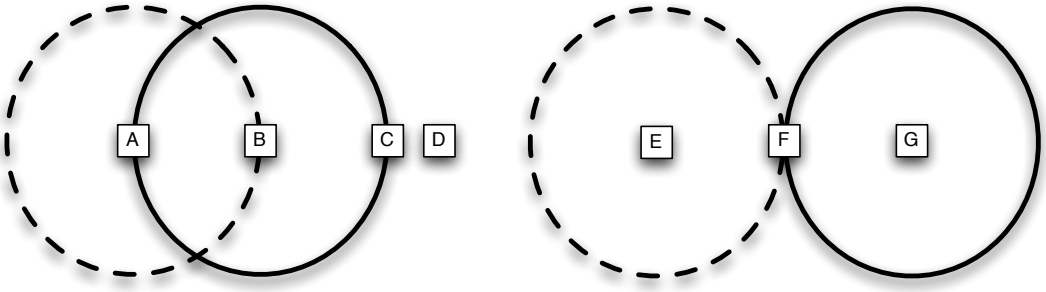


Figure 3.1: Terminal node problems

struction with minimum overhead and bandwidth consumption, so that repeated computations do not adversely effect the network resources.

When an end-to-end path fails to exist between a source and destination, both standard and MANET routing protocols no longer suffice. As a result, a number of proposals for disruption or delay tolerant (DTN) routing have recently surfaced. These schemes do not assume that an end-to-end network path exists necessarily, but rather that such paths(s) exist over time. The network can be partitioned either due to the movement or due to unavailability of peers. With the advent of VOIP and rapid spread of high bandwidth media, there is a shift towards synchronous technologies. However, asynchronous communication has its own advantage, e.g. it still works even if both the parties are not available simultaneously and it is less sensitive to link failure. These fundamental difference characteristic to DTN, motivate us that existing protocols for synchronous networks are not anymore valid and there is requirement to develop new routing protocols for moving devices with either pedestrians or vehicles.

The infrastructure-less nature of MANETs posses several challenges for network researchers including different design issues as compared to wired counterparts and distributed network management. The possibility that any node may move arbitrarily in any direction, constraints researchers to think in a topology free paradigm to solve the issues like route maintenance, handling frequent network partitions, and possible packet loss. Routing information is not always assumed to be 100% accurate, therefore a node may only have a probabilistic chance of successfully delivering a message. In addition, the physical layer radio interface at each node uses broadcast for transmitting the signal, which results in inherently

collision prone environment as shown in Fig 3.1, giving birth to hidden and exposed terminal problems. Moreover, wireless devices communicate with each other via bandwidth constrained, variable capacity, error prone and insecure wireless channels that give birth to more congestion related problems. The devices constituting such an ad hoc network may be heterogeneous having different frequency bands, and asymmetric links due to different radio capabilities. The devices may also have different energy constraints as most of them are operated by batteries. Therefore, researchers have to place bounds on time and resources needed by the protocol to make the routing decision, to keep processing power of the desired transmission in check.

In such scenarios, reliable network connectivity is obtained by sharing routing information and forwarding messages among multiple nodes. This feature combined with the lack of centralized management creates conditions such as bandwidth overload, nodes acting selfishly or network having broken links. Although, mobile networks are generally more vulnerable to information and physical security threats than fixed line networks and misbehaving nodes may severely effect the communication performance, such issues are out of scope from the work presented here. Other than the inadequacy of shared wireless broadcast channel, the network mainly relies on individual security solution due to its distributed nature. Many mobile ad hoc network applications involve large networks with ten of thousands of nodes, and therefore scalability is a very critical issue. The evolution of a medium size network with constrained resource into a large network is not a simple issue, posing challenges related to addressing, routing, location management, configuration management, interpretability, security, and so on. All these wireless and mobile ad hoc network characteristics, pose extra difficulties in the required QoS guarantee that typically include a wide set of metrics including throughput, jitter, packet loss, delay etc. These added challenges, coupled with critical importance of routing protocols in establishing communication among mobile nodes, make the routing area the most active research area within wireless ad hoc networks.

3.2 Issues involved in routing algorithms

We can classify routing algorithms according to several criteria as discussed in the following.

3.2.1 With or without replication

We may classify the routing schemes based on the fact that they either do use or do not use replication. Generally speaking, strategies that do not employ repli-

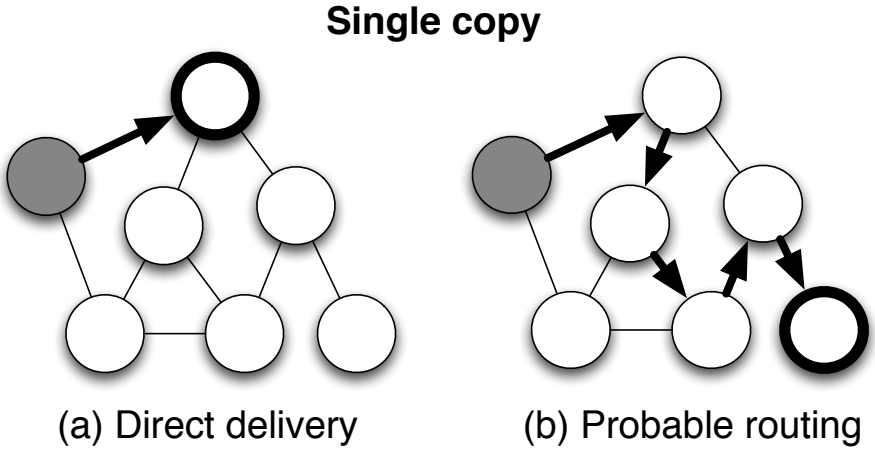


Figure 3.2: Routing Options - Single Copy Case

cation use computationally intensive procedures to determine the path for the message [BGJL06, JFP04, LDS03]. This path may be calculated at the source and then regularly updated on each hop. Each node has to decide in the given circumstances that whether the next encountered node can bring the message closer to destination or not. For routing decision, such algorithms usually incorporate network information available, such as topology structure, historical pattern of node behavior, path congestion indicators, etc. Fig. 3.2 shows examples for a single copy case where the source attempts to deliver the message directly to the destination and secondly, an ideal algorithm attempts to find the multi-hop path to the destination.

These techniques are also referred to single copy techniques [TSR08a] or smart strategies in literature. The simplest of them is *direct delivery* [SRJB03], where the source of the message delivers the message directly to the destination without intervention of any other node. As can be expected, this scheme is used as a maximum delay baseline for routing algorithms and requires no processing or *a priori* information compared to an algorithm that tries to find a path to the destination with the help of connectivity of the network nodes. One more practical variation of single copy includes MaxProp [BGJL06] that tries to maximize the utility function, which is based on encounter history of nodes throughout the network. Another simple method to take advantage of the history of past encounters is presented in order to make fewer and more "informed" forwarding decisions [JOW+02]. It is a general understanding that in a realistic scenario a single copy per message is usually not enough to deliver the message with high reliability and acceptable delay [TSR08b].

Mechanism that involve replication create multiple replicas to enhance the deliv-

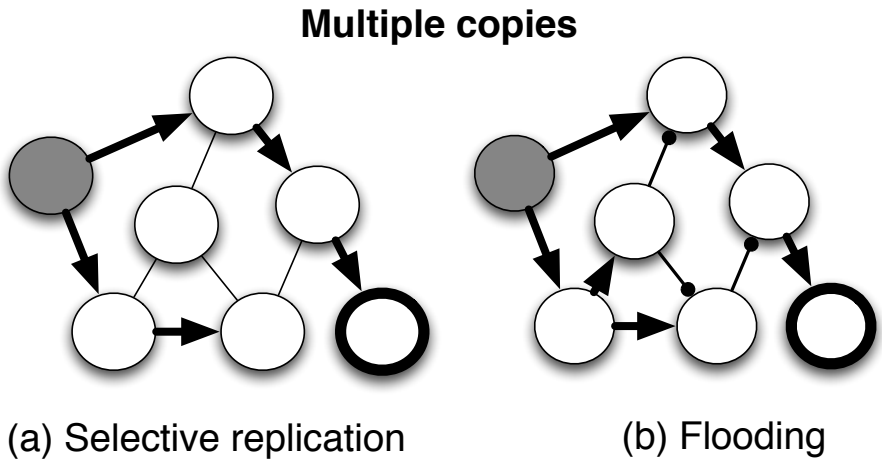


Figure 3.3: Routing Options - Multiple Copy Case

ery of message through the network [JFP04, JDPF05, BBL05, WJMF05], are also referred to as multiple copy methods [TSR08a]. The most prominent category of multiple copy (cf. Fig. 3.3) is epidemic routing or flooding that has been very actively used in networks whenever a speedy information delivery is a requirement. An epidemic protocol is described as a method that causes every node to exchange its messages with every other node in its range [VB00]. This way, every node in the network may end up having every message from every other node irrespective of whether this message was destined for it or not.

Flooding is believed to put a lot of strain on the local storage capacity of mobile nodes as well as on bandwidth of the network. However, almost every work in the category of message routing mentions flooding protocol as an upper performance benchmark for comparing results. Most of these works do not consider either flooding or flooding based algorithms for practical application due to buffer space and/or bandwidth consumption issues [LDS03, TSR08b, JLS07]. In contrast, no one has contested the fact that flooding has the capability of delivering the message in a timely manner. In realistic scenarios, compared to traditional flooding in the space domain, the DTN paradigm is more suitable for flooding as it increases the success rate of disseminating a message at a reduced transmission count. The improvement is due to the feature that intermediate nodes temporarily store copies of messages and keep looking for future transmission opportunities if the message is not deliverable at the current time. In other words, messages not only propagate in space domain by transmissions, but also propagate in the time domain together with nodal movements. Despite the fact that a large number of existing approaches are based on epidemic-routing or some other form of controlled flooding, they are claimed to be plagued by the shortcomings of flooding-based schemes [SPR05].

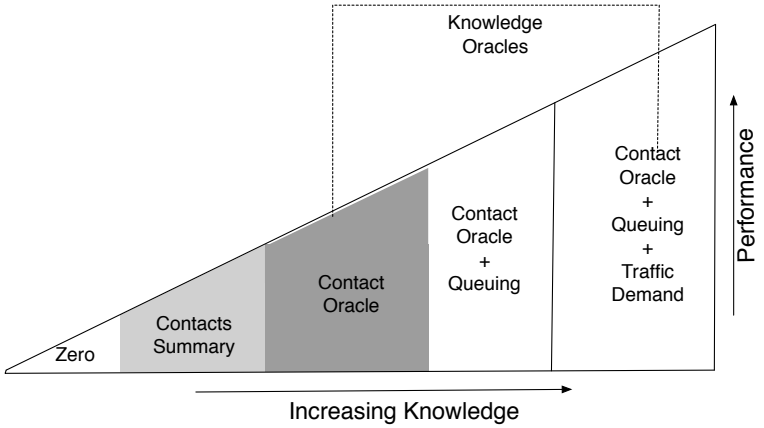


Figure 3.4: Conceptual performance vs. knowledge trade-off for different oracles. [JFP04]

Finally, there are some strategies that try to use the inherent advantages associated with erasure coding techniques [WJMF05, LTZG06]. They partition and encode the messages to reduce the load on bandwidth and then routing all of them over same or different paths. Erasure coding has the potential to improve the delivery ratio of the messages because not all fragments have to reach the destination.

3.2.2 Amount of knowledge

To make informed decisions, some routing strategies require more information about the network than others. In one extreme, a device can make decisions with zero network knowledge utilizing only currently connected devices. In other scenarios, devices have full knowledge about the upcoming events and they use static rules that are configured when the routing strategy is designed, and every device obeys the same rules. This leads to simple implementations that require minimal configuration and control messages, since all the rules are hard-coded ahead of time. The disadvantage is that such a strategy cannot adapt to scaling networks and changing conditions, so it may not make optimal decisions in the network other than it is designed for.

Provided that the information is accurate, routing strategies can make very efficient use of network resources by forwarding a message along the best path. Those sources that provide such an accurate information, which is difficult to gather in realistic scenarios about the network, are known as *oracles* in literature [JFP04]. These oracles are consulted to predict the aspect such as contact timing of devices, delivery probability to destination, traffic load on paths, etc.

As depicted in the Fig. 3.4, a zero knowledge protocol could be one that forwards the messages randomly or whoever receives it first. Contact summary gives insight into the past contact frequencies and more frequent contacts receive priority over the others. The most complicated oracle not only predicts the exact timings of contacts and local queue load of devices but also also the demand of devices. One can safely assume that the higher the accuracy, the less likely it is to actually construct such an oracle in the real world scenario.

In our opinion, the information that is necessary for making intelligent routing decisions and that can be constructed in real-world scenario lies between the two extremes of zero network knowledge and full knowledge of network contacts contacts and its traffic volume.

3.2.3 Reactive and proactive protocols

Another important issue of ad hoc routing is the timing of the path computation/update. The timing factor of path calculation, can play an important role to keep in check the volume of routing information. *Proactive* protocols compute routes ahead of time where mobile nodes update their routing tables by periodically exchanging routing information among themselves. *Reactive* protocols, on the other hand, compute a route to a destination when traffic for the destination is ready to be sent, thus trying to save network resources by not advertising routes that are never used.

In the case of proactive routing protocols, such as DSDV (destination sequence distance vector) [PB94], a large number of control messages are generated, due to periodic information exchanges about the update in the network path structure. Hence, proactive routing protocols are not considered suitable for frequently mobile ad hoc networks. To overcome the limitations in such scenarios, reactive routing protocols, such as DSR (dynamic source routing) [JM96] and AODV (ad hoc on-demand distance vector routing) [PR97] protocols have been proposed. Reactive routing protocol generally discovers the path only when it is required. A global search procedure is used by the route discovery mechanism in which a source node uses usually "flooding like" mechanisms to discover all the available paths to a destination. Once all paths have been discovered, a source node chooses a path, which is the shortest or the best according to its utility function. The drawback could be that the initial traffic for an unknown destination may be delayed as the route discovery process takes place on demand.

In the case of opportunistic networks, the choice is not simple because oppor-

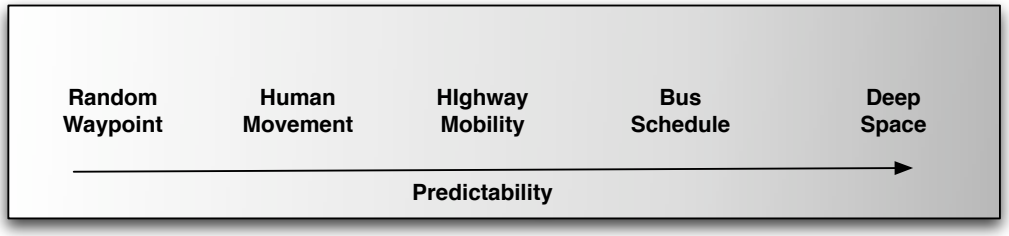


Figure 3.5: Prediction Spectrum

tunistic networks inherit characteristics from both, DTNs and MANETs. Dense opportunistic networks may behave like MANETs where reactive protocol will be suitable to operate. On the other hand, delays encountered in sparse portions of an opportunistic network may be so high that by the time a route is calculated on-demand, the message life has expired. Therefore, proactive methods must be incorporated to keep the paths ready for use, so that a message can be forwarded as soon as the request is generated.

3.2.4 Random and fixed schedule

Another way of looking at an opportunistic network is the predictability of the contact schedules. This factor is somewhat related to section 3.2.2 but it has a a few new aspects that are worth mentioning. As already discussed, oracles can predict the contact patterns of devices, however it is difficult to find such oracles in the real world. We can still find some example opportunistic networks in the real world where we can predict the movement of devices very accurately, e.g., in the deep space where we can very precisely forecast the contact timing of devices (supposedly mounted on space crafts, satellites and other moving objects). The reason being that we can foresee the movement patterns of all the heavenly objects and we are also aware of the transmission capabilities of the mounted devices, which makes it easier to decide the traffic volume that can be transmitted.

The other extreme as shown in Fig. 3.5, is that we have random waypoint model that describes totally unpredictable contact behavior by devices. Variants of random waypoint model are widely studied in the ad hoc network community because the models are relatively simple to work with. Some of the work in this area has also investigated networks with proactive mobility, where nodes actively move to try and help the network routing [ZA03]. As one may observe, several optimizations are possible in the *bus schedule* scenario, where contact schedules are more predictable.

Between the two extremes, we have those phenomena that have a medium accuracy

prediction associated with them, e.g., an opportunistic network constituted by the metropolitan bus system of a city. These buses have a schedule, but they may not be able to follow it to maximum accuracy. Due to traffic, equipment failures, or accidents, the actual arrival times can vary significantly. The next step, are those opportunistic networks that have an implicit schedule. Most human activities would fall into the category where no guarantee can be given about the timing of a person to be at work, but where for the most part their schedule is fairly regular.

3.2.5 Link state or distance vector

As already stated earlier in this section, the aim of a routing protocol is to dynamically communicate information about all network paths throughout the network. The terms *distance vector* and *link state* are used to group routing protocols into two broad categories based on whether the routing protocol selects the best routing path according to a distance metric (the distance) and an interface (the vector), or selects the best routing path by calculating the state of each link in a path and finding the path that has the lowest total metric to reach the destination. Each of these data distribution methods is generally tied to a specific method of finding the best path to any given destination within the network.

By distributing the state of the links attached to the routers

Each router floods (or advertises to all other routers in the network, whether directly adjacent or not), the state of each link to which it is attached. This information is used independently by each router within the routing domain to build a tree representing a topology of the network (called a shortest path tree). Routing protocols that distribute the state of attached links are called link state algorithms.

Link state: Link state protocols, rely on each router in the network to advertise the state of each of their links to every other router within the local routing domain. Link state protocols track the status and connection type of each link and produce a calculated metric based on these and other factors, including some set by the network administrator. Link state protocols know whether a link is up or down and how fast it is and calculates a cost to 'get there'. The result is a complete network topology map, called a shortest path tree, compiled by each router in the network. The link state information is flooded through the routing domain unchanged, just as the originating router advertises it. As a router receives an advertisement, it will store this information in a local database, typically referred to as the link state database, and pass the information on to each of its adjacent peers. This

information is not processed or manipulated in any way before it is passed on to the router's adjacent peers. As each router builds a complete database of the link state information as advertised by every other router within the network, it uses an algorithm, called the shortest path first algorithm, to build a tree with itself as the root of that tree. The shortest path to each reachable destination within the network is found by traversing the tree using Dijkstra's algorithm.

By distributing vectors

Each router in the network advertises the destinations it can reach along with some information, which can be used to determine the best path to each reachable destination. A router can determine the best vector (path) by examining the destinations reachable through each adjacent router or neighbor combined with some additional information, such as the metric, which indicates the desirability of that path. There are two types of vector-based protocols: distance vector and path vector.

Distance and path vector: From the standpoint of routing protocols, the vector is the interface traffic will be forwarded out to in order to reach a given destination network along a route or path selected by the routing protocol as the best path to the destination network. Routers advertise the vector (path) or distance (metric) or both for each destination reachable within the network to adjacent (directly connected) peers. Distance is the cost of reaching a destination, usually based on the number of hosts the path passes through, or the total of all the administrative metrics assigned to the links in the path. This information is placed in a local database as it is received, and distance plus an outgoing network interface is used to determine which path is the best path to each reachable destination. Once the best path is determined, these best paths are advertised to each directly connected adjacent router.

Distance vector protocols are usually considered light weight protocols as a node keeps record only for its neighbors and directs the messages to that neighbor that has the path to the destination. However, link state protocols provide the flexibility to utilize the knowledge of faster links with possibly more hops than slower links with fewer hops (a much desired feature in an opportunistic network protocol). Therefore, we argue that, it is desired for a DTN protocol to contain wanted features of both categories and may reside on the bridge between Distance Vector and Link State protocols. Such a hybrid protocol is not totally a novel idea as similar solution have been presented in traditional network routing where they were such requirements [SCE+05].

3.3 Multi-path Routing

Studies in [KMK04], [PP03], and [TH01] show that the shortest path algorithm may not be a good choice for MANETs. When the shortest path algorithm is used, nodes located around the center of a network carry more traffic compared to other nodes that are located at the perimeter of the same network. Particularly, when multiple connections are set up in a network, the wireless links located at the center of the network carry more traffic and the network can therefore become congested. This type of congestion problem may affect the performance of a network in terms of delay and throughput. In highly mobile scenarios, the shortest path may break due to node movement. Moreover, communication through a wireless medium is inherently unreliable and is also subjected to link errors. While designing a multi-path routing protocol, the following two major fundamental issues have been addressed in the literature [TTAE09]:

Path discovery One of the major reasons for using multi-path routing is to discover multiple paths that should be node-disjointed or link-disjointed. In the node-disjointed paths, nodes on the paths should not be common while, in the link-disjointed paths, links on the paths should not be common. Once all node-disjointed or link-disjointed paths have been discovered, other issues arise such as, how to select a suitable path or a set of paths from all the discovered paths and what node should make this selection, namely, the source or the destination.

Path selection Once multiple paths are discovered, a multi-path routing protocol should decide how to select a path for sending data messages. If a number of paths are discovered, there is a question as to how many of these paths should be used–only a few as in [Lee02]– or all of them as in Tsirigos et al. [TH01]. If only a few paths are used, the performance of a multi-path routing protocol should be similar to that of the shortest path routing protocol. On the other hand, if all paths are used, there is a chance of selecting an excessively long path, which may adversely affect the performance of a multi-path routing protocol.

Once a path or a set of paths is selected, yet another issue arises, i.e. how a source node should send a message. It may divide a message into multiple fragments and send these fragments by using different paths or it may send duplicate copies of a message by using different paths.

3.3.1 Advantages and disadvantages

1. **Fault tolerance:** Multi-path routing protocols can provide fault tolerance by having redundant information routed to the destination via alternative paths. This reduces the probability that communication is disrupted in case of link failure. More sophisticated algorithms employ source coding [AIGM93] to reduce the traffic overhead caused by too much redundancy, while maintaining the same degree of reliability. This increase in route resiliency is largely depended on metrics such as the diversity, or disjointness, of the available paths. We delay the discussion on disjoint routes to the next section.

2. **Load balancing:** Load balancing is the ability of a routing protocol to distribute traffic equally among the nodes. When a link becomes over utilized and causes congestion, multi-path routing protocols can choose to divert traffic through alternate paths to ease the burden of the congested link. A proactive load balancing strategy should decide how to use these multiple paths while sending data messages, once a path or a set of paths is selected. While using the paths, the following issues need to be addressed:

   - All paths can be used in a round-robin fashion [WL05].
   - Paths can be selected at random.
   - One path can be selected to send a preset number of messages [CL05] and then a different path can be selected to send the same number of messages.
   - Paths can be selected to satisfy the reliability or the delay constraint of a network.

3. **Bandwidth aggregation:** By splitting data to the same destination into multiple streams, each of them routed through a different path, the effective bandwidth can be aggregated. This strategy is particularly beneficial when a node has multiple low-bandwidth links but requires a bandwidth greater than an individual link can provide. End-to-end delay may also be reduced as a direct result of larger bandwidth.

4. **Reduced delay:** Employing single path on demand routing protocols for wireless networks, a route failure means that a new path discovery process needs to be initiated to find a new route. This results in a route discovery delay. The delay is minimized in multi-path routing because backup routes are identified during route discovery.

Although, multi-path routing protocols improve load distribution, reliability, delay and energy efficiency, they also have some disadvantages:

1. **Longer paths:** In a multi-path routing protocol, some messages may travel for longer hops compared to the shortest path routing protocol. Hence, these messages suffer longer delay. To avoid using excessively longer paths, a multi-path routing protocol should not use all the discovered paths. Instead, the protocol should use a selected number of paths.

2. **Special control message:** In addition to route discovery and route maintenance control messages, a multi-path routing protocol uses additional control messages. The additional control messages are used by a mobile node to collect information about its neighbors so that suitable (node-disjointed or link-disjointed) paths are discovered. The special messages used in a multi-path routing can overwhelm the network, especially when the network is large. These messages can occupy a significant portion of the available bandwidth and hence can adversely affect the performance of a network.

3. **Route request storm:** A multi-path routing protocol can generate a large number of route request messages in a MANET. In the routing protocols proposed in [Lee02], [YZ03] and [MS05], an intermediate node is not allowed to discard a duplicate request message. Instead, intermediate nodes forward a duplicate request message, which can cause a large quantity of redundant overhead messages in the network.

4. **Inefficient route discovery:** The route discovery process of a multi-path routing protocol may not be as efficient as traditional protocols. To find node-disjointed or link-disjointed paths, the multi-path routing protocols proposed in [YZ03] and [MS05], prevent an intermediate node from sending a reply from its route cache. Thus, a source node has to wait until a destination replies back. Hence, the route discovery process of a multi-path routing protocol takes longer than that of DSR or AODV protocols.

5. **Duplicate message processing:** To ensure reliable data transfer, the multi-path routing protocols proposed in [TH01, WL05] send duplicate messages using different paths. Duplicate messages create redundancy and thus occupy useful bandwidth. Moreover, to generate duplicate messages at the source and to filter out these duplicate messages at the destination, a special arrangement is required.

Chapter 4

Trace Analysis

> "I try to trace the connection between the characters and that way a story or plot emerges". Anita Desai

This chapter discusses the conventions and traditions that exist in the research community for the selection of test data for routing protocol simulations. Other then presenting an example of a real life setup for DTN, we discuss the motivation behind relating social networks to the field and why, in our opinion, this relation is not valid. We, then describe the source and motivation for all of our trace selections and present the filtering process and synthesis of the selected data from a statistical view point.

4.1 Relationship between network and protocol

Typically, an ad hoc network protocol evaluation is performed using network simulation with a mobility model that dictates how nodes move. While nodes move, a network path (a cascaded list of nodes) is generally assumed to be available between any sender and receiver. One of the reasons behind such an approach is that research in wireless networks is seriously starved for data. Data captured from live wireless networks can help us to understand how real users, applications, and devices use real networks under real conditions. Moreover, this data helps us to identify and understand the real problems, to evaluate possible solutions, and to evaluate new applications and services. In contrast, most research today is based on analytical or simulation models. Due to the complexity of the real-world radio propagation and the lack of understanding about behavior of wireless applications

and users, these models are severely limited. Experimental studies, however, are extremely difficult to set up. To collect data about real users on real networks requires extensive amounts of equipment, specialized software for collecting and anonymizing data, organizational permission and assistance to collect data, and human-subjects research clearance from the appropriate legal authorities.

In order to route messages in wireless ad hoc networks, protocols have to predict the network topology. One of the important aspects in designing an efficient and scalable routing protocol is to take advantage of the underlying topology nature, movement patterns of devices and their respective transmission capabilities, cluster densities, network diameter etc. Therefore, mobility models have significant influences on important metrics for path-based routing protocols, such as link duration and path duration [CBD02, BSH03, SBKH03]. Hence, it impacts performance of path-based routing protocols by increasing cases of path breakages and route salvaging/maintenance operations are required [BBH04]. Intuitively, mobility patterns should also have impacts on delay tolerant encounter-based forwarding, since different mobility pattern will certainly change the frequency and duration of encounters. The decisions such as, whether to use multi path routing,the number of message replicas to be created, to choose between reactive and proactive protocol, can be easily made keeping in mind the network characteristics discussed above.

4.2 Current practices

In most of the works [TSR08b, WJMF05, LTZG06], it is customary to create an artificial environment with assumptions for node speed/velocity, node coverage area, i.e. node transmission range [LFC05, LDS03, MHM05]. In such cases, the direction of nodes is dictated by the random waypoint model [CBD02] and these variables are then manipulated to create several variations of dense or sparse networks. In our opinion, these kinds of environments may be suitable for simulating wireless ad hoc networks, however, they fail to capture the true essence of opportunistic networks. Wireless ad hoc networks are usually vicinity bound and enclosed mostly in one building or a field where the network is mostly connected.

Liao et al. [LTZG06] for instance, utilized a restricted random waypoint model. Different from the traditional random waypoint model, each node in the restricted random waypoint model has a given set of waypoints. After the node arrives at a waypoint, it will remain at that point for a random “thinking time”. Then the node will randomly pick up one destination from its waypoint set and move towards that point. Therefore, instead of having uniform visiting probability on the whole area, each node may only have a high chance to visit points in its waypoint set.

Moreover, they have selected a network of 150 nodes moving in a 10000 x 10000 unit square area with no node failures.

We can find other examples where something other than the random waypoint model has been used. Work in [MHM05] has used a group based mobility model with a form of hierarchical clustering that better reflects the ways in which collections of people are structured at an organizational level and, consequently, the ways in which they move. We can see this as a hybrid approach where a model is inspired by real life data but then in a second step hypothetical network is created by defining simulation area, device speed, group speed, etc.

Despite of the fact that these variables can be tuned to create dense or sparse networks, as we will show in the following sections that artificially generated simulation environments are not a good replacement for a real-world scenario. In a real-life scenario, it is probable that not all the devices are available for the whole period of simulation, i.e. new nodes join the network in the middle of trace period. On the other hand, existing nodes leave the network in the middle of a trace period, and different segments of network do not only have time dependent connectivity but also have highly irregular contact patterns [IW08].

Thus, opportunistic network exhibits characteristics that can only be partially covered by wireless ad hoc networks. Devices constituting an opportunistic network, may become disconnected from the network either due to being out of range or the user having turned them off. Duration of such lapses can be of random length and are hard to predict. Moreover, the network location where a device will reconnect after such a disappearance can also not be predicted.

4.2.1 In the context of social networks

To ease the above mentioned problem, some efforts have been made to learn the characteristics of social networks and the corresponding mobility model so that artificial models with different parameters can be built. As the movement of data mules (devices) is dependent on users carrying them, it is intuitional to model connectivity based on social behavior of humans. If we jump into the deep waters of social sciences to find routing solutions, we must be able to predict the meeting patterns of people. To be precise we face questions such as, can we predict the contacts of any person, and, most importantly, how often and how long do the contacts last? Can we identify people who have many contacts but do not belong to the one community? Such individuals known as connectors, act as a bridge between two or more communities and play a very important role in information

diffusion. Another important issue is to identify of clusters in such a way that they will aid us in routing.

Efforts such as those by Musolesi and Mascolo [MM07, MM06] and [KSB09] try to grasp the social structure of the users. In such a mechanism, we may take advantage of personal information of the user like age, profession, gender, hobbies etc. to detect the connections among network units and use the information to compute the routing information. Some of these works have based their decision of creating these imaginary simulation scenarios on the conclusion that most of the observed social encounters follow power laws [BC03a]. It is argued that human behavior depicts power law, both in social contacts and physical location. From the pool of all possibilities, humans spend most of their time at a few locations with a few close contacts.

Based on these assumptions, authors in [DH07, HCY08, CMMP08] have argued that we may consider only social networks because opportunistic networks may be realized as Pocket Switch Networks (PSN) where the handheld devices of the users carried in their pockets, form an ad hoc network. Since the problem at hand is to develop an efficient message transfer protocol for PSNs or opportunistic networks, it is very reasonable to assume for the underlying network to have characteristics of a social network. Where users communicate only with their friends or people they already know, every source knows the destination before there is any message transfer between the two.

Chaintreau, et al. [CHC+06] have studied transfer opportunities between wireless devices carried by humans and observed that the distribution of the inter-contact time, meaning the time gap separating two contacts of the same pair of devices over a large range of value, exhibits a heavy tail, (such as one of power laws) over a large range of value. Based on this conclusion, Spyropoulos, et al. [Spy07] has assumed that 100 nodes move according to the Community-based Mobility Model [SPR06]. In the Community-based model, each node has its own small community (network size = 500 x 500 unit, community size = 50 x 50 unit) inside which it moves preferentially for the majority of time (e.g., the user's department building on a campus).

Whether a social network model is valid for communication in mission critical environments remains an open question. In our view, it is not that easy to correlate opportunistic networks with social networks. High quality friendship is characterized by prosocial behavior, intimacy, low level of conflicts and rivalry. Metrics like contact frequency and time duration spent around each other can only be justified for characterizing opportunistic interactions. Humans may maintain their

personal and official connections but this does not necessarily mean that they regularly spend time with each other. On the other hand, the device interaction includes phenomena also known as "virtual friends". Virtual friends are the devices of those people, who are not acquainted with each other. Due to common working or living vicinity of device owners, the transmission range of these devices overlap frequently. Such phenomena results in creating these unintentional contacts that can make a big difference for an opportunistic network routing protocol.

4.2.2 Real-life setups

We can also find more practical examples where a lot has been invested to spread the nodes using the existing infrastructure. In the case of [BGJL06, DF07, KO07], the metropolitan bus service has been equipped with wireless devices so that their traces can be used as a realistic mobility model for the verification of the desired protocols. As represented in Fig. 3.5, it is relatively easy to predict the movement of devices. Similar to wired networks, one cannot only position devices with strong communication abilities, at more frequently used routes and also configure the routing tables to balance the traffic load among all the available paths. The protocols that emerge as an output of such an extensive experimental setup may benefit from the time schedule and the regular nature of the transport service, such as regular dispersion of the messages that are back-logged during the time when buses/service is either not running or at all with less frequency.

Although, such an approach may provide a very suitable solution for the network for which it is designed, it may be very much prone to failure, if the underlying setup is disturbed resulting in a very case specific solution. These networks also receive the assistance from several aspects including predictable meeting schedules and static paths (public transport), stationary routers (Inter-hop access points), GPS, Internet connectivity, etc. Presence of such steroid based assistance is a good motivation to test these mechanism in a so called neutral environment.

4.3 Data selection

We argue in favor of real-world traces for the performance analysis of different protocols. We also believe that it is mandatory to test the results in several different scenarios to obtain an acceptable confidence level of any routing mechanism. This step can be complicated, if we have to gather multiple real-life traces to run our tests.

The network trace captures both, underlying on-off usage pattern and mobility

access_point (string, anonymized)	day (date)	moment (time)	mac_address (string, anonymized)

Figure 4.1: IBM trace structure

pattern of nodes. It can be easily argued that user traces, which include information of cell tower IDs and the duration for which they have been connected to these towers may be used for proximity discovery. Similarly, in the case of access points, the SNMP protocol maybe utilized for similar purposes. A recent idea is to use bluetooth traces. Bluetooth is a wireless protocol supported by most of the PDAs, mobiles, and laptops. Bluetooth gives wireless connectivity in the range of 1 - 50 meters depending on the device. All these economic and technological trends support the direction of our research.

We have opted for real-life traces and considered three different kinds of data sets, all of which have been obtained from the Community Resource for Archiving Wireless Data At Dartmouth[1]. CRAWDAD is a wireless network data resource for the research community that has the capability to store wireless trace data from many contributing locations. Additionally, it allows the researchers to develop better tools for collecting, anonymizing, and analyzing the data. As already stated, we decided to utilize data available from ready resources. However, since this data was not in the form we would have preferred, we developed some tools to extract the useful information from these large amounts of data.

4.3.1 Trace Description

We have considered three different kinds of data sets, all of which have been obtained from CRAWDAD. The motivation behind choosing these three traces has been to have a broad spectrum between dense and sparse networks. Two of the data sets have been synthesized from the reality mining project [EP05] at MIT and spans on 16 months, i.e. February 2004 to August 2005. The third data set consists of the SNMP logs for one month from a IBM campus [BC03b]. As the duration span of the MIT reality mining trace is longer than that of the IBM trace, we have filtered the MIT data to match the time span of the IBM trace. The relevant filtering issues for each of the selected traces are discussed in the following sections.

[1] http://crawdad.cs.dartmouth.edu

oid	endtime	starttime	person_oid	celltower_oid

Figure 4.2: MIT trace structure

Access point -*IBM*

Simple Network Management Protocol (SNMP), a UDP-based network protocol, has been used to log IBM trace data. Generally, SNMP is utilized mostly in network management systems to monitor network-attached devices for conditions that warrant administrative attention. In the case of the IBM access point trace, SNMP was used to poll access points from July 20, 2002 through August 17, 2002. A total of 1366 devices have been polled over 172 different access points during approximately 4 weeks. Each of the 172 access points has recorded the mac address of those devices that have been connected to it at that point in time after every 5 minutes.

To turn these samples into continuous data, we assumed that the snapshot data would remain constant for the next 5 minutes. Therefore, we assumed that one such record would show that one device had been connected for 5 minutes to the access point that provided that log record. This assumption introduced a few irregularities such as a few devices that had been logged at two different access points within a period of 5 minutes. This can be explained by the fact that the clocks on all the access points had not been synchronized and that it had been possible for a mobile device to have been recorded with 2 different access points within 5 minutes. In those rare cases where this would cause an overlap with another snapshot from another access point, we assume that the transition happens halfway between the 2 snapshots. Moreover, if a devices has been consecutively logged at one access points for more than five minutes, the period of connectivity is, as expected, prolonged until 5 minutes after the time of last recorded log.

Cell towers -*MIT*

This trace holds exact timestamps recording the exact time durations of device contact times, which relieved us from playing with the trace the way we had to do with the IBM trace. However, as mentioned by the creators of the data, due to several lapses in data gathering, , only 89 of 100 devices are included that roamed through 32768 different cell towers. In the case of the MIT cell tower as

Figure 4.3: MITBT trace structure

well as the bluetooth (explained in following section), a mobile application has been responsible for logging the connectivity of that device. In the case of the cell tower, a log record has been generated whenever a devic has been disconnected from the cell tower recording the connectivity duration of that device with the corresponding cell tower.

The MIT cell tower trace spans around 16 months, i.e., from February 2004 to August 2005. For a meaningful comparison among the traces, we reduced the time span of the MIT trace to 1 month. From the 16 available months, we decided to select the 1 month with the maximum activities. We define activity as the commutative time durations that all devices in the trace spent while connected to cell towers. November 2004 here proved to be the month with maximum activity where 81 devices had been roaming through 12592 cell towers.

Bluetooth - *MITBT*

For bluetooth traces, the bluetooth capability has been used which allows them to collect information on other Bluetooth devices within 5 - 10 meters. This information includes the bluetooth MAC address (BTID), device name, and device type. Bluetooth was primarily designed to enable wireless headsets or laptops to connect to phones, but as a byproduct, devices have also become aware of other bluetooth devices carried by people nearby. The generated logs have recorded timestamps with BTIDs encountered in a proximity.

The MIT bluetooth trace, like its brother trace, spans also around 16 months. Each device scanned every five minutes for active bluetooth neighbors. We used a similar criteria, which we employed for the MIT to limit the trace to one month, and where any visible bluetooth device was considered a candidate connection. In this case, we define the activity as commutative time duration spent by all devices while connected to other devices. Not surprisingly, November 2004 proved to be the month with maximum activity. Although, we observed a lot of devices that were not running the logging application, the resulting connectivity remains sparse. The trace showed 1858 bluetooth devices suggesting a huge number of

unknown devices as compared to the 89 known devices that were used to gather the data.

4.3.2 Peripheral issues related to IBM and MIT

In the above mentioned two traces, the connectivity model assumes that whenever two devices are connected to one base station, may it be an access point or a cell tower, a direct edge exists between the two devices unless one of them is disconnected from the common base station. In the case of access point, the wireless range is comparatively less (usually less than 100 meters) and this scenario practically simulates a relatively sparse network with low mobility and low device density. On the other hand, cell tower is characterized by its long communication range (in urban environments up to 5 km) and practically forms a dense network with high mobility. Although, the frequency with which the wireless beacon is emitted effects the timings of node connections when nodes encounter each other, for the sake of simplicity, we will here assume that nodes instantaneously "see" each other when they come within range.

It is imperative to mention that the assumption that two devices connected to one base-station (access point or cell tower), does introduce inaccuracies [CHC+06]. On one hand, this is overly optimistic, since two devices attached to the same access point may still be out of range of each other. On the other hand, the data might omit connection opportunities, since two nodes may pass each other at a place where there is no base-station, and, thus this contact would not be logged. Another issue regarding these data sets is that the devices are not necessarily co-located with their owner at all times (i.e. they do not always characterize human mobility). Despite these inaccuracies, such traces are a valuable source of data, since they span many months and include thousands of devices. In addition, the assumption that two nodes connected to the same access point, and are potentially in contact, is not altogether unreasonable, as these devices could indeed communicate through the access point without using end-to-end connectivity.

4.3.3 Peripheral issues related to MIT and MITBT

There have been some holes and errors reported by the creators [EP05] of data in both of the reality mining traces. For example,users either had not carried the devices with them or had them not turned on continuously; due to several reasons. Additionally, some cases of exhaustion of local memory on mobile phones have been reported that caused loss of some data. The application crash has also been one of the cause for the hole in the data.

We have simulated three different configurations of source-destination pairs for the MIT trace. We have constructed two different source-/destination pair configurations for November 2004 to confirm the behavior of protocols. This way we have tried to avoid the coincidental bias to any protocol that may have been due to the nature of the input. Furthermore, for out own interests, we have created one more configuration for another month that has the second highest activity among the traces. This helps us to see the correlation between the different months depending on the nature of activity for several protocols. The second most active month proved to be October 2004 with 79 devices and 11784 antennae. The results were not only statistically similar, but the performance of the algorithm was close to what we observed in November 2004. This supports our hypothesis that the month of November is not an exceptional case.

The bluetooth trace had 81 participant devices but several non-participant devices had more online time than participant devices. We therefore decided to include them in our analysis as well, which resulted in a device count of 1858.

4.4 Trace profiling

Several methods could be employed to describe a mobility pattern–for instance, it could be based upon history information regarding contacts that the node has already had. Mostly in opportunistic routing, a contact space or a mobility space is based on the assumption that there will be regularities either in the contacts that nodes have or their choices of locations that they visit. There is always the possibility that, we may encounter mobility patterns similar to the ones observed with random mobility models. The efficiency of the virtual space as a tool may be limited, if nodes rapidly change their habits. Some problems could occur even if nodes have well defined mobility patterns, e.g. a message may reach a local maximum if a node has a mobility pattern that is the most similar in the local neighborhood to the mobility pattern of destination node. For one reason or another, this argument is not sufficient to achieve the delivery. It is also possible that nodes visit similar places, but for timing reasons, i.e. as being on opposite diurnal cycles, they never meet.

Researchers are also interested in building synthesis models that capture important factors of these underlying factors of the traces [MWYL04] [TG05]. In our opinion, regular patterns in a trace can be found only in those traces that are purposefully designed with such aspect in mind. As discussed earlier in Section 4.2.2, we can find a few real life deployments of opportunistic networks [KO07, BGJL06], all of which receive either the assistance by inserting Internet-Connected data mules

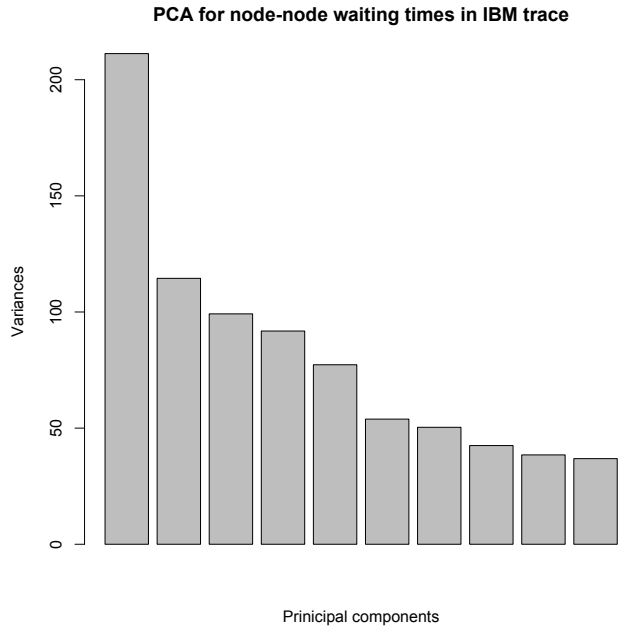


Figure 4.4: PCA with respect to inter-device waiting time

and/or fixed contact schedules. Regular patterns can be found in the contact timings of devices mounted on public transportation systems.

In contrast, the network trace we used is actually a combined result of underlying factors that may have influences on the usage pattern, such as user mobility, preferences, correlation between users, etc. Our trace analysis shows that it is not easy to find a particular contact behavior or pattern among the nodes in real-life traces. We have analyzed three different networks obtained from real-life traces that have considerable variations among network size and density.

4.4.1 Trace analysis using PCA

Principal Component Analysis (PCA) is a statistical technique for determining the key variables in a multidimensional data set that explain the differences in the observations, and can be used to simplify the analysis and visualization of multidimensional data sets. Speaking in visual terms, PCA can be used to find a projection to a plane – or more generally, to a linear subspace – which preserves as much as possible of the original variance in the data. The solution of the optimization problem of finding the projection plane such that the variance of the projected data is maximized leads to an eigenvalue problem [BBHK10]. This method for dimension reduction that is based on two assumptions:

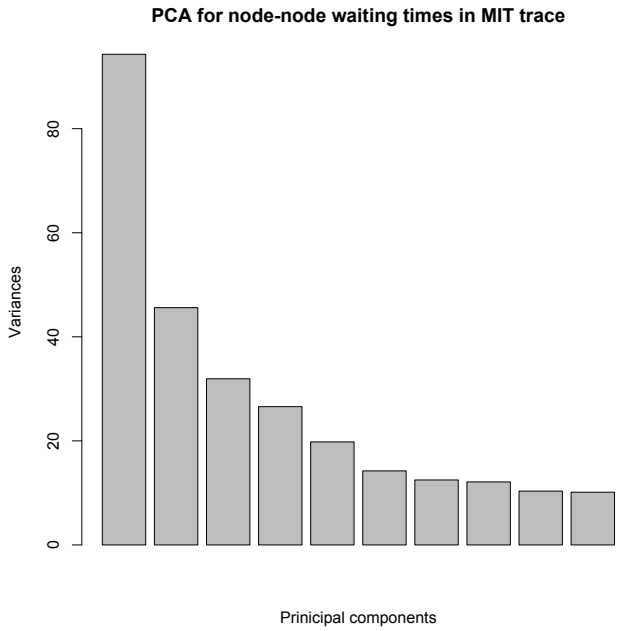


Figure 4.5: PCA with respect to inter-device waiting time for MIT

- The representation of the data in a lower-dimensional space should be obtained by a projection to a linear subspace.
- The criterion to evaluate the representation of the data in the lower-dimensional space is the preservation of the variance.

PCA can be used for visualization purposes by considering only the first two principal components. More generally, PCA can carry out a dimension reduction to any lower- dimensional space; even more, PCA also provides information about how many dimensions the data set actually spreads. This information can be extracted from the eigenvalues $\lambda_1 \geq \cdots \geq \lambda_m$ of the covariance matrix [BBHK10]. When we project the data to the first q principal components $v_1, \ldots\ldots, v_q$ corresponding to the eigenvalues $\lambda_1 \ldots\ldots \lambda_m$, this projection will preserve a fraction of

$$\frac{\lambda_1 + \ldots\cdots + \lambda_q}{\lambda_1 + \ldots\cdots + \lambda_m}$$

Because reduction of dimensionality (focussing on a few principal components versus many variables) is a goal of principal components analysis, several criteria have been proposed for determining how many PCs should be investigated and how

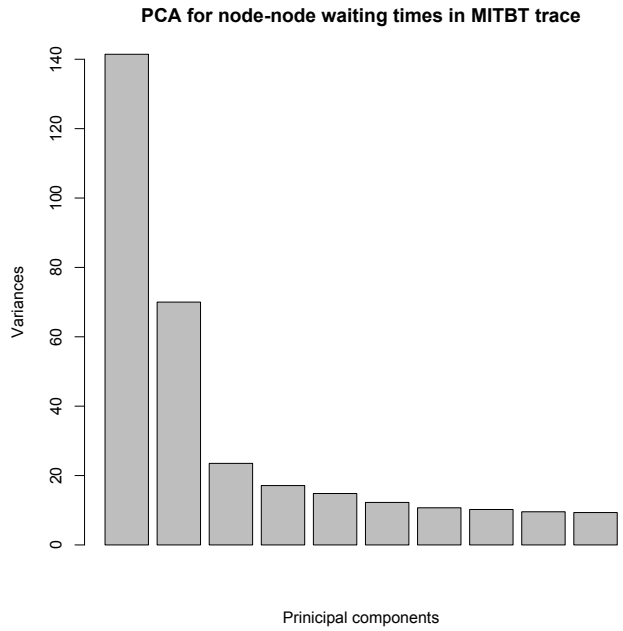


Figure 4.6: PCA with respect to inter-device waiting time for MITBT

many should be ignored. One common criteria is to ignore principal components at the point at which the next PC offers little increase in the total variance explained. A second criteria is to include all those PCs up to a predetermined total percent variance explained, such as 90%. A third standard is to ignore components whose variance explained is less than 1 when a correlation matrix is used or less than the average variance explained when a covariance matrix is used. Tthe idea being that such a PC offers less than one variable's worth of information. A fourth standard is to ignore the last PCs whose variance explained is all roughly equal [AS95].

Inspired by this discussion, our motivation for the following experiment is to classify devices based on their contact patterns with the assumption that daily routine remains more or less the same for the majority of people. In other words, we want to see if we can reduce the dimensions of our trace data assuming the presence of a few such devices whose contact patterns can summarize the pattern of the majority of the remaining devices in the trace.

To verify this behavior with the help of PCA, we have extracted the following dissimilarity metrics,

- Mean contact waiting times: Average time two devices had to wait for their next meeting. If two devices came into contact only once, the waiting time

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Standard deviation	9.71	6.75	5.65	5.15	4.44	3.77	3.53	3.48
Proportion of Variance	0.25	0.12	0.08	0.07	0.05	0.04	0.03	0.03
Cumulative Proportion	0.25	0.37	0.45	0.52	0.58	0.61	0.65	0.68

Table 4.1: Importance of components obtained from MIT PCA summary

was considered to be 1 week.

- $1/contactFrequency$: Average number of times two devices come into contact with each other on a daily basis.
- Combined metric: Ratio between mean contact durations and contact frequencies. $meanDuration/contactFrequncy$.

We used PCA on all of the above mentioned metrics expecting that the first few components will account for most of the variability, which could be interpreted as an indication that as far as contact patterns are concerned, the majority of devices in the networks are classified in a few categories. We considered only office timings, i.e. 09:00 – 17:00 hrs, for experimentation. The reason being that variations during the office hours are bound to be lower than evening hours. The PCA performed on all the above mentioned metrics, unfortunately failed to provide any hint of patterns. The PCA results of device-device waiting time of the IBM, MIT and MITBT traces are shown in Fig. 4.4, 4.5 and 4.6 respectively. We cannot identify the *principal components* in any of the figures using any of the cut off criteria mentioned before. When we look at the PCA summary of the MIT trace in Table 4.1, the cumulative proportion of the first 8 components has failed to reach to 0.8 suggesting that these component have failed to cover up the majority of the variance.

We also experimented with the rest of the metrics mentioned earlier but they all showed similar pattern; hence, the their output is not included here. We observe that there is no significant difference between the output of PCA performed on all the traces other than the magnitude of variance, which is the minimum for the dense MIT trace and the maximum for the access point IBM trace. It is imperative to mention that our primary goal is to reduce the dimension of our traces (by identifying the principal components) therefore the different magnitudes of variances at Y-axis the for the three traces are meaningless for our analysis.

Chapter 5

Simulation Setup

> "Prediction is very difficult, especially if it's about the future." Niels Bohr

This chapter shows the process of data transformation of all the selected network traces for a common simulation platform. It then analyses the traces on the basis of group activity. Group activity is defined as time duration and number of devices in the meeting(group). Following, we describe all the mechanisms that we have chosen on the basis of their recency to the community. At the end, we briefly discuss the criteria according to which we will be comparing the performance of these mechanism. In this chapter, we observe the evidence of trace density and understand the issues related to working of the simulator.

5.1 Why simulate?

Simulation has proven its value as a tool to understand the nature and extracting the model of mobile ad hoc networks. In a real world scenario, where repetition of complex event series is very unlikely, scientist apply simulation techniques on a variety of phenomena like weather, health economics, effects of atmosphere, etc. In opportunistic networks aspects like multiple paths, movement of devices, variable bandwidth, and obstacles, are either created artificially or obtained from a real world source. This helps a great deal to improve and refine the protocols, especially in the the study of ad hoc wireless networks.

Several attempts have been made to devise efficient routing mechanisms using

different experimental or simulated data whereby, different claims have been made in favor of several mechanism. We have observed that most of the attempts have made several over-simplistic assumptions, which either favor them for routing or give them access to unlimited resources. Hence, there is no way to verify the claims made by the researchers in an unbiased manner. We have made an effort to compare the performance of several protocols on the same realistic data so that their effectiveness can be compared. Moreover, we want to explore what kind of factors have been considered properly and which ones have been ignored. The ones ignored may have considerable effect on the performance of those mechanism that could only be exposed if these mechanism would have been tested in a different environment.

For this purpose we have embarked on developing a routing simulator that will help us to examine in depth the issues involved with existing solutions. We selected 8 different routing algorithms that have been presented in the recent past to the research community. We have simulated the selected protocols under different environments that vary with respect to the nature and the size of underlying network, varying bandwidth, number and sizes of messages. The variance in communication ranges of device in selected networks is such that these networks can be classified as sparse, medium and high density networks. These environments are introduced in section 4.2.2.

The variance in the approach of the selected simulated protocols is multidimensional. Some of these protocols are considered on the one hand computationally intensive and intelligent, while on the other hand they are not realistically deployable. Such protocols can be a good tool to understand the dynamics of the opportunistic network. We also have a few rather simple protocols that use brute force replication to disperse the messages in the network. Such protocols are believed to perform optimally but overhead costs make them very unattractive for deployment. Additionally, we also have selected a few protocols that can be seen as practical and partially intelligent that have been tested in a restricted environments (discussed in Section4.2). Most of these algorithms have used history to find the next best hop for the message to be carried to destinations. Some algorithms exploit replication to enhance the probability of message delivery. The extent of replication is primarily dependent on how intelligent the algorithm is. More intelligence means it needs less replication and intelligent methods can avoid the obvious overheads of replica generation and transmissions.

5.2 Simulation issues

Opportunistic networks being a newly developed area, therefore there is a lack of simulators for a comparative study. As mentioned previously, DTNRG is also working on a simulator that is, however, limited to introduce the functionalities of bundle and licklider layers into the TCP/IP stack. During the course of our work, we discovered The ONE (Opportunistic Network Environment) simulator [KOK09] that can route the messages between nodes with various routing algorithms. The ONE is also dependent on different movement models for the generation of node movement data, which was in conflict with our aim of using real life traces for performance analysis. Thus, we preferred to continue to work with our simulator with the goal to make it available for the research community.

The aim of our simulator is to help us determine the delays incurred by messages of different routing algorithms. The output is analyzed on the basis of both the number of messages as well as the amount of data delivered. As already mentioned, three different traces have been used that significantly differ in the number of devices involved as well as in the number, frequency, and distinctness of meetings that were taking place among the participants. As the span time of access point (IBM) trace is approximately one month, whereas for cell tower (MIT) and bluetooth (MITBT) traces is more than 1 year. Thus, we have chosen 1 month from cell tower and bluetooth data on the basis of highest activity, so that the results can be compared. We observed that November 2004 had the highest activity among all the months for which the reality mining (MIT, MITBT) trace has been recorded. Furthermore, to see the correlation with other months, for the cell tower trace, we decided to include the month that ranked second, i.e., October 2004.

5.2.1 Data transformation

For the sake of uniformity and simulation simplicity, we transformed all the three data sets into one universal format. We designed a XML Document Object Model (DOM) presented in Fig 5.1, and transformed all three above described data according to this DOM. As one may observe, the presented DOM is suited very nicely for the base station trace data, i.e. IBM and MIT but we have to improvise to transform the MITBT trace to make it suitable for given DOM. In the base station case, when none of the devices are connected to one base station, may it be access point or cell tower, there is no connectivity among devices. When only one device is connected to a base station for a particular duration, it also does not bring any connectivity. Two devices are considered connected to each other only when both of them are connected to the same base station during the same time

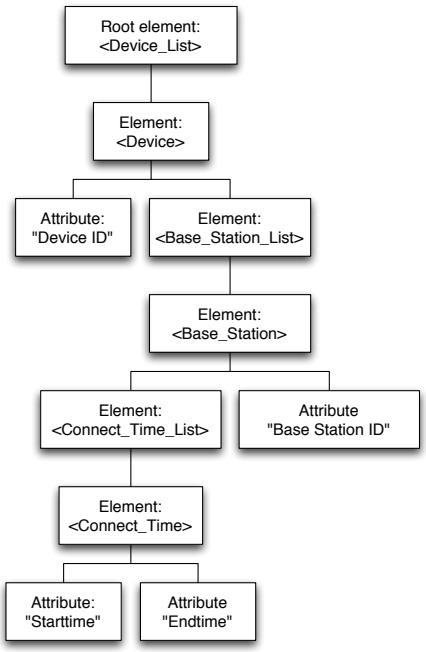


Figure 5.1: Common XML DOM tree for simulator input

period. These arguments are the motivation behind adding one of the bluetooth devices as a common base station for itself as well as other device with corresponding contact times. Once a device is selected as base station, it is considered to be the base station for all of its upcoming contacts. This way we will end up with a structure that represents multiple devices connected to each other, appearing as if they are connected to one base station for a overlapping duration of time. The algorithm for this transformation is presented as control flow diagram in Fig. 5.2.

5.2.2 Message creation

For each trace, we have created 100 messages by selecting random pairs of source and destination. Although, source and destination pairs are selected randomly, we have placed an eligibility criteria for a device to be a source or destination to reduce the size of the selection pool. It does not make sense to have a device as either a source or destination that has shown very little contact activity. For the dense and medium sparse cases of the MIT and the IBM traces respectively, devices that are included in 30th percentile with respect to their online time, can be selected as source and destination; all devices may help out as intermediate nodes.

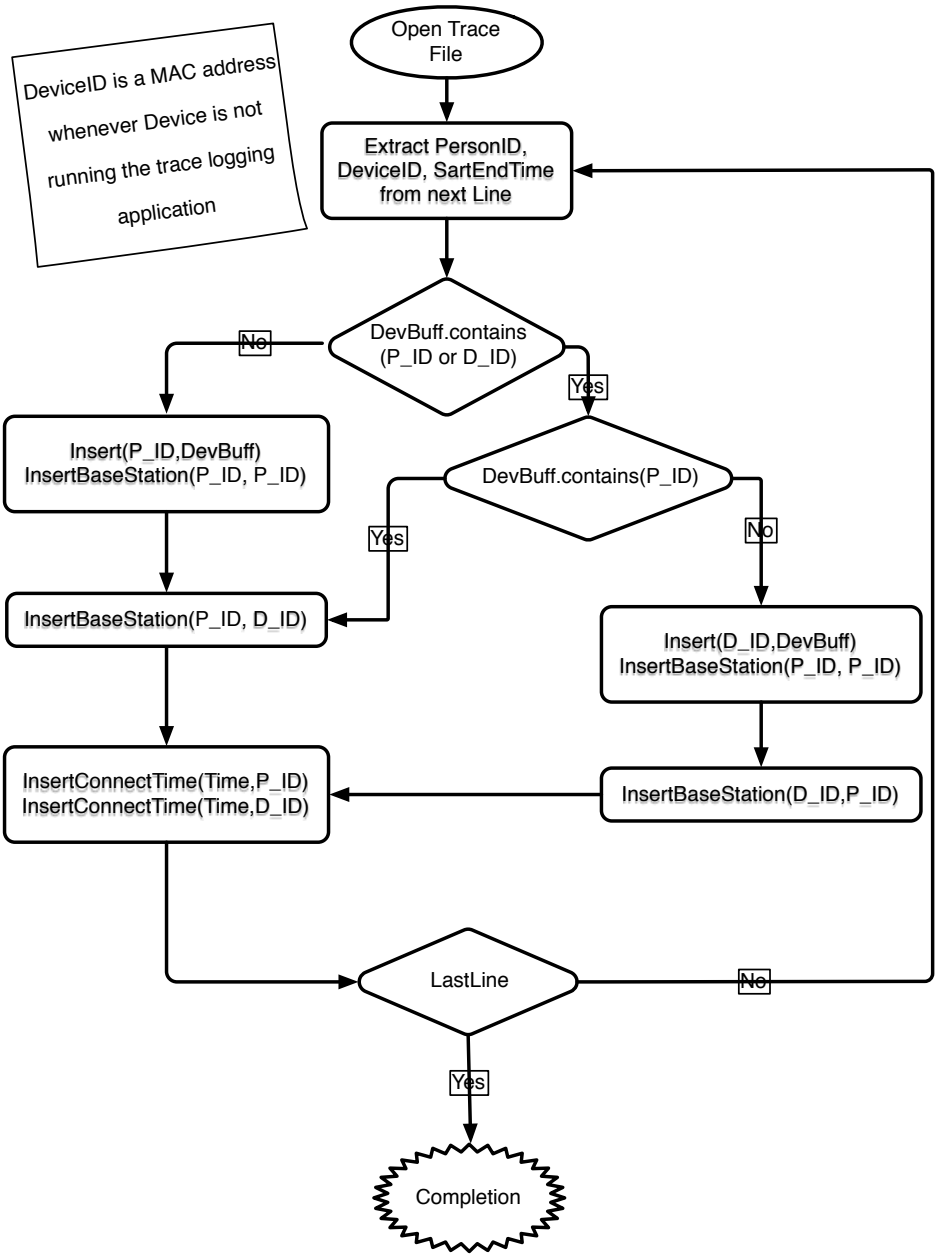


Figure 5.2: Transformation of MITBT trace for DOM Fig 5.1

Our initial results showed very little routing activity for all algorithms when we used the 70th percentile criteria with the bluetooth trace. Therefore, we have used only those devices as source and destination for the MITBT trace that are included in the 70th percentile with respect to online time. It was astonishing to note that this selection included some of the devices that were not designated to collect the traces. This shows that several devices that were originally designated

Message count	100
Message size	1.6E3. . . 1.6E7 B
Message size distribution	Power law
Message creation period	10th - 17th day
Message lifetime	7 days
History computation interval	10 - 360 min
Replication	$r = 4$
Erasure coding	$k = 4$
Bandwidth (low)	100 kiB/s
Bandwidth (high)	10,000 kiB/s

Table 5.1: Simulation parameters

for data collection were not active during the trace collection phase. This fact has also been mentioned by the data creators [EP05].

We have used the power law to determine the messages sizes, i.e. several small size messages and a few large size messages as shown in Fig. 5.3. Several studies can be found that have analyzed web traffic with respect to size and self similarity and this motivated us to model the message size for our experiments according to power law [PKC96].

As discussed earlier, we have dedicated the first 10 days for each algorithm to obtain history from the sources. Therefore, message creation starts from 10th day onward. The randomly selected birth times lie between the 10th and 17th day. We were forced to make adjustment in this regard because a few selected sources in the case of the IBM and the MITBT, did not appear in the traces during the 10th and 17th day. Consequently, we had to postpone the creation of message further up to the point in time when the source device appeared. This factor brought some unexpected issues into our simulations that are discussed later in Chapter 7. Moreover, we have used our intuition to select input factors for those algorithms that need predetermined parameters like replication and erasure coding. These simulation parameters are summarized in Table 5.1.

5.3 Simulation Methodology

The simulator follows an event based process where initial events are defined by the birth times of all the messages. Simulation, then continues by removing the first message to be born, from the system and consumes its life according to the connectivity patterns of the device carrying the message. During the consumption

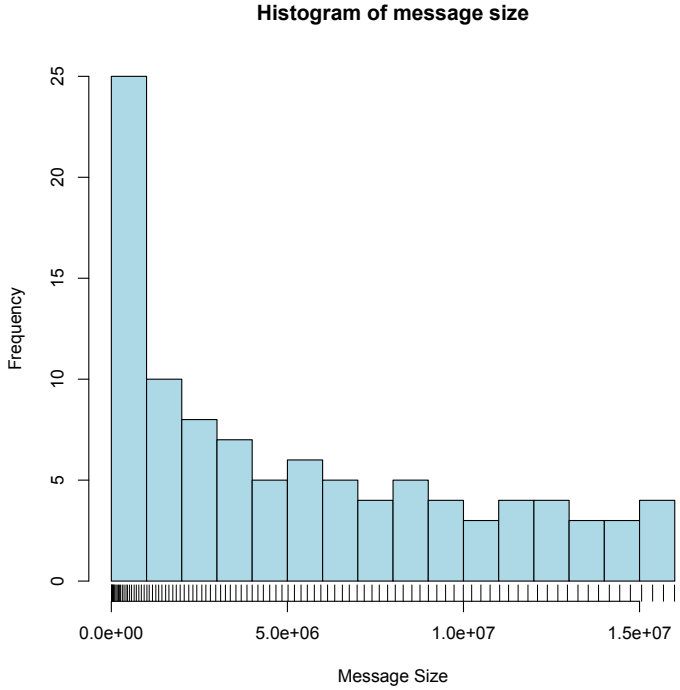


Figure 5.3: Message size distribution according to power law

of its lifetime, a message may be propagated to the next hop, replicated to the next hop or simply just lapse for some time when there is no good candidate-next hop has been in contact with its carrier. The choice between these decision is off-course made by the protocol that is being simulated at a particular time. The time consumed during propagation from one hop to another is dependent on the parameters of available bandwidth and size of the message. This process continues until the message either reaches the destination or its life time expires. We have simulated 100 messages of different sizes, the simulation ends when the last message exits the system.

5.3.1 History computation

As is evident from the above description, the simulation is driven only by message related events, the contacts between the devices that trigger no exchange of messages are not considered as an event. An independent history module computes history for algorithms that need any kind of history based profile summaries. We have dedicated the first 10 days of each trace for history computation. This means, all the contacts during the first 10 days are summarized according to the principal dictated by the respective routing protocol. Moreover, the history gathering

process continues once the message propagation starts after the history gathering period, by invoking the independent history module at regular time intervals.

As we will see in Chapter 7, the duration of time intervals effects the computation accuracy of history for those protocols that rely on transitive histories (A knows B and B knows C, this means A knows C). The smaller the inter computation interval, the more accurate transitive history tends to be. Due to computation time overhead, we cannot adjust this duration to as minimum as possible; we have consequently tuned the interval length between two successive history computation depending on the computation overhead of the routing algorithm. For the information of curious readers, this interval ranges from 10 minutes to 6 hours.

5.3.2 Miscellaneous issues

The simulation of opportunistic network is different from other network simulations as there is no constant connectivity among the devices, i.e. the underlying graph of the devices is changing very rapidly. The graph characteristics like depth, width, average fan out and topology, are all variables (some time dependent). Moreover, the devices are not online all the time. Even if a device is turned on but there is no nearby neighboring device, we consider this device as being offline. This phenomena occurs in base station traces where only one device may be connected to a base station at a particular point in time. Due to all these issues, we faced some initial hiccups in our effort to make a simulator, but in the end we were able to find a stable model for simulating different algorithms.

Connection In the case of base station (IBM, MIT) traces, we define a transfer opportunity between two peers if they are connected for overlapping times to the same access point or cell tower. This setup is obviously not needed in the case of the bluetooth (MITBT) trace. As our simulations do consider control traffic to take no time at all, only the distribution of transfer times is significant: For low bandwidth, it takes $\approx 1.5 \ldots 15000$ seconds to transfer messages, and high corresponds to $\approx 0.015 \ldots 150$ seconds.

Cleanup In access point (IBM) trace, we removed 7 devices that had no connectivity to any other device due to spatial or temporal locality. We have found 3 clusters of devices in this trace, while it has been suggested by the creators that the trace is collected from 3 different buildings. We observed very weak connectivity among the clusters, and therefore, we used the network present in the largest building in which. We found 129 access points with 928 devices connected at different time intervals in the largest cluster. No such arrangement was required in

the rest of the traces.

Link sharing Another issue is to decide the number of simultaneous transmission a device is allowed to make. Each device can only participate in one communication with another device. We have taken a rather restricted point of view and allow a device to be involved in one transmission at a time. In other words, while two devices are exchanging one message, they become invisible to the other devices that are present in their radio range. When one transmission completes, the next message from the event queue is selected and allowed to create the next transmission. Thus, either the same or a different pair of devices can exchange messages in the following transmission.

Moreover, we have ignored the mac layer issues like frequency hoping and wireless channels intentionally to observe the real effect of the logic of protocol in forwarding a message. There are enough independent channels available so that any number of node pairs can communicate at the same time with full bandwidth, independent of their proximity to other pairs.

Shortcut We have tried to treat all the algorithms nicely by delivering the message to its destination in case a direct contact between the current carrier and the ultimate destination takes place. This even holds true, when the predetermined route requests to transmit to another node first. This propagation mechanism is depicted in Fig. 9.1.

5.4 Trace comparison

To understand the nature of our trace data, we have analyzed all of them according to the following three criteria.

5.4.1 Contact density

One way to visualize the contact density is shown in Figure 5.4. It shows that in the cell tower (MIT) case (both for the selected month of November as well as the preceding month), essentially all active nodes could directly communicate with every other node at least once during one month. In the access point (IBM) case, the majority of 928 devices were able to contact 50...100 devices, many of them up to 200.

In the bluetooth (MITBT) scenario, the 1858 nodes are getting in touch with at

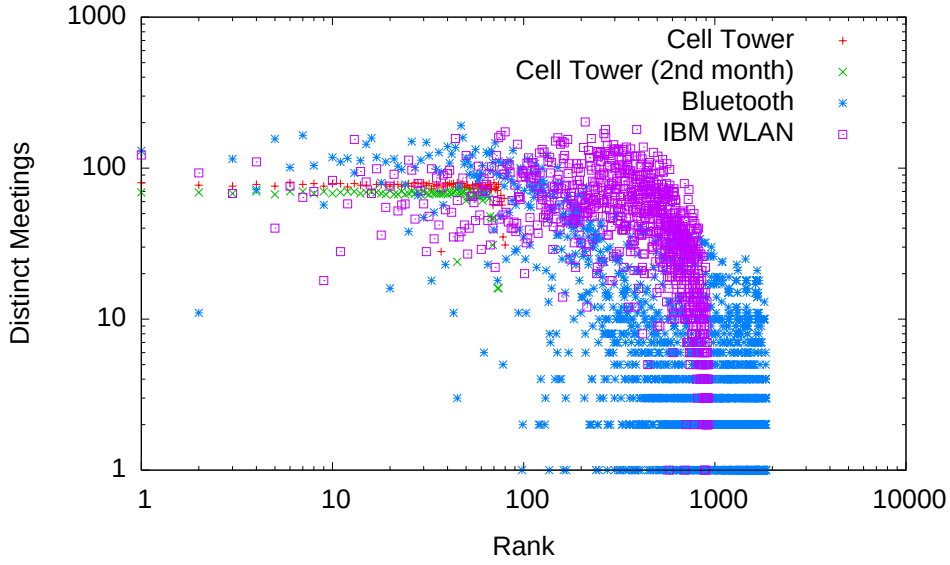


Figure 5.4: Comparison of all the traces: Rank of total meetings vs. distinct meeting count

the most with 100. . . 200 nodes, the majority only with 5. . . 50. Interestingly, some of the highest-ranking devices were not one of 89 participant nodes, but showed up more frequently in the communications range of trace group members than other members. We have to expect that these non-participants had contact with many other non-members; unfortunately, there is no way to tell given these traces.

We can see that the wider the communication range, the more the likelihood to communicate among peers; thus making the network more dense. On the other hand, the data communication rate is mostly lower for long-distance networks (e.g., access points vs. cell towers; not so much for bluetooth vs. access Points). For current networks, the effects of the density (or lack thereof) seem to dominate the bandwidth effects.

5.4.2 Meeting durations

If we look at the distribution of meeting time durations Fig. 5.5, we observe that although the meeting count in the MIT is very high, the durations of meeting are very small. The plot in Fig. 5.5(b) shows that there is large number approx. 100,000 of small duration (less than 5 min.) meetings and very few, approx. 1000 long duration meetings. If we consider the number of devices in the network, i.e. 89, we can conclude that the device movement has been considerably fast because devices have very frequent and very small duration meetings. The phenomenon

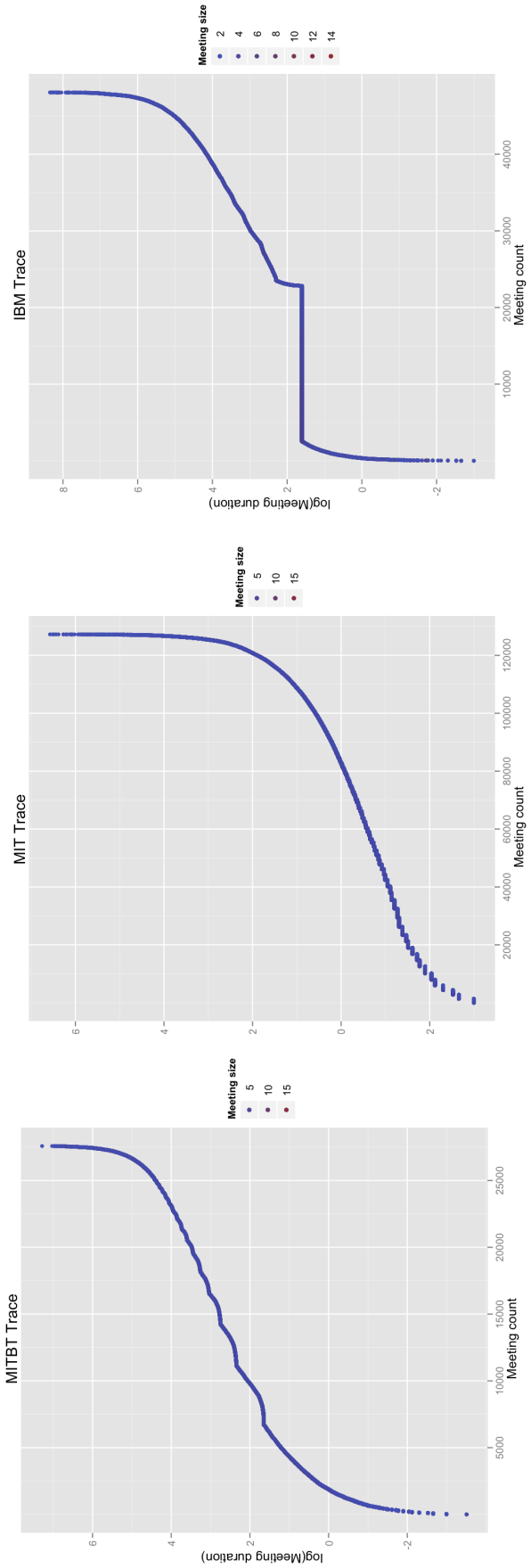


Figure 5.5: Natural log of meeting duration (a)MITBT, (b)MIT, (c)IBM

can be explained by the fact that the MIT cell tower data has been gathered mostly from an urban area where mobile service providers deploy the cell towers in such a way that their range frequently overlaps. In such a situation, when the users move, devices have to make regular hand-offs among cell towers to maintain connection quality. Thus, these regular hand-offs cause a large number of small duration contacts.

In the MITBT scenario shown in Fig 5.5(a), we observe approx. 5000 small duration meetings and then the curve eases up to show some medium duration (between 5 and 60 min.). Although, the MITBT consists of about 20 times more devices than the MIT, the meeting frequency is lower than that of the MIT. The reason being the small range of bluetooth radio waves, and considerable number of medium duration meetings shows that devices did spend longer time duration in each other's vicinity while carrying out common activities.

As expected, the IBM being the medium density network, shows approx. 250,000 meetings with small duration (less than 5 minutes) and a considerably high number of approx. 40000 of medium duration meetings (from 5 to 300 minutes). The big number of medium duration meetings evidences the lack of rapid movement among the devices. This could be explained by the fact that the IBM trace has been recorded from an office environment and time overlaps of coworkers are bound to be longer than those of other traces. Furthermore, our assumption of one device being connected to an access point for at least 5 minutes, may also have helped the IBM trace to get a respectable number of 5-minute duration meetings.

5.4.3 Meeting size

When we analyze the meeting patterns of all the traces with respect to days of the month, we see a clear working office pattern in the IBM trace. We observe a high meeting activity in terms of duration as well as size during the working days of the weeks followed by low activity weekends as shown in Fig. 5.6(c). The absence of this pattern is missing in both of the cell tower (MIT) traces because, MIT traces as shown in Fig. 5.6(a)(b), are logged by mobile phones and mobile phones are carried by their users not only during working hours, but also during the leisure activities. The access point (IBM) trace, on the other hand, has been logged with the help of office devices like laptops, palmtops, PDAs, etc, and the access points polled during the trace are only limited to one campus of IBM.

As far as the size of the meetings is concerned, both, Fig. 5.5 and Fig. 5.6, show the obvious dominance of blue color. This dominance represents that the frequency

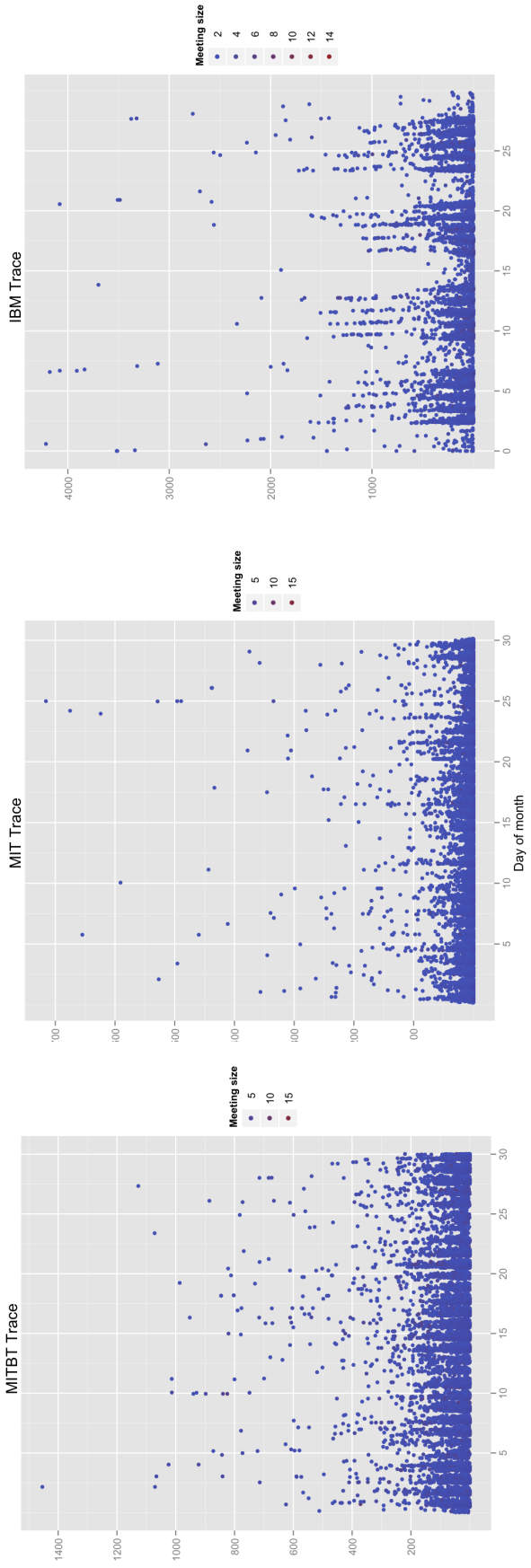


Figure 5.6: Meeting size (a)MITBT, (b)MIT, (c)IBM

of small size meetings is far higher than bigger group size meetings. In the access point IBM trace, the ratio of more than 5 people in a meeting is higher than in the other two traces. In contrast the MITBT has a considerable number of medium duration and medium size meetings that take place throughout the month.

5.5 Algorithms

We grade opportunistic network routing strategies based on the characteristics that a strategy uses in order to find the destination. The strategy may be replication based, thereby relying primarily on replicating the messages to enough nodes so that the destination receives at least one of the replica. In contrast, a forwarding strategy relies on knowledge about the network to select the best path to the destination.

Before presenting the descriptive summary of all the simulated algorithms, the qualitative summary in Table 5.2 shows that in contrast to *perfect oracle*, which achieves high performance on the cost of processing and communication, *flooding* has an equally high performance on the cost of storage and communication. *MaxProp* on the other hand, has an impressive performance just with processing cost. *Estimation based erasure encoding* is successful in reducing storage and communication overhead due to inherent advantages of erasure encoding.

Direct delivery. This strategy waits until the source comes into contact with the destination before forwarding the data. The source holds the data until it comes in contact with the destination. This can be a degenerate case of the flooding family, where the set of relays contains only the destination. It can also be considered a degenerate case of the forwarding family, where it only selects the direct path between the source and the destination. Since this strategy does not require any information about the network and only uses a single hop, it uses minimal resources. In spite of its simplicity, it may incur long delays [WJMF05] and frequently shows poor performance (Table 5.2). One of the interesting example of direct delivery is the *Infostation* architecture proposed by [FBY00] that uses direct contact delivery between mobile nodes and fixed gateways as a technique for increasing wireless network throughput and decreasing cost. Grossglauser and Tse showed that in their mobile ad hoc network scenario, direct communication between sources and destinations alone cannot achieve high throughput, because they are too far apart most of the time [GT02].

Algorithm	Knowledge	Processing	Storage	Communication	Performance
Direct Delivery	+	+	+	+	+
Flooding	+	+	+++	+++	+++
First Contact	+	+	+	+++	+
Simple Replication	+	+	++	+	++
History-based Replication	++	++	++	+	++
History-based Erasure Coding	++	+	++	++	++
Estimation-based Erasure Coding	++	+++	++	++	+++
Mobile Vehicle	++	++	++	+	+
MaxProp	++	+++	+	+	+++
Perfect Oracle	+++	+++	+	+++	+++
Imperfect Oracle	++	+++	+	++	+
Directed Flooding	+	+	++	+++	+++
Nile	++	++	++	++	+++

Table 5.2: Algorithm Characterization

First contact routing. Messages in this scheme follow a seemingly random path determined by a hot-potato algorithm. The next hop is chosen randomly from the available neighbors, if any are available. Otherwise, it is handed off to the node coming into proximity first [JDPF05]. This phenomenon can cause the message to hop among a group of two or more peers for a long time until the one having received the message leaves the group. To reduce this overhead, our simulation prevents returning the message to one of the previous 10% of the hops the message has travelled. The choice of a next hop does not try to make progress towards the destination; therefore, messages may aimlessly propagate through the network.

5.5.1 Two-hop relay

In this category, the source copies the message to the first n nodes that it contacts. The source and the relays hold the message and deliver it to the destination. Since there are now $n + 1$ copies of the message in the network, more bandwidth and storage are consumed. However, the resource consumption is limited and can be tuned by adjusting the number of copies. This strategy has a much better chance of delivering the message than the *direct contact* strategy. If we assume that each node contacts the destination with an independent probability p, then this strategy will deliver each message with probability $1-(1-p)^{(n+1)}$, which is approximately $(n+1)p$ if p is very small. Similarly, increasing the number of copies decreases the average latency, since the message is delivered as soon as any of the $n + 1$ nodes contacts the destination. This strategy has the same fundamental limitation as *direct contact*: If the $n + 1$ nodes never reach the destination, the message cannot be delivered. In scenarios where the mobility is random, this might be rare, but in networks with structured connectivity this could be very common.

Simple replication. This is a simple replication strategy in which identical copies of the message are sent over to the first r contacts, with r known as the replication factor. Only the source sends multiple copies of the message, the relay nodes are allowed to send them only to the destination; they cannot forward them to another relay. This makes it a mixture between direct delivery and flooding [WJMF05]. This algorithm has medium consumption of bandwidth and storage.

History-based simple replication. In this technique, the source creates r identical copies of a message, which are then delivered to the "best" r nodes, where quality is determined by history. The intermediate nodes will then each perform *direct delivery*. Our simulation follows the ZebraNet model of relying on the frequency at which a node has encountered the destination [WJMF05, JDPF05].

History-based erasure coding. This mechanism works very similar to history-based simple replication, but kr fragments totaling r times the message size are generated and sent to the best kr intermediate nodes. The intermediate nodes will deliver only to the final destination, where any k fragments can reconstruct the message. This has the same performance as *simple replication* when the path failure model is Bernoulli and the contact volume is sufficient for an entire message [WJMF05].

5.5.2 Multi-hop relay

Flooding/Epidemic routing. The most effective DTN routing protocol is flooding or epidemic flooding. In this scheme, messages are simply copied to any node that is reachable and does not already have a copy of the message. Each node forwards all the non-duplicated messages (including messages received on behalf of other nodes) to any other node that it encounters. As new nodes become reachable due to mobility or other reasons, additional copies are made. Flooding has the potential to deliver messages with the minimum delay if there are no resource constraints, such as link bandwidth or node storage [WJMF05, VB00]. The normal perception of such protocol is that they are generally deemed to be too expensive for practical use, although, they have been used for small networks [JOW+02]. In our implementation, flooding avoids transmitting a message to a devices that already has a copy (shown by non-arrowed links in Fig. 3.3(b))using the ihave/sendme model [KL86].

Directed flooding We have been inspired by flooding to design several *controlled flooding* techniques and directed flooding is among the first. As the name suggest, it floods or forwards the replica of messages toward only the destination. This means, it has to know from where the message came and in which direction is the destination. As the geographical location of devices is meaningless in opportunistic networks, directed flooding expects devices to advertise those devices that they and their contacts have encountered. This sharing of information is transitive in nature, therefore a message will be forwarded only to those peers that have a direct or indirect access to the final destination of the message. Provided an un-partitioned opportunistic network and appropriate time period, a device will have the knowledge about every other device in the network.

As described in Alg.1, a device inquire from all its neighbor if anyone has the access to the destination of the given message. All those neighbors that have a record of the destination and do not currently hold the replica of message, are then considered candidates as next hop for the replica of the message.

Input: Set P of all directly connected neighbors at any node X, Destination D_M of Message M, Ratio r_M between Primary and Secondary path list for M
Output: A set $P_M \subset P$ i.e. set of next hops that have accessibility to D_M
foreach *Node* $N \in P$ **do**
  if D_m *is accessible via* p *and* N *does not have* M **then**
    Add N to P_M;
  end
end

Algorithm 1: Directed flooding.

5.5.3 Gradient based

An alternate approach is to assign a weight to each node that represents its suitability to deliver messages to a given destination. When the custodian of a message contacts another node that has a better metric for the message's destination, it passes the message to this node. This approach is called gradient routing because the message follows a gradient of improving utility function values toward the destination. This requires more network knowledge than location-based routing for two reasons. First, each node must store a metric for all potential destinations. Second, sufficient information must be propagated through the network to allow each node to compute its metric for all destinations. One of the shortcomings of gradient routing is that it can initially take a long time for a good custodian to be found, since it may take some time for the utility function values to propagate, or because the metric values in the region around the initial custodian are all equally poor.

One interesting problem the delay brings to routing protocols other then improving their performance in gradient based methods, is how strictly we must follow the path provided by the routing protocol. If the source based routing scenario and source computes a path (S,B,C,D) to destination D, and on the way to destination, while the message is at B, the destination comes into direct contact with device B, then it makes perfect sense to deliver the message directly to D. However, the question whether the source S encounters C before it encounters B. Thus, we have implemented a loose path following the method shown in Fig. 5.7 where a attempt is made to deliver the message to any hop that is available in the path. This feature, on one hand improves the performance of routing algorithms, on the other hand it creates some issues for our max-flow analysis that we will discuss in chapter 9.

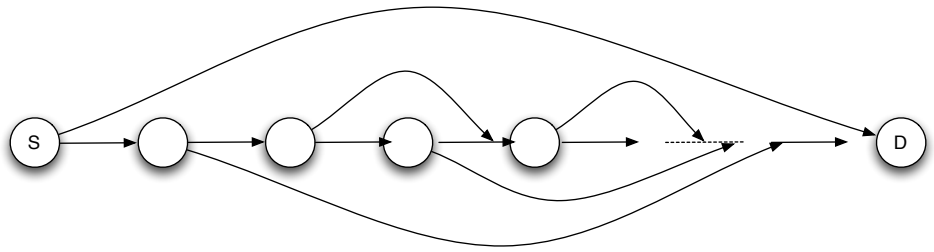


Figure 5.7: Propagation method [WH01]

Estimation-based erasure coding *EBEC*. History-based erasure coding is an all-or-nothing function: The nodes with highest probability receive all the data and the path length is limited to two hops. EBEC [LTZG06] is more adaptive, as the two communicating intermediate nodes exchange data until the number of fragments for a given destination is proportional to the nodes' probability of meeting the destination. To accelerate the simulation, history is calculated at intervals of 5 minutes and history oracle is used.

MaxProp routing. The Maxprop and RAPID protocols have been used in the context of DieselNet, a network of buses in Amherst. These protocols rely on distributed algorithms to optimize delivery using constrained replication in the face of limited storage and bandwidth resources. Unlike the networks we typically see in developing regions, DieselNet approaches assume a fairly random connectivity graph, and the protocols reflect this assumption in their structure. MaxProp attempts to forward the message to any device that has the greater probability to deliver the message to its destination. MaxProp involves calculating the path for each message at each transfer opportunity using a modified Dijkstra algorithm with history as pivotal criterion. MaxProp defines its own way of computing history to dictate the path computation. Moreover, it is assumed that topology information does not consume bandwidth.

MaxProp is the only method that incorporates a fancy mechanism of message queuing at peer level. It prefers the newly born messages and degrades the priority of messages based on their delivery probability and the number of hops they have traveled [BGJL06]. It uses a heuristic, which in case of bandwidth deficient environment, prefers to forward those messages that have not been able to travel farther from their respective sources. Even without the computational complexity of erasure coding, MaxProp is hungry for processing resources as the maintenance of the local queue is expensive for mobile devices under high message counts.

Earliest delivery a.k.a. Perfect oracle. Jain et al. [JFP04] present a variety of different strategies for networks, each of which require different amounts of information as shown in Fig. 3.4. However, all of the strategies assume that the contact schedule is precise. The first strategy, called *minimum expected delay* (MED), requires the least amount of information provided by contact summary. This oracle can answer questions about aggregate statistics like average waiting time of the contacts. When we insert additional information equivalent to knowing the time-varying DTN multi-graph provided by contact oracle, we obtain Earliest Delivery that we renamed as *perfect oracle*. This oracle can also correctly answer any question regarding contacts between two nodes at any point in time, which however, is only possible where the contact schedule is already known, i.e., unrealistic for real world scenario.

Shortest paths can be computed using Dijkstra's shortest path algorithm when edge costs are time-invariant. However, if the costs are changing with time the straightforward approach does not work. We must make two modifications to overcome this problem. First, the time a message will arrive at a particular node must be predicted. Second, the predicted arrival time must be used to determine the cost of taking subsequent edges. This would in turn, affect the time the message arrives at neighboring nodes. Interestingly, Dijkstra's algorithm can be adapted to compute the shortest paths for this case. This modified Dijkstra's algorithm is presented here as Alg. 2.

Jain et al. [JFP04] have also proposed more advance version of *earliest delivery* which include knowledge of the local queue, or using an even less realistic oracle, a global overview of all messages in the system at any given time in the future. With local queue knowledge (as in our simulation), a new path is generated when a node realizes that the message has been unable to reach the next hop in time. These strategies can assist to route around congested nodes either due to buffer limitations at any node at any time and traffic demand on the system. This results in the consumption of excessive processing power if the bandwidth is low or the path is a repetitive failure (e.g., for bottleneck paths). We have opted not to analyze the last two strategies as they are even more difficult to realize.

Imperfect oracle In analogy to the observations of Apostolopoulos, et al. [AGKT98] for frequent link QoS updates, frequent transmission of connectivity changes or history will severely reduce the network bandwidth, more so in a mobile environment where dynamics are high and bandwidth scarce. The simulator assumes that these topology exchanges happen out-of-bound, clearly unrealistic, especially for Earliest Delivery. We therefore wanted to examine the impact of small prediction errors on

Input: G = (V,E), src, T, w(e,t)
Output: L
$Q \leftarrow \{V\}$ L[src] $\leftarrow$ 0, L[v] $\leftarrow \infty \; \forall \; v \in V$ s.t $v \neq src$.
while $Q \neq \{\,\}$ **do**
  Let $u \in Q$ be the node s.t L[u] = $min_{x \in Q}$ L[x] ;
  Q = $Q \leftarrow \{u\}$;
  for *each edge* $e \in E$ *, s.t* e = *(u,v)* **do**
    if *L[v]* $>$ *(L[*u*] + w(e, L[*u*] + T))* **then**
      L[v] $\leftarrow$ L[u] + w(e, L[u] + T);
    end
  end
end

Algorithm 2: Dijkstra's Algorithm modified to use time-varying edge costs. s is the source and T is the start time. $L : V \rightarrow R$ is the array returning the cost of the shortest path for all nodes. The cost function w : E x $R^+ \rightarrow R^+$, gives the cost as a function of edge and time. The interpretation of $w(e,t)$ is the following: Let e be an edge from node u to node v. Given a message at u at time t, $w(e, t)$ is the cost (delay) of sending it to v. Therefore, if e is taken the message will reach v at time t + $w(e, t)$. The algorithm also works if the network topology is a multigraph [JFP04].

the routing performance. To do this, we created two versions of *earliest delivery*: As described above, *perfect oracle*, which corresponds to the scheme presented by Jain et al. [JFP04], and Imperfect Oracle, described here. To bring Perfect Oracle at par with real world scenarios, we have created an imperfect contact oracle that shares most of the properties of contact oracle but its answers for predicting the future are only perfect in $1/3$ of the cases. We have introduced 2 different types of weak errors in the contact oracle, each responsible for modifying the predictions of $1/3$ of the devices through the following mechanism.

Mistiming: Assume that the start and end times of contacts may be off. This is done by randomly moving the start or end point while maintaining the middle of the active period as active and the middle of the idle period as idle. This ensures that two devices will still meet, but maybe somewhat earlier, later, longer, or shorter.

Systematic errors: Exchange the timelines of two similar devices, namely the ones that have seen each other the most number of times. This is comparable to someone changing habits and, while still a weak modification, stronger than mistiming.

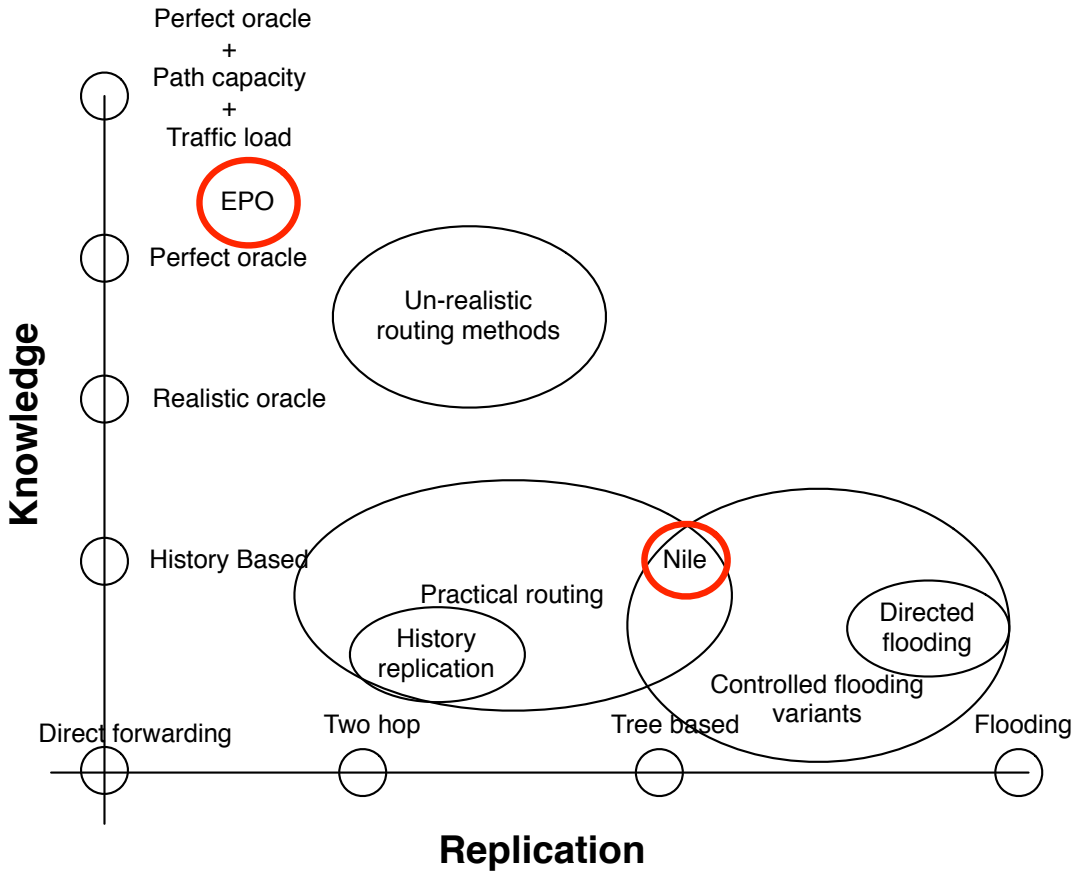


Figure 5.8: Protocol Taxonomy Adapted from [JW05]

In this way we have tried to introduce the alterations that are in line with the system and may be somewhat closer to what a realistic contact oracle can achieve. Here again, whenever the system realizes that a message has failed to take a hop or the hop was not available at the predicted time, then a new route to destination according to imperfect oracle is computed.

Table 5.2 summarizes the relative performance. For storage, note that we assume the sum of message sizes in the network dominates the amount of node information.

5.6 Protocol taxonomy

We have discussed several routing protocols in the previous section belonging to different overlapping categories. Based on their characteristics, it is not easy to

establish clear boundaries among the protocols. We still have placed them in different classes to highlight their properties. As shown in Fig. 5.8, we have plotted routing protocols based on the information they require to route the messages and the replication extent. The simplest of all is *direct delivery*, which needs to know only the destination and involves no replication to deliver the message. Along *X axis*, we have all the replication based classes where flooding is the extreme case that tries to exploit replication to an extreme extent with any knowledge about the network structure. On *Y axis*, we have the different knowledge levels that a protocol may enjoy for routing. We have set the extreme knowledgeable limit with an oracle that has not only contact and path capacity information, but also the knowledge about traffic currently present in the network.

We expect a realistic protocol to use history knowledge coupled with multi-path routing, forwarding multiple replicas of messages along distinct paths to destination. Our novel protocol *Nile* presented in Chapter 6 justifies its place in the structure as it only depends on history information about contacts and traffic, and then deploys multi-path routing on disjoint paths to increase the delivery probability. Moreover, *flooding* is considered to be a universal upper bound; hence, it is used mostly as a benchmark for routing performance. Instead of replication based upper bound, we argue in favor of a knowledge base upper bound known as *EPO* in Chapter 8.

Chapter 6

An adapting opportunistic network protocol-*Nile*

> "There's a difference between knowing the path and walking the path."
> Morpheus [Laurence Fishburne], The Matrix (1999)

This chapter derives the conclusion based on the behavior of routing protocols and then describes the measures that we have taken to set up the effects of these conclusion to construct our proposed solution. *Nile* redefines the path metrics in such a way that it considers not only device contact frequency but also embeds the path delay along with it. *Nile* differentiates between knowing the path and delivering a message by employing such link dependent traffic parameters that play an important role in associating a reliability measure to the paths. Moreover, *Nile* uses a very simple methods to identify time dependent clusters to control the replication of the messages.

6.1 Issues involved

During our experimentation, we analyzed the performance on 3 different kind of networks given different data transmission rates (discussed in Section 5). Single copy protocols inherently depend on the information from the network, and are not able to perform unless the required information flows throughout the network. This information is shared very rapidly and frequently in the case of strongly connected networks, which helps these protocols to out-perform others. *Flooding*, in the case of strongly connected networks suffers very strongly from the bandwidth shortage

as every node tries to send every byte it has.

6.1.1 Selecting the suitable path metrics

At one instant in time, a set of two-way intermittent links may connect one device to another in opportunist networks. This represents an opportunity for a device to send the data to the other end of the path. In the case of a any protocol, metrics such as throughput of the path given by the protocol, changes due to intermittency, mobility, failures, and traffic overload play a an important. The fact that there is a possibility of several paths being joint with each other, makes it difficult to select the suitable metric that can classify paths in a meaningful manner. Therefore, it is important to understand that the components of time one message has to wait for in order to reach from source to destination.

1. **Waiting time:** The waiting time is the amount of time a message must wait between the time it arrives at a node and the time connection to the next hop becomes available. This depends strictly on the contact schedule and the message arrival time.
2. **Queuing time:** The queuing time is the time it takes to drain the queue of messages with higher priority, which must be delivered over this link first. This depends primarily on the local message queue length of a node whereby a message passing through a congested path will incur very heavy delays.
3. **Transmission delay:** The transmission delay is the time it takes for all the bits of the message to be transmitted, which can be computed from the data rate and the message length.
4. **Propagation delay:** The propagation delay is the time it takes a bit to propagate across the connection/medium, which depends on the link technology.

The most significant of these factors is likely to be the waiting time, since it might range anywhere from seconds to days in contrast to others that are likely to be much shorter. We must consider the first two delay factors as the rest are device dependent. A routing protocol must try to minimize the *waiting time* by making precise predictions about the arrival of the next contact. The *queuing time* becomes significant in high traffic volume scenarios. When the amount of bytes grows larger than the throughput of the path, queuing time at the bottleneck link may grow exponentially high. Moreover, the fact that one path may consist of

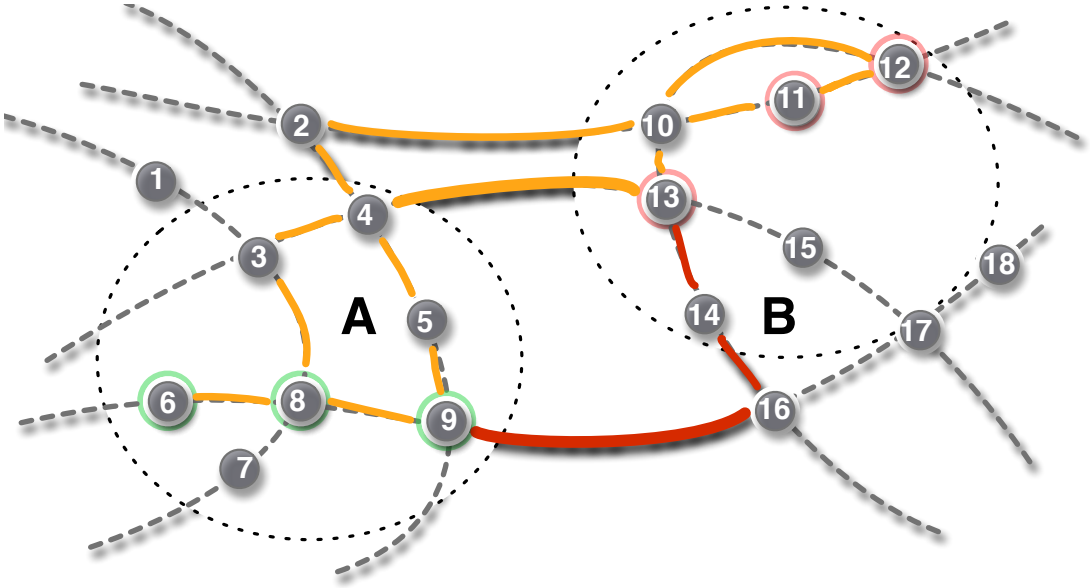


Figure 6.1: Congestion-adaptive routing takes advantage of dispersion

devices that have low waiting but high queuing times, makes it a big challenge to associate paths with metrics that can rank given paths in a realistic fashion.

6.1.2 Adaptable path selection

Another issue involving clustered opportunistic networks is that traffic between two clusters is always routed through the set of minimum delay links between the clusters. As long as the volume of the messages is less than the threshold that such an inter-cluster link can transfer, this mechanism works fine. As depicted in Fig 6.1, when this limit is crossed all the messages being routed through this link can get choked due to lack of bandwidth and local limited storage. Here, source nodes depicted in green (6,8,9) are trying to find a path to destination nodes depicted in red (13,11,12). Assuming the link (9,16) being the most frequently occurring link between cluster A and B, every node in cluster A may select it for routing all the traffic destined for cluster B, creating congestion at this link (shown by thickness of the edge (9,16)).

Earlier approaches do not not utilize the secondary path, because they use a metrics that have no notion of current traffic volume to rank the paths. Message switching is one of the simple solutions where large messages can be fragmented and routed through distributed paths towards the destination. The appropriate

fragmentation factor can be chosen at any middle hop depending on the congestion it is facing at the current point in time where performance of such a solution is highly dependent on the throughput of the nodes. *Nile*, on the other hand, deploys a different strategy where paths are ranked not only according to delays they may incur, but also current traffic volume is transferred through them. Initially, all the paths will be prioritized according to time-delay but as the time progresses, paths with high traffic volume loose their priority. Consequently, they give an opportunity to other paths to come into service that otherwise do not necessarily have the minimum delay. Such a mechanism plays a very important role in the case of opportunistic mobile networks, where not only bandwidth is very scarce, but also mobility causes unpredictability. Pictorially in Fig 6.1, when edge(9,16) has been carrying data to its limits, node 9 preferably lowers the ranking of this link and routes the messages through the next available path containing links (4,13),(2,10). This way the load in the network is balanced thus increasing the messages delivery probability.

Reaction from traces

It is not a trivial tasks to visualize opportunistic networks because of time dependency that makes them dynamic graphs. Although, we are prone to loose information in the process of summarizing the dynamic graphs into a static graph, the motivation is to proceed with this reduction in this way so that we are able to build some heuristics that may help us to make good routing decisions. As a real example, in Fig 6.2 we have shown a snippet for the flow of traffic for MaxProp (described in section 5.5.3), in the IBM trace, which we have created with the help of Visone [BS08].

The way different colored nodes and edges are presented in Fig. 6.2 and 6.4 can be interpreted as shown in Fig. 6.3. Nodes represented in these graphs are source, destination, source and destination simultaneously, and mules that are neither, source or destination. Thus, they merely play a role to bring the message closer to the destination. The size of the node represent the sum of bytes received and bytes sent by that node. The links between the nodes are also color specified where each line represent a different activity between two nodes. Black line represents message transmission attempts, blue line represents messages successfully transmitted, green represents that surplus bandwidth has been available during the time of simulation, and red represent a bandwidth deficient link. The weight of the lines represent the magnitude of each factor where as the dotted line and the solid line represent the high and low bandwidth activities respectively.

In Fig 6.2, we have tracked the device activity on both high and low bandwidth

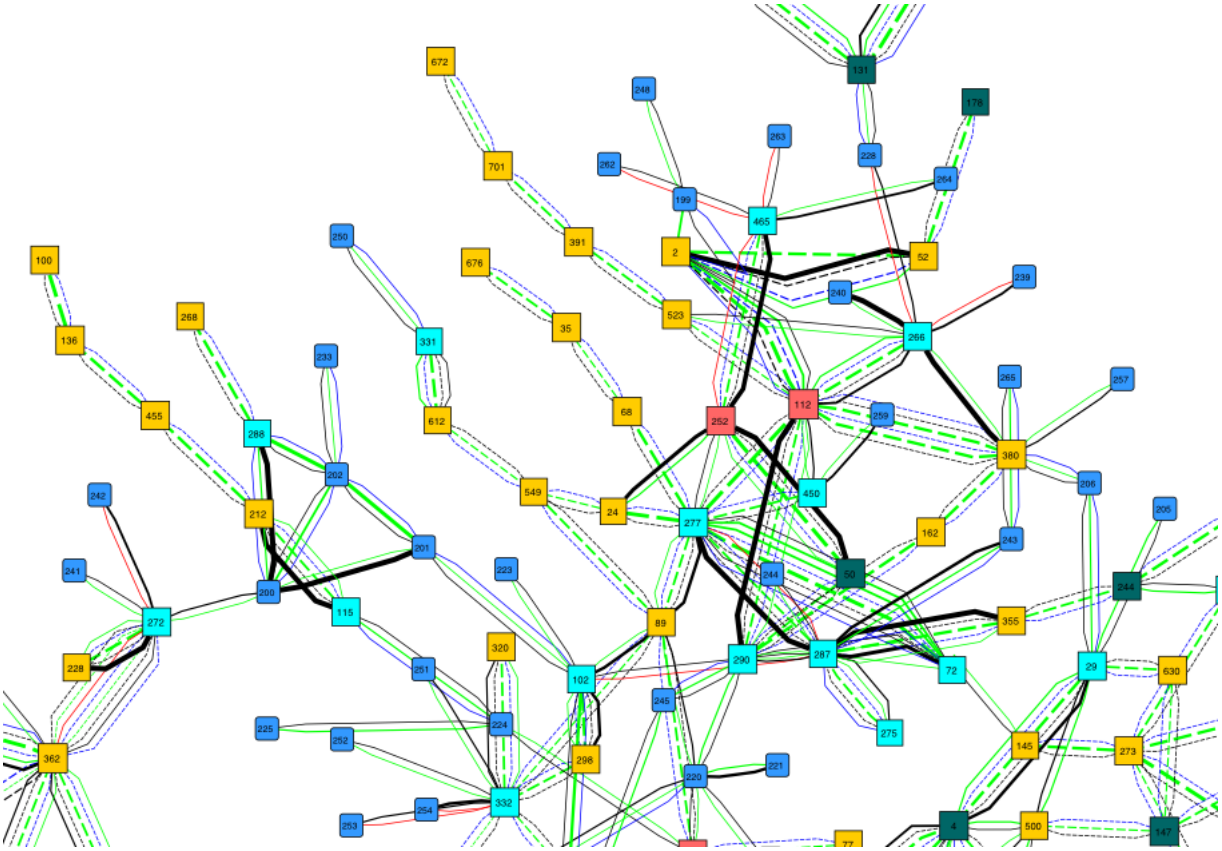


Figure 6.2: IBM network snippet (MaxProp)

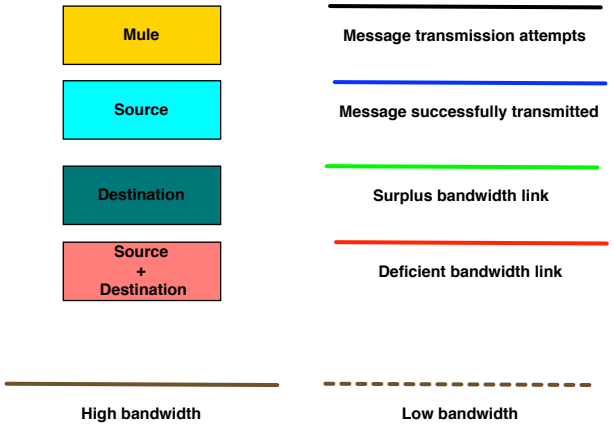


Figure 6.3: Color key for Fig. 6.2 and 6.4

scenario and merged the results to see the difference of IBM network behavior in the two scenarios. We see that dotted lines go far into the boundaries of the network where as solid lines get trapped half way in the middle where they encounter a bottleneck. We also observe that there is no thick black dotted line showing that all the message transmission attempts have been successful in high bandwidth. On

the other hand, we can see several thick solid black lines, e.g. (112,290), (252,465), (287,277) in cluster situations representing that a link has been attempted multiple times to transmit a particular message. If we concentrate on the size of the nodes, we see that most of the nodes with thicker solid lines have relatively larger sizes than others. These bigger nodes not only include source and destination but also data mules, e.g. 380, 355.

In Fig. 6.4, we have presented the MIT network behavior for history based replication (described in section 5.5.1) in a high bandwidth scenario. As the MIT is a relatively dense network, we have removed all the links except those representing message transfer attempts (black line), so that the extracted graph can be easily understandable. First of all, we can observe that the size of all nodes is more or less the same because 100 messages have been distributed among 81 nodes and the majority of nodes are designated as source as well as destination. We can see that nodes with thicker edges have relatively high degree of edges compared to the rest of the nodes. We find two nodes to be prominent, i.e. 24, 43 as both of them have the degree of 6 and 7 respectively and most of the edges are thicker denoting multiple message transmission attempts. We should keep in mind that this figure represents the MIT network behavior in high bandwidth and that there were no green lines in the graph, which means that a high bandwidth network can also behave like a bandwidth deficient network if the contact durations are small and volatile.

6.1.3 Dynamic replication strategy

As already stated, single copy protocols have to be unrealistically intelligent to give acceptable outcome in sparse networks. In our opinion, a protocol may replicate aggressively in a sparse but resource-rich network and at the same time can restrict the replication in a dense network where replication will not be beneficial. It is not possible for a node to have a global picture of the opportunistic mobile network (as several oracles have), rather a node may only have the local neighborhood information such as the current node density, local congestion, replicas present for a particular message, and so on. Moreover, there are many pivotal and unexplored aspects of opportunistic networks that replication-based algorithms can exploit to optimize the performance.

As discussed earlier, it is imperative to replicate messages to obtain acceptable results in sparse networks, but it is also important to control congestion in dense environments thus this raises the question of the replication factor. In our opinion, it is desirable to keep the replication approach very flexible and dynamic, i.e. the

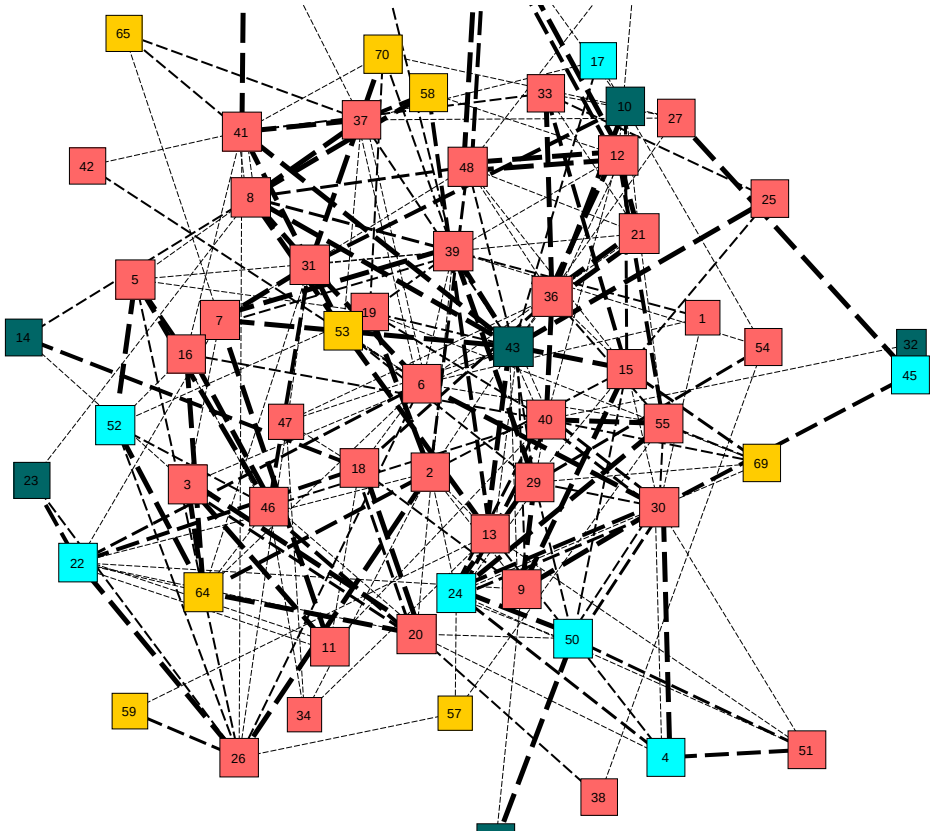


Figure 6.4: MIT network snippet (history based replication)

extent of replication should not be fixed by the source at the time the message is born. The nodes that act as middle hops between source and destination can replicate the message depending on the local traffic conditions. The node that is at the centre of a cluster must refrain from generating extra replica as compared to a node which has very few direct contact, i.e. at the boundary of the network. As shown in Fig 6.5, an adaptive protocol may select those paths that have a low delay as well as those that experience low congestion. If such a *suitable* path is not available, the protocol will tolerate delay of the path that is experiencing low congestion. Congestion can also be controlled by prioritizing the messages depending on the TTL, proximity to the destination or message sizes. In our opinion, messages having low TTL, smaller size, or are closer to destination, deserve the priority over their counterparts.

Identifying all these factors helps us to perceive building blocks of a realistic, efficient and practical protocol. It is now evident that a desired protocol should

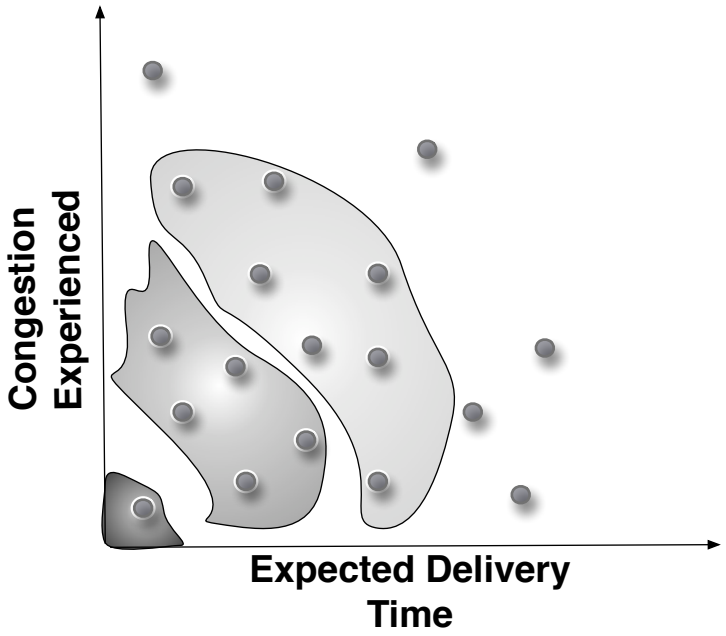


Figure 6.5: Route selection methodology

try to minimize all these delay components that a message may encounter to ensure speedy delivery of messages through an opportunistic network. The aim now is to figure out the information needed by the protocol to select paths based on this criteria. As we will see in section 7.3, many of the existing protocols have failed to identify these factors and they have naively considered several variations of encounter frequencies of the nodes. Motivated by these observations, we find it necessary to have a sound compromise between *flooding* and intelligent techniques, borrowing the strong points of both the methodologies.

6.1.4 Suppressing jittery routing

All the routing schemes that use some kind of utility function minimization to select among the suitable paths, may suffer from *jittery routing* that consumes a considerable amount of bandwidth. We consider a scenario as described in Fig. 6.6 where a source has to choose among 4 neighbors as the next potential hop and if all of them have the similar accessibility to the destination, the difference between the outcome of utility function for all of them will be minute. Whenever, anyone of the candidates comes in contact with the source, it may gain priority over others due to tiny improvements in the path metric and the source will attempt to propagate the message to this hop. If the message suffers from a failed transmission due to congestion, then the source will attempt to propagate the message to the device it comes into contact with afterwards.

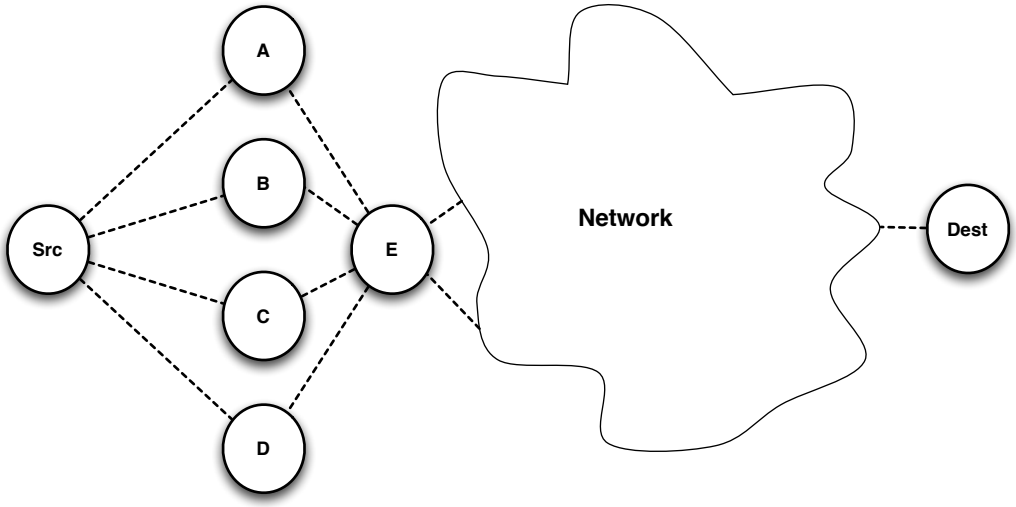


Figure 6.6: Route selection methodology

In our experiments, some of the protocols have shown that this jitter routing behavior causes considerable bandwidth consumption, when the failed transmission is repeated several times among different next hops. This pushes the congestion problem from bad to worse for all the messages that have partially overlapped paths with these devices. This problem can be solved by setting a threshold on the path metric such that all the updates below this threshold can be ignored and only when a path presents a bigger change in the path metric, it can be considered for propagation.

6.1.5 Avoiding reverse routing

One of the issues involved with multi-path routing is that fragments/replicas of different messages destined for the same destination may end up taking opposite direction paths. *Estimation based erasure coding* is one example in our experiments that shows such a behavior in sparse networks. This method propagates multiple erasure coded fragments on multiple paths depending on their accessibility to the destination. We assume that the device A in Fig. 6.6 has advertised the best path metric and it holds maximum fragments (A fragments) of a message while device B is carrying the rest (B fragments) of them. After a while, when these fragments have assembled at device E, there is no provision hindering the propagation of B fragments back to device A. This phenomena occurs due to both, the sensitive utility function and the sparsity of network.

We believe that the designers of the protocol in question have not been able to see this behavior as they have been using an artificial simulation scenario with

a prescribed bounded area and randomly moving devices. Such a setup create frequently occurring contacts and minimizes the probability of fragments taking a reverse path.

6.1.6 Using suitable local queuing

Out of all the various routing technique presented in section 5.5, MaxProp is the only example that has touched the issue of local message queues [BGJL06]. MaxProp uses a dynamic threshold related to the number of bytes transferred during each transfer opportunity say x and size of the local queue in bytes. If x is smaller than the queue size, then messages with the low hop count are preferred to be propagated to the next hop. The reason as explained by the authors, is to give a delivery chance to those packets that have a low probability to be delivered as the protocol favors high delivery probability packets. If the difference between x and queue size decreases, this priority mechanism is gradually ignored as in the end, it will have no effect if x is almost equal to the local buffer size.

We investigated the effect of this local queuing strategy and analyzed the results of MaxProp with and without local queueing. Our experiments showed that this mechanism does play a role and improves the performance of MaxProp up to 10%. This improvement is not very significant but convinced us to look into the matter of local queueing. We then experimented with several other methods and found that local queueing is not the pivotal factor in opportunistic routing. The only point that makes a difference is to give priority to those messages that are destined for the next hop, otherwise, as we will show in section 7.3.3, giving preference to one message over the other does not really increases the delivery chances.

When a device prefers one bigger message over the others, that message may propagate a little further in the network but it may also create a congestion at a later time and thus, it may reduce the delivery probability of those messages that have a joint path with it. Giving preference to small messages also will not work, unless they have a long enough/sufficient residue life time.

6.2 Nile

Nile is basically a proactive routing protocol where each network node maintains a delay augmented routing table that in turn is used to control the replica propagation of the messages. *Nile* constructs these routing tables throughout the network using direct encounters to disseminate the contact information (a similar idea is presented by Marina and Das [MD01]). *Nile* can be seen as a modified distance

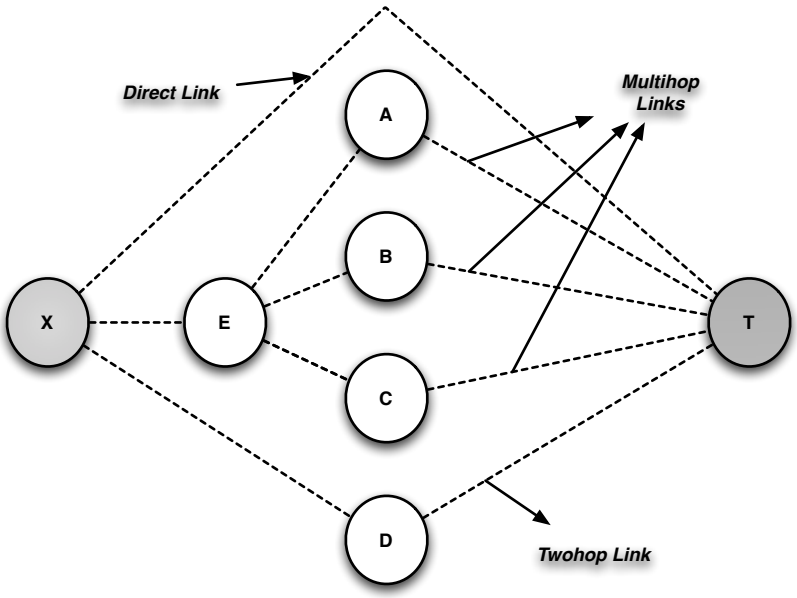


Figure 6.7: Tuple sharing among peers

vector protocol, where at every node encounter, each node advertises a 3 tuple (details given below) list of nodes that it has logged during the previous encounters. This information is accumulated in a routing table of each node and this table is referred to whenever a routing decision is to be made. This implies that the mobility of nodes plays a very important role in the performance of *Nile*, as mobility ensures a precise and accurate local information is shared for each node, that closely represents the true picture of the network.

6.2.1 Algorithm description

For each encounter between two directly connected nodes X and T as shown in Fig 6.7 as *Direct Link*, The routing table is populated as follows.

a. Node ID of T if it is a first contact between X and T.

b. Update or insert expected delay ϵ of message transmission from X to T, ϵ_{XT}, includes the duration messages stay in the local queues as well as inter-hop transmission times.

c. Update or insert link congestion η_{XT}. It is the ratio between the average number of bytes that are actually delivered b_{XT} and potentially could be delivered b^v_{XT} i.e.

$$\eta_{XT} = b_{XT}/b^{v}_{XT}$$

where $\eta_{XT} = 1$ indicates that the path is congested.

d. Timestamp. Update the timestamp to current contact time.

For an indirect neighbor, as shown in Fig 6.7, *Twohop Link* device D will advertise its minimum delay path to device X. The augmented information in X's routing table about the device T accessible via D looks like:

a. Node ID D if X has no earlier record of D via X.

b. Update or insert expected delay from X to T,

$$\epsilon_{XT} = \epsilon_{XD} + \epsilon_{DT}$$

c. Update or insert link congestion on the path X, D, T

$$\eta_{XT} = \begin{cases} max(\eta_{XD}, \eta_{DT}) & \text{if } \eta_{DT} < 1,\ \eta_{XD} < 1 \\ \eta_{XD} \times \eta_{DT} & otherwise \end{cases}$$

d. Timestamp. Update the timestamp to current contact time.

In an indirect neighbor scenario, if D has multiple paths to T then it will advertise the only path that has the minimum delay. Intuitively, D should "somehow" advertise that it has multiple paths to T so that D receives preference over any other device that has a single similar delay path to T. The complication involved in this step are discusses in Section 11.1.2. We would like to mention here that although D has only advertised one path to T it still can propagate multiple replicas to T once it has acquired the message that may be destined to T.

Several works have argued in favor of using the recent history and discarding the old encounters [LDS03, CMMP08]. This cannot only save storage space on mobile devices but also reduce computation overhead. In this implementation of *Nile*, we have used the limit of 7 days, i.e., those records that are more than 7 days old are discarded.

If there are multiple paths to T than the path with the minimum value of ϵ_{YZ} with its corresponding attributes is stored. This ensures that a faster path (irrespective

of number of hops) receives the preference over a slower path that may have fewer hops. Any node X carrying a message M with destination D_M looks into the set of all direct peers denoted by P that have advertised the destination D_M through their direct or indirect contacts.

6.2.2 Congestion handling

The critical measure to keep congestion in check is to control the number of replicas a node creates. One simple measure to restrict this number is to push replica on disjoint paths. The idea is to propagate replicas on those non-overlapping paths that are not congested at the moment, in particular in dense clusters. To check whether two paths are disjoint, *Nile* employs *cosine measure similarity*. Literature shows that cosine measure similarity has been used to estimate the similarity between two documents [WH91]. We have adapted the problem at hand to use this measure in the following way.

For each next hop that has a path to destination, *Nile* forms vectors consisting of ϵ_{XZ} as elements where Z may be any node on the path. Formally, a vector for message M at node X is calculated as,

$$PVec_{D_M} = \{\epsilon_{XZ} | \epsilon_{XZ} \leq \epsilon_{XD_M}\}$$

where D_M is the destination. A next hop vector will be removed from the candidate list If the *CosVectSimilarity* is higher than $1 - \eta_{XD}$, where η_{XD} is the congestion indicator on the path. This implies, a congested path will only be placed in the candidate list, if it is highly disjoint compared to all the earlier paths selected. Conversely, a non-disjoint path may be selected only if it is not experiencing congestion.

It is not easy to avoid the question of complexity of computing cosine vector similarity. *Nile* can be seen as a modified Distance Vector protocol, where a node has to choose only among suitable direct neighbors. If X has n direct contacts that have accessibility to the destination, X will have n such vectors. For example, a node may have 10 paths to destination and only 3 direct neighbors, in which case it does not have to process 10 but only 3 vectors. So the complexity of cosine vector similarity does not depend on the number of paths but on the number of direct contacts through which the destination is accessible.

Graphically shown in Fig 6.5, *Nile* prefers the paths that are experiencing neither long delays nor bandwidth shortages. The first chosen candidate is the one that has the minimum delay among the available candidates. The next candidate will

Input: Set P of all directly connected neighbors at any node X, Destination D_M of Message M, Ratio r_M between Primary and Secondary path list for M
Output: A set $P_M \subset P$ i.e. set of reliable next hops for M
foreach *Node* $N \in P$ **do**
  if D_m *is accessible via* N **then**
    Add N to P_M;
    $PVec \leftarrow \epsilon_{ND_m} - \epsilon_{NX}$ where $\epsilon_{NX} < \epsilon_{ND_m}$
  end
end
Sort ascending P_M w.r.t. ϵ_{ND};
for $i = 0$ *to* $Size(P_M)$ **do**
  for $j = i + 1$ *to Size*(P_M) **do**
    if *CosVect* $(PVec[i], PVec[j]) > 1 - \eta_{XD}$ **then**
      remove j from P_M;
    end
  end
end
Divide P_M by r_M

Algorithm 3: Path selection for given message M.

be chosen when it either has a relatively disjoint path to destination or its path has a relatively low congestion thus keeping a reasonable mix of paths. The route selection algorithm is presented at Alg. 3 .

As discussed previously, *Nile* may select a congested path provided its disjoint(er) than others paths. To avoid the adverse affect of disjointness estimation, *Nile* further keeps on tuning the replication factor by maintaining *primary* and *secondary* candidate node lists for any message M. The ratio between the size of these two lists is represent by r_M in Alg. 3. A node will try to produce a replica for only those paths that are available in the primary list. If a node X encounters a secondary list node carrying the message M, X will reduce the size of its primary list. This way, *Nile* uses an abstract local node density criteria to reduce the replication factor. This scenario can be imagined as an overlay network of all the nodes that are on the route to a particular destination. *Nile* keeps on pruning this overlay network, provided the nodes are able to see the disagreements between their *primary* and *secondary* lists related to a message M. Whenever a message is able to be replicated in the undesired neighborhood region of a particular device, that device reduces its capability of replicating that message by a factor of f_M.

6.2.3 Feedback

Nile also shares the back trace knowledge about the delivered messages to reduce the network load . When a node X carrying message M either delivers the message itself to the destination D_M or encounters another node Y that has delivered the message to D_M, then it also kills its local replica of M. This way a message replica stuck in a bottleneck is removed from the propagation process if one of the bottleneck devices contacts a device that has witnessed the delivery of that replica to destination.

6.2.4 Local queuing

As mentioned earlier in section 5.5.3, we can find a few examples of device level message priorities in literature. After analyzing the performance of MaxProp [BGJL06], with and without queuing, we devised a multi-facet local queuing strategy for *Nile*. The messages are prioritized according to the following order for propagation to the next hop. When one criterion fails to give any result then the next criterion is chosen for selection.

1. The message destined for next hop.
2. The message that has less paths to destination. All the devices including the source, tag each message with the number of paths that the devices have selected for the message. If a middle hop has more options than the source for a particular message, the message is tagged with higher tag value.
3. The message with shorter remaining life time.

As each device is autonomous and makes its routing decision only on the basis of local network information, it is hard to guarantee the efficiency of such a priority system. Based on this criteria, a selection of messages by two different devices may end up selecting such messages whose paths coincide and whose volume will congest one of the links on that path.

Chapter 7

Routing simulation results

> "A work of art is the unique result of a unique temperament." Oscar Wilde

This chapter presents the results of all the methods that have been described in previous chapters for all the network traces. Results are presented according to the criteria established in section 7.1. As we have advocated the use of QoS measures to be part of routing algorithms, we have used three bandwidth configurations, i.e. high, medium and low, to verify the change in performance in such a variance. The rest of the simulation parameters are summarized in table 5.1. At one point we observed that that medium bandwidth had no unusual results and we explained our conclusion with the help of low and high bandwidth scenarios. We have therefore included these two variance in this section and they are discussed separately for each evaluation criteria.

Setup Every history based algorithm compared in this study has been provided first 10 trace days to constitute mature history as prescribed by the algorithm. In contrast, Nile not only requires contact information from network but also the congestion indicator that we have devised. For the purpose of providing a mature history of traffic as well as avoiding any unbiased advantage to Nile, we have gathered the metric related to traffic volume by simulating *directed flooding* for the period of the first 10 days of the trace times. *Directed flooding* is a predecessor and very crude version of Nile where the attempt is made to create a replica for every node that has access to the destination. We have chosen *directed flooding* so that all the possible paths with their corresponding delivery capabilities are

exposed. It is important to mention here that in the case of the IBM and MITBT traces, the directed flood simulation did not finish within the expected time as a few source devices appeared after 10 days for the first time in trace. Therefore, the history was prolonged more than 10 days due to such devices.

In the results presented in the following sections, we have neglected the overhead of gathering network topology information, giving an advantage to "wise" algorithms; this means that sharing of routing information does not consume bandwidth and device resources. Despite this biasness, we find that "ignorant" algorithms perform amazingly well.

7.1 Evaluation criteria

In order to compare routing strategies, we must define some metrics for evaluating their performance. Since the exact numbers for the metrics depend on many factors, we will only discuss them in relative terms.

- **Path latency:** As we identified the difference between finding a path and delivering a message, we also analyze the latency encountered by the best path computed by each of the protocols, assuming zero size messages. The difference between the path latency and message latency then proves our point that traffic volume cannot be ignored while routing in opportunistic networks.
- **Delivery ratio:** In an opportunistic network, the most important network performance metric is the delivery ratio, while in opportunistic networks a message is rarely actually "lost", rather, the network was unable to deliver messages within an acceptable amount of time. Thus, we define the delivery ratio as the fraction of generated messages that are correctly delivered to the final destination within a given time period.
- **Message latency:** A secondary metric is the latency, meaning the time between when a message is generated and when it is received. This metric is important since many applications can benefit from a short delivery latency, even though they will tolerate long waits. Many applications also have some time window where the data is useful. For example, if an opportunistic networks is used to deliver email to a mobile user, the messages must be delivered before the user moves out of the network.
- **Transmissions:** Some routing strategies transmit more messages than others, either because they use multiple copies of each message, make different

decisions about the next hop, or because of protocol overhead. The number of transmissions is a measure of the amount of contact capacity consumed by a protocol. It is also an approximate measure of the computational resources required, as there is some processing required for each message. Additionally, each transmission consumes energy; this is also an approximate measure of power consumption.

- **Overhead:** No performance analysis is complete without a look at the overheads encountered. We have defined two overhead measurements, i.e. device storage and bandwidth consumption. Device storage refers to local storage consumption, which each device had to tolerate during the course of the simulation. We have used the network storage consumption, i.e. commutative device storage to represent this metric. The other metric, we have used is bandwidth consumption, which represents the amount of bandwidth consumed in the network.

- **Network activity:** We have introduced this metric to observe the extent to which different protocols include network devices to participate while routing messages. This measure is further categorized into transmission attempts made, transmissions completed, and bytes transferred by different devices.

7.2 Path latency

As we identified the difference between finding a path and delivering a message, we also analyze the latency encountered by the best path, which is computed by each of the protocols, assuming very small size messages. The difference between the path latency and message latency then proves our point that traffic volume cannot be ignored while routing in opportunistic networks.

When we look at the latencies experienced by different algorithms for finding a path that reaches the destination in all three traces as shown in Fig 7.1, we can to some extent infer characteristics like density and mobility of given networks. Here we have simulated the algorithm with messages of one Byte and all replication based algorithms enjoy the presence of an oracle that informs every device immediately when the message is delivered, so that they can stop forwarding the rest of the replica of that message.

The results show that single copy algorithms are struggling very hard in the medium and highly sparse networks, i.e., access point (IBM) and bluetooth (MITBT). In the bluetooth (MITBT) case, the lower bound case of *direct delivery* has not

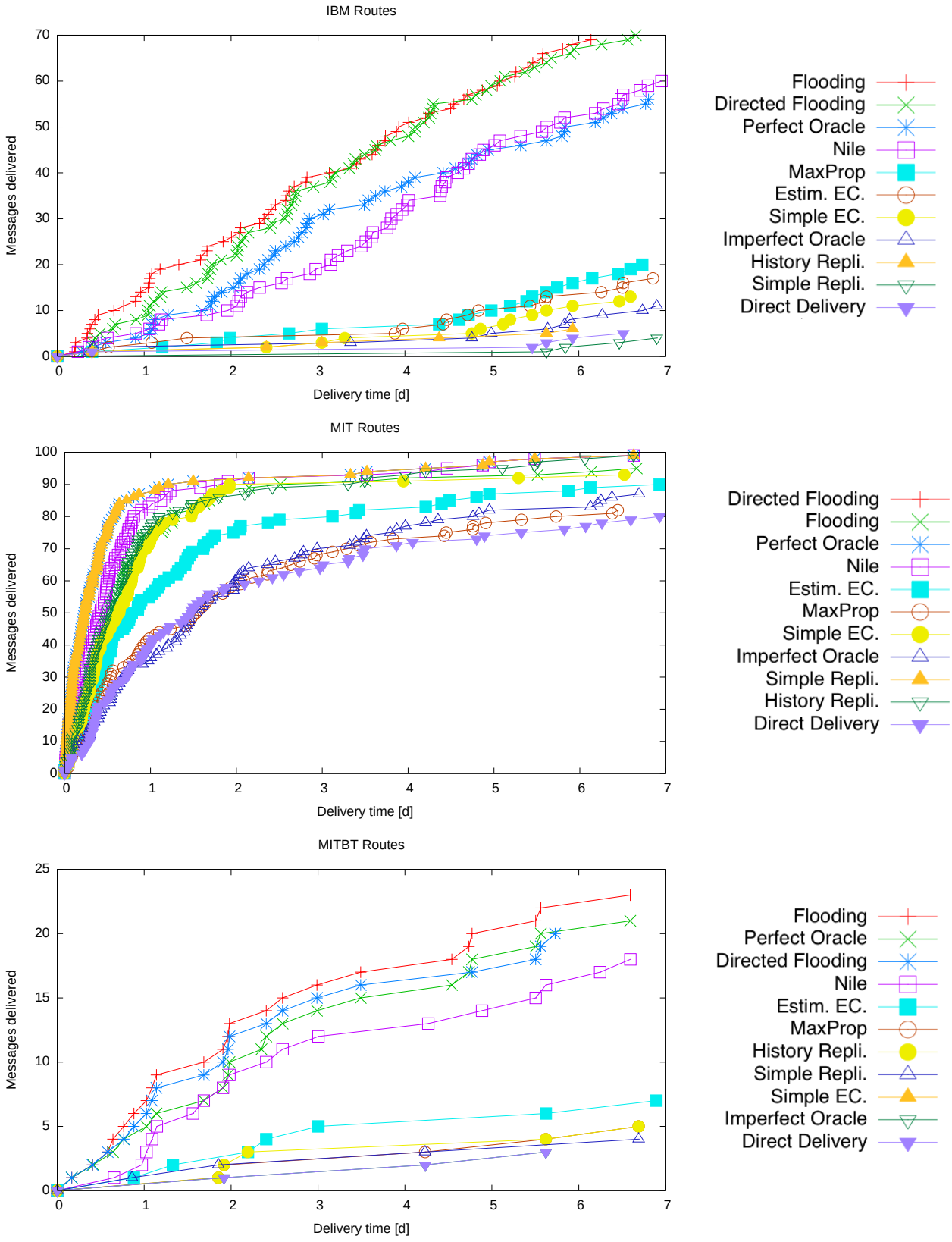


Figure 7.1: Latency encountered for reaching the destination

been able to find a path for a single message representing the high sparse nature of the network, whereas *MaxPropMaxProp* has been successful enough to gain access to destinations in 5 cases. *Flooding* variants and oracle based algorithms gained access to destination in approx. 20 cases.

In the dense scenario of MIT, *direct delivery* has found 60 paths in the span of 2 days. Moreover, the dense nature of the MIT trace is evident also by the fact that every other method has been faster and has obtained access to destination within one day for approx. 80 cases. Such a high destination access rate very clearly hints towards a network that is dense due to both a high communication range and a high mobility of devices.

Results from moderately sparse example of the access point (IBM) trace show that replication based techniques have access to approx. 70% of the cases but the time duration for this process has been relatively long. The non-replication methods are also somewhat successful, but their performance cannot be classified satisfactory. *Flooding* is the first to reach the destinations followed by *directed flooding* that is propagating messages only on those paths that have been stored by devices during the history duration. This means, *flooding* takes advantages of a few paths that did not exist in the history.

We have a curious case of perfect oracle that has a relatively slow access to destination. This can be explained as an artifact of our implementation. Our implementation of modified Dijkstra's algorithm assures that every contact has a duration of at least 5 seconds to become part of the path. We introduced this step to reduce the computation time of the algorithm, otherwise the computation was very slow particularly in the case of the cell tower (MIT) trace. As the MIT trace had a large number of small duration meetings, the computation time of modified Dijkstra's has been to the scale of a few days. We can assume that very short connections are likely from the users who are present at the edge of the reception range or from the users who are passing by swiftly. We therefore believe that resulting connection are unreliable. Moreover, we also had to push the simulation artificially by consuming 1 second with each message transmission. Without this step, 1 Byte messages cannot consume any contact duration and simulation did not proceed. *Flooding* did not suffer from this artifact because it can afford the loss of one contact opportunity as the another replica would avail the next contact opportunity. In contrast, *perfect oracle* had to re-route the message as soon as it had realized that a contact opportunity had been missed due these artifacts.

7.3 Message latency and delivery ratio

In an opportunistic network, the most important network performance metric is the delivery ratio. However, in opportunistic networks, a message is rarely actually "lost", rather, the network was unable to deliver messages within an acceptable amount of time. Thus, we define the delivery ratio as the fraction of generated messages correctly delivered to the final destination within a given time period. We combined delivery ratio with latency that represents the time between when a message actually being generated and when finally being delivered to the destination. This metric is important since many applications can benefit from a short delivery latency, even though they will tolerate long waits. Many applications also have some time window during which the data is useful. For example, if an opportunistic network is used to deliver email to a mobile user, the messages must be delivered before the user moves out of the network.

To verify the reliability of our calculations, we have made multiple runs for the cell tower (MIT) as well as the bluetooth (MITBT) traces in various time periods. As already discussed, we observed maximum activity in the month of November 2004 and after that in October 2004. We performed one run choosing November and one run choosing October as simulation period but we did not find any mentionable difference in the behavior of routing protocols. In the bluetooth (MITBT) case, the performance of all the algorithms was very ordinary due to lack of connectivity among the nodes and the size of the network. To obtain a meaningful output from bluetooth traces, we intensified the selection criteria of source and destination to the top 30% of online devices and as predicted some algorithms showed considerable performance with this setup. We have plotted 2 different kinds of graphs against time, i.e. number of messages and amount of data delivered.

Fig. 7.4, 7.2, 7.3 show the graphs of time vs. number of messages with high and low bandwidth respectively. These figures show how well messages are delivered for the three environments. For example, in the access point (IBM) and the bluetooth (MITBT) high bandwidth case (Fig. 7.4 and 7.5), we can observe that *perfect oracle* and *flooding* are dominant among all the algorithms and as the bandwidth decreases (Fig. 7.2,7.3), the performance degrades.

In the bluetooth (MITBT) case (Fig. 7.5 and 7.3) we can see that *perfect oracle* from early on starts delivering messages and after almost 6 days, it has delivered close to 50–120 MBytes, about 10%–25% respectively, of the total message load. The runner-up, *flooding*, starts a little bit slower, but after 2 days, it has caught up and will remain close to the winner. In the the access point (IBM) scenario

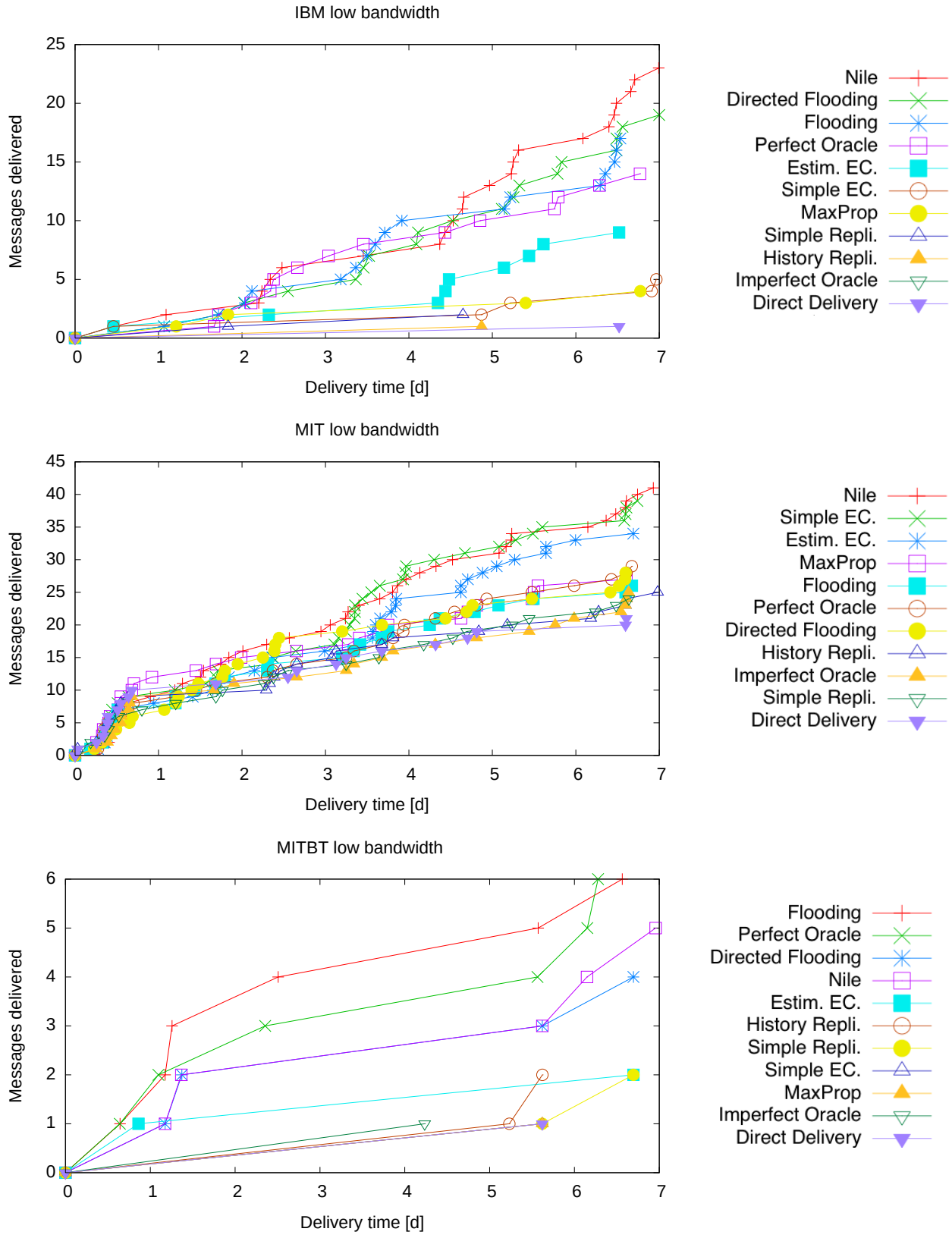


Figure 7.2: Number of messages delivered with low bandwidth from access point IBM(a), cell tower MIT (b) and bluetooth MITBT traces(c)

(Fig. 7.5), however, *flooding*, *direct delivery*, *imperfect oracle*, and *EBEC* all deliver their first big message 1.5 days after it was injected into the network; *Perfect oracle* requires about 4 times longer to deliver the same volume. In contrast to other protocols, this volume is composed of 4 smaller messages. In the meantime, its ugly stepbrother *imperfect oracle* has taken the performance lead for 1 day, but after 7 days, *flooding* has delivered roughly 25% of the data, collecting the Maillot Jaune.

In the cell tower (MIT) scenario (Fig. 7.4,7.5), high bandwidth plots show the strong connectivity of the network as *direct delivery* also shows considerable performance and its is not easily to rank the algorithms in this case. However, in the low bandwidth case (Fig. 7.2,7.3), things look very different again: *MaxProp* immediately leads the pack with a huge margin, having delivered about 90% of the total data in just 2.5 days. *EBEC* and *simple replication* start then to catch up, but remain without chances.

7.3.1 Reasoning

Why does this happen? Why does the Perfect Oracle behave so poorly despite its omniscience? The following paragraphs will answer these questions.

Omniscience is not everything. First of all, *perfect oracle* is *not* omniscient, it lacks knowledge about concurrent traffic so it cannot avoid bottlenecks. Even worse, it does not include the message size into the calculation, which can result in the choice of a path that does not provide long enough connection times to transmit the message even in the absence of other messages. The latter could be avoided, but not the former.[1] It also seems that the perfection (perfect oracle) is not promising, as the selection of the "best" path is predictable. Even in the cell tower case, connectivity seems to be sparse enough to create a few "attractive" bottleneck links, through which large portions of the traffic should be funneled. When this fails, the remaining messages need to be rerouted, probably again along similar routes, creating more bottlenecks.

This is where other algorithms, including *flooding* take control of the network. In a well-connected environment such as cell towers, *direct delivery* and its two-hop cousin *simple replication* perform well, the latter delivers about 80% of the data.

[1] It would probably require a high-speed ubiquitous wireless network for topology/traffic information exchange.

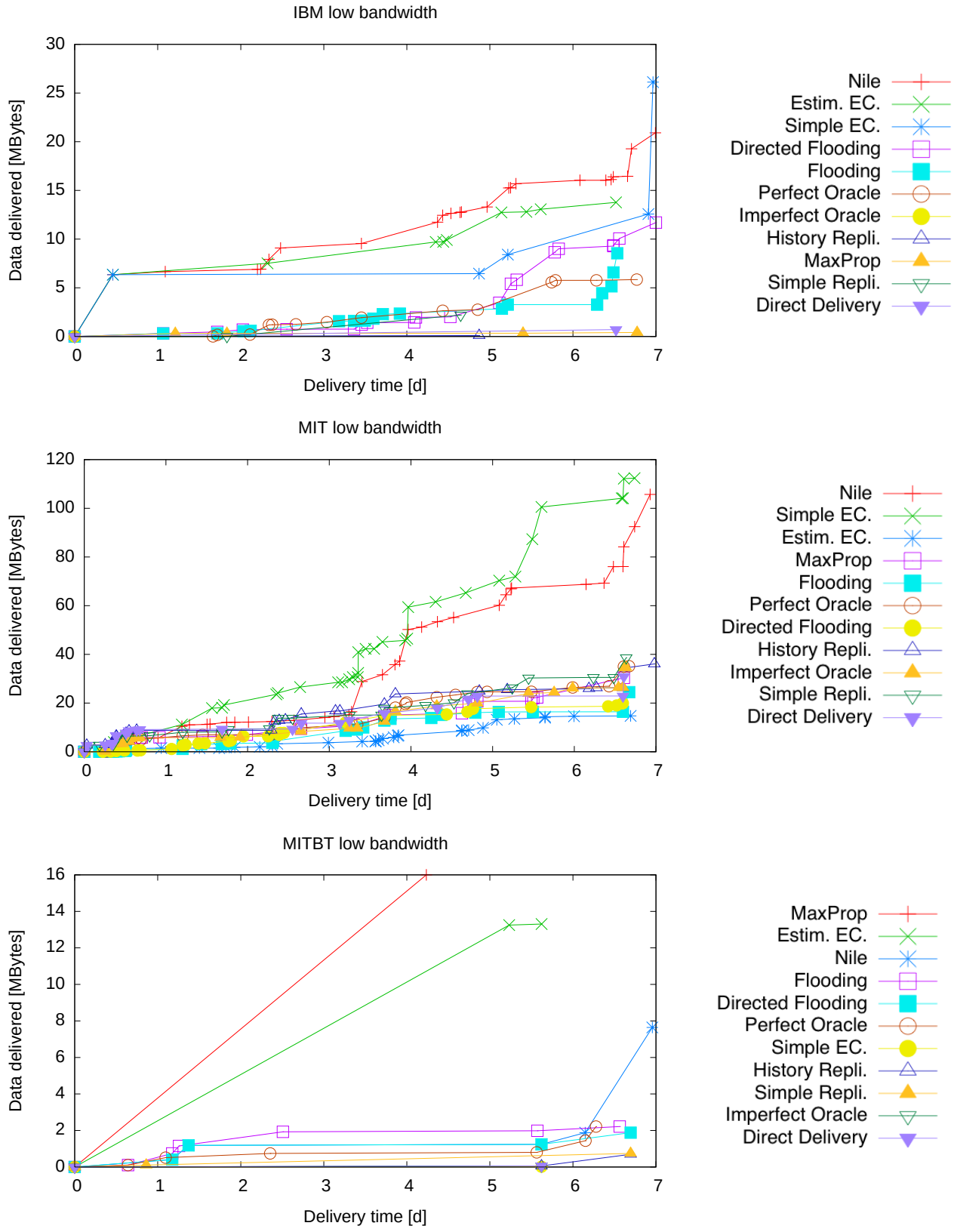


Figure 7.3: Data delivered with high bandwidth from access point IBM (a), cell tower MIT(b) and bluetooth MITBT traces(c)

Comparison

What we can see is that only 5 algorithms make it to the top: *Flooding* and *perfect oracle* dominate under weak connectivity, while *MaxProp* works well under high connectivity. *Flooding, EBEC* and *simple replication* can take advantage of "nice" networks (high bandwidth, high connectivity). Since *simple replication* gives opportunity to each replica of the message to take at least one hop in a dense network like the cell tower (MIT), its performance in high bandwidth case is very impressive.

MaxProp performs very well when the bandwidth is low, but does not seem to be able to take advantage of higher bandwidths: It does not delivers significantly more despite 10- or 100-fold increases of bandwidth. Its noteworthy that *MaxProp* is the only algorithm that defines the queue management mechanism for the devices that apparently gives it an edge. This technique gives priority to messages that are destined for the next hop and sorting the remaining messages according to their age, i.e. younger messages are given priority for next transfer opportunity. We believe that this strategy provides quick delivery of fresh messages while being persistent about older messages. Together with the fine-grained proportional message sharing, this ensures that many pieces of the message follow the best path quickly.

MaxProp also employs a unique strategy to compute history information that in our view helps the algorithm to better determine the candidate next hops. We believe that the history normalization employed by *MaxProp* helps reducing the chance that too many messages will be loaded onto a very mobile device, which does not stay long enough in the vicinity of the destination to actually deliver all messages.

If you wanted to implement only one algorithm of these preexisting algorithms, hoping it would perform reasonable under most regimes, *flooding* and *EBEC* are the candidates to consider (*perfect oracle* can be excluded, as it seems impossible to implement the necessary oracle in real life).

In first low bandwidth case in Fig. 7.2 and Fig. 7.3, we observe that *Nile* has performed almost equal to *flooding* as far as number of messages are concerned. If we analyze the sizes of the messages Fig. 7.3, *Nile* is far better than *flooding* and is among the best that have performed size-wise; i.e., erasure coding based methods [WJMF05, LTZG06]. The reason why erasure coding techniques have performed better as the size of the message is shrinked due to the fragmentation of the messages. As far as the number of delivered messages in high bandwidth

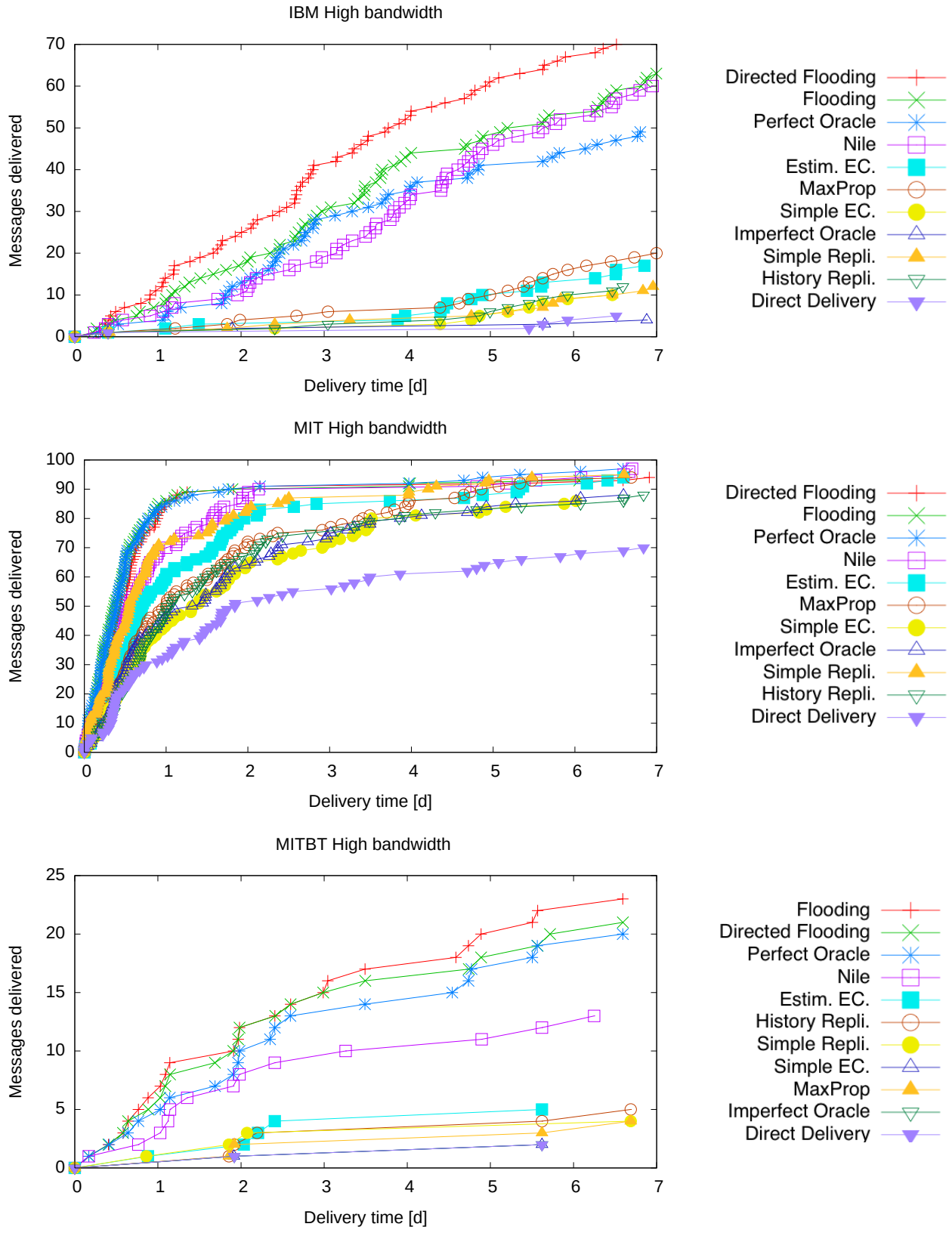


Figure 7.4: Number of messages delivered with high bandwidth from access point IBM(a), cell tower (MIT)(b) and bluetooth (MITBT) traces(c)

is concerned (Fig. 7.4), *flooding*, *directed flooding*, *perfect oracle*, and *Nile* are in the top category. In Fig. 7.4, we observe that *Nile* has appropriately adapted to the available bandwidth whereas none of the other competitors except *flooding* have been able to do so. The best performing method is *directed flooding*, the predecessor of *Nile*. The rest of the protocols have not been able to take advantage of extra available bandwidth.

In Fig. 7.3(b) and Fig. 7.2(b), in the case of dense and resource deficient cell tower (MIT), *Nile* performs as good as the best competitor, i.e., *simple erasure coding* [WJMF05]. Erasure coding techniques have performed significantly well because the contact durations of the MIT network are smaller than those of the IBM trace, although, the network is quite dense. The majority of the nodes in the MIT network had a direct contact with each other but the duration of the contact was small, hindering the delivery of large messages. This can be seen in high bandwidth scenario (Fig. 7.4(b)), in which each of the protocol has taken advantage of the dense network and delivered majority of the messages.

The most sparse network in our analysis, i.e. the bluetooth (MITBT) (Fig. 7.5(c) and Fig. 7.4(c)) has proven the necessity of replication more than any other network. Since there have been paths that do not occur frequently, the history is not very reliable and *flooding* remains the uncontested winner in this case. As far as the low bandwidth case is concerned, MaxProp has been a pleasant surprise to deliver a few big messages (Fig. 7.3(c)). As far as count is concerned, *flooding* and *perfect oracle* at the top as usual (Fig. 7.2(c)).

To answer further questions, we compiled a figure of merit (Fig. 7.6), summarizing all the best algorithms in each of the bandwidth/connectivity areas. In each square, the top row corresponds to the top performers, which are roughly on par with each other. The second row describes the second group, and so on.

7.3.2 Peripheral message metrics

In this section, we analyze the overhead encountered by each message for each of the protocols. Each message is examined according to three criteria (presented below) irrespective of whether it was delivered or not delivered. Some routing strategies transmit more messages than others, either because they use multiple copies of each message, make different decisions about the next hop, or because of the protocol overhead. The number of transmissions is a measure of the amount of contact capacity consumed by a protocol. It is also an approximate measure of the computational resources required, as there is some processing required for

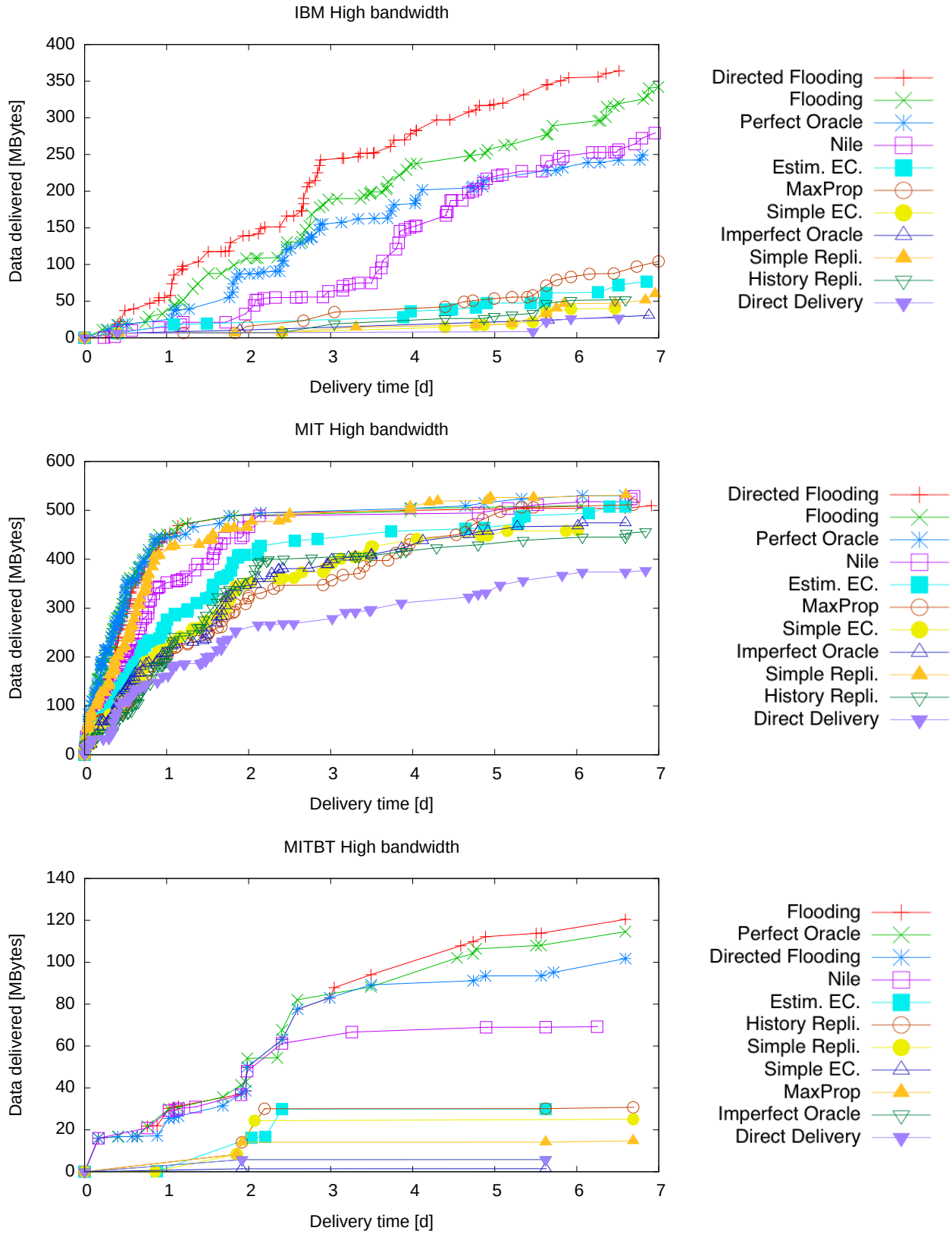


Figure 7.5: Data delivered with high bandwidth from access point IBM(a), cell tower MIT(b) and bluetooth MITBT traces(c)

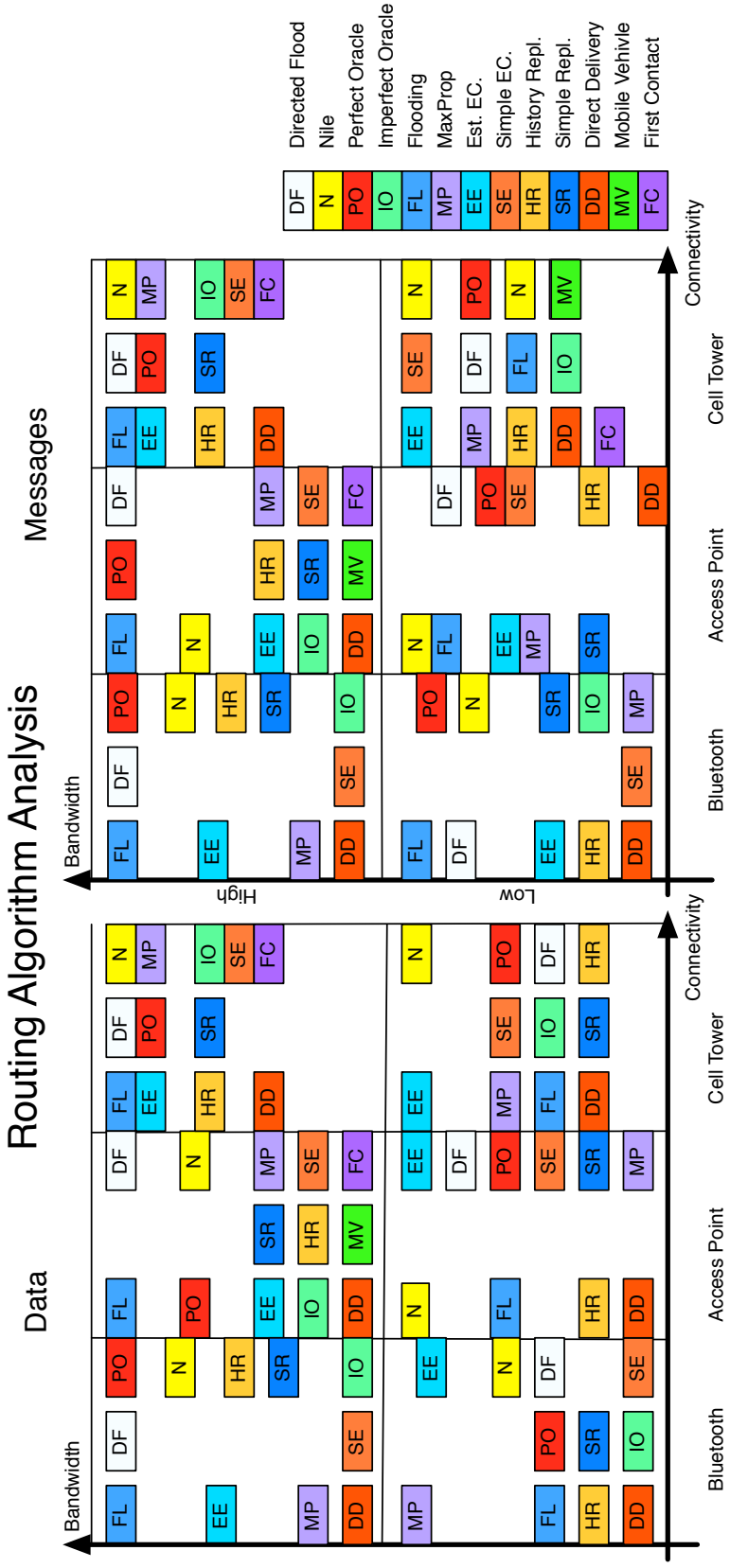


Figure 7.6: Performance behavior of routing algorithms with respect to amount of data (left) and number of messages (right) delivered

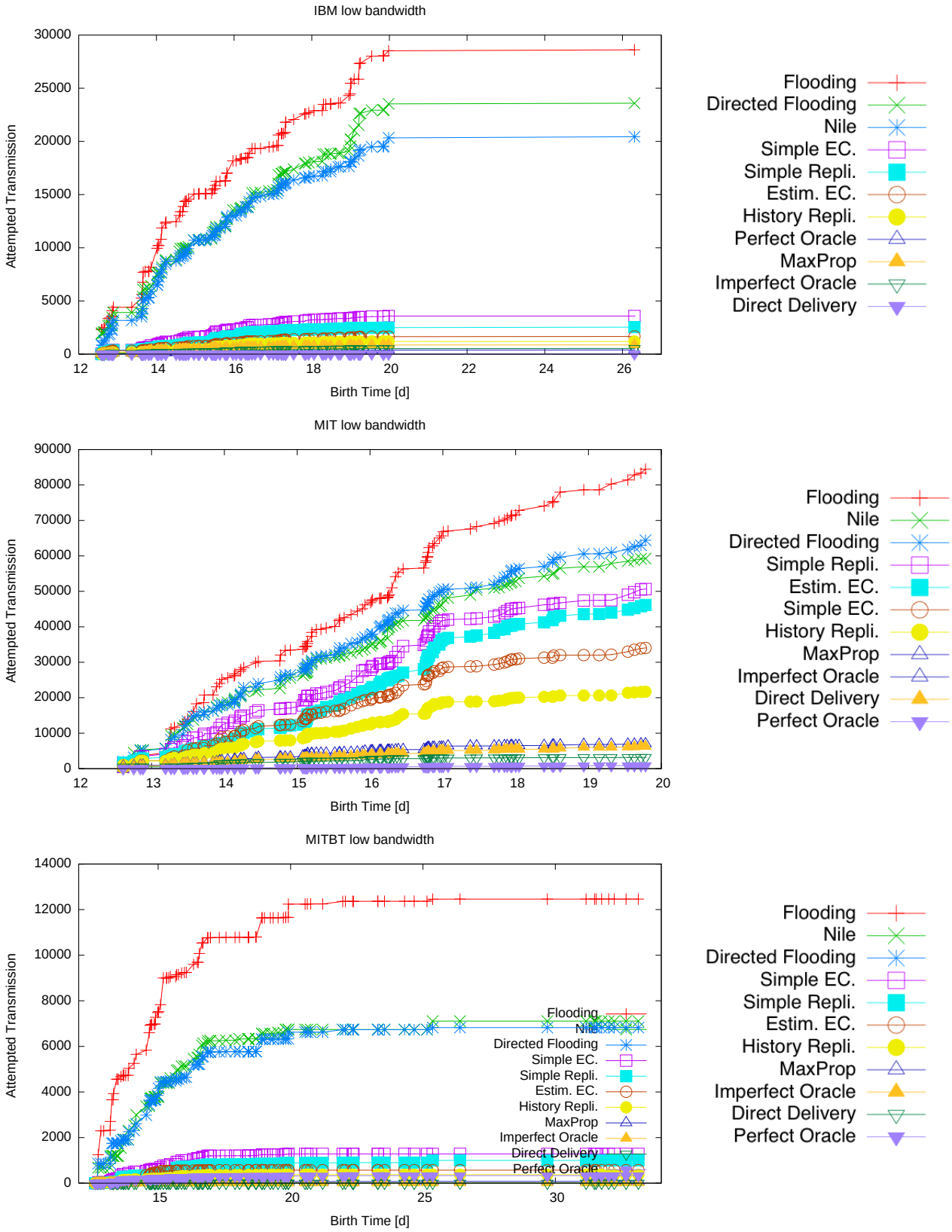


Figure 7.7: Transmission attempts made in low bandwidth

each message. Additionally, each transmission consumes energy, so it is also an approximate measure of power consumption.

- **Attempted transmission** represents the number of times a message has been attempted to be forwarded to the next hop. This metric will be higher for those protocols that are replication based and suffer from congestion, because such a protocol will have to make unsuccessful attempts to forward the message to the next hop. Attempted transmission is reduced to $1/fragCount$ of it value for all protocol that fragmentize the messages.
- **Completed transmission** represents the number of times a message has been successfully forwarded to the next hop. This metric has been presented to show the extent of difference between unsuccessful and successful propagation. Completed transmission metric also is dealt with in the same way as the previous metric as far as fragmentation based protocol are concerned.
- **Bytes transferred** represents the number of bytes that are associated with *attempted transmission* showing the real extent of congestion exerted by a protocol on the network.

The motivation behind presenting these peripheral metrics is to show the hidden aspects that reveal the working of the protocol. These aspects may not be visible to the naked eye but when a protocol is simulated on various opportunistic networks, those aspect will highlight the potential weak points that can degrade the performance in the scenario where the protocol has not been tested.

Low bandwidth

When we look at the attempted transmissions in low bandwidth scenario of all the networks in Fig. 7.7, obviously *flooding* has the highest count of attempts to forward a message followed by *directed flooding*. The exceptional behavior of *simple replication* in the cell tower (MIT) network can be explained by the fact that *simple replication* attempts to forward one replica of message to any device that comes coincidently in contact with the source. As the cell tower (MIT) trace has many small duration meetings, most of these forwarding attempts are unsuccessful and the protocol has to reattempt the procedure with the next coincidental contact.

As far as Fig. 7.8 is concerned, *directed flooding* has more completed transmissions as compared to *flooding* in the cell tower MIT and the access point IBM cases. The reason for non-compliance of the bluetooth MITBT trace with other traces is

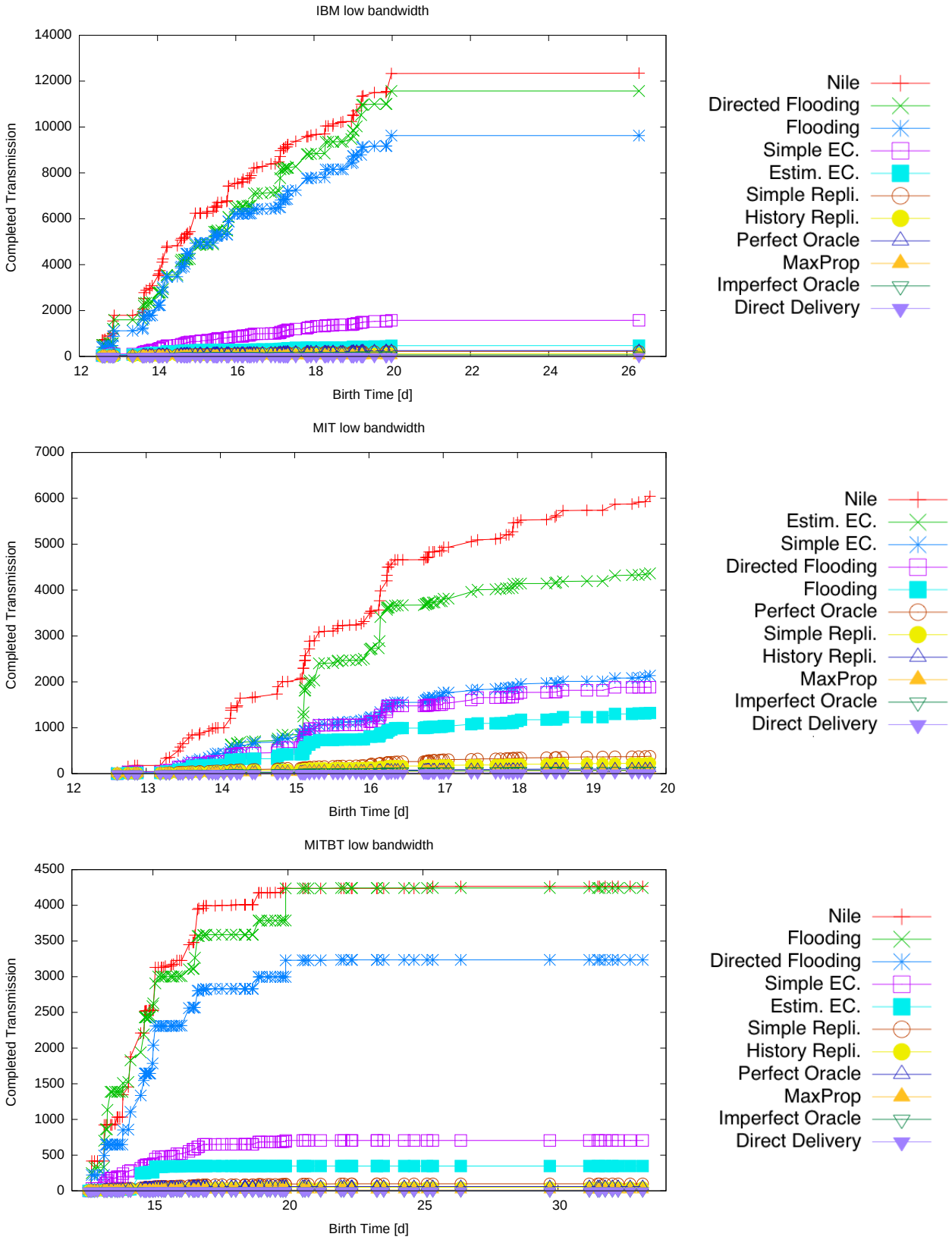


Figure 7.8: Transmissions completed in low bandwidth

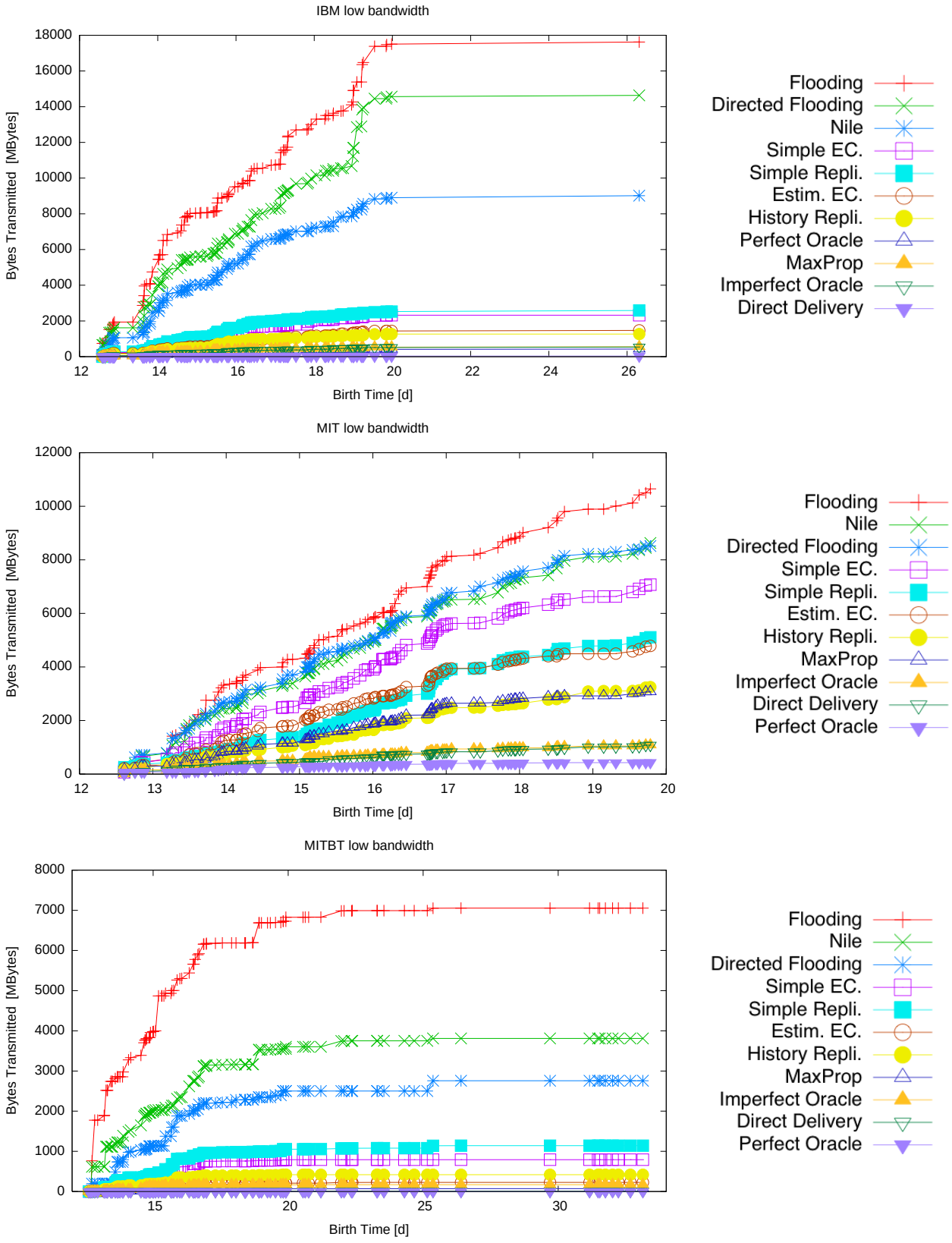


Figure 7.9: Bytes transmitted in low bandwidth

lack of mobility. As *directed flooding* is dependent on gathering the path information shared by mutual interaction of devices, and due to low bluetooth (MITBT) mobility, not all the paths have been reported to the source device in time. This lack of information has resulted in a considerable decrease in the number of paths available for *directed flooding*.

In the case of bytes transmitted (Fig. 7.9), with the three *flooding* based protocols being on top, the behavior is pretty much as expected, transmitting most of the bytes for all the messages. In the dense cell tower (MIT) scenario the non-flooding based protocols have been able to transmit more bytes as compared to the relatively sparse access point (IBM) and bluetooth (MITBT) networks.

High bandwidth

The high bandwidth scenario generally represents a similar behavior as the low bandwidth scenario, however, the magnitude of all the defined metrics is relatively high. Both, *flooding* and *directed flooding*, have forced all the metrics to skyrocket thereby exploiting the availability of surplus bandwidth to the maximum extent. The fact that *Nile* has not followed both of its predecessors shows that protocols have a chance to deliver *flooding* comparative performance with reduced overhead on network resources.

In Fig. 7.10, we do not see much of a difference between *flooding* and *directed flooding* in dense cell tower (MIT) case, but when we correlate the attempted transmission results with completed transmission as shown in Fig 7.11 we observe evidence of more successful transmissions for *directed flooding*. This can be explained as *directed flooding* has a crude path pruning capability, Furthermore, it enjoys lower message interference that makes it possible to propagate several small messages, which increases the magnitude of completed transmissions. This argument is supported by Fig. 7.12(c), where we can observe that the difference between *flooding* delivered bytes and *directed flooding* delivered bytes is smaller than the difference between *flooding* completed transmissions and *directed flooding* completed transmission.

EBEC has behaved in an interesting manner by showing an unusual large magnitude of all three metrics in consideration in the cell tower (MIT) trace. After investigating the reasons behind this behavior in depth, we found jittery metrics responsible for this phenomena. Since the protocol has not defined any threshold that must be surpassed to commit a probability update, and the MIT being a dense network, several messages start showing "hot potato" behavior among the devices. The reason being that with every meeting encounter the probability of

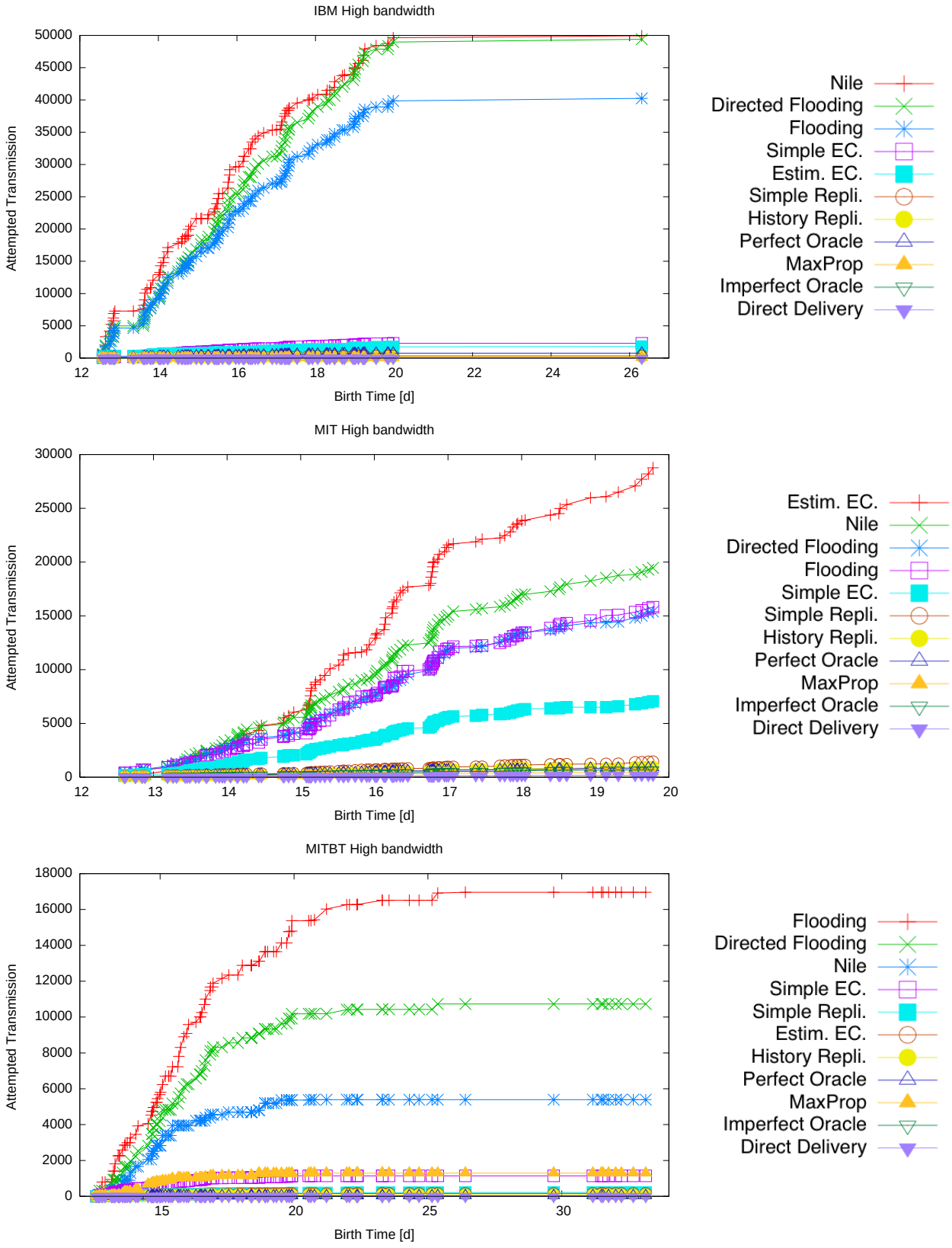


Figure 7.10: Transmission attempts made in high bandwidth

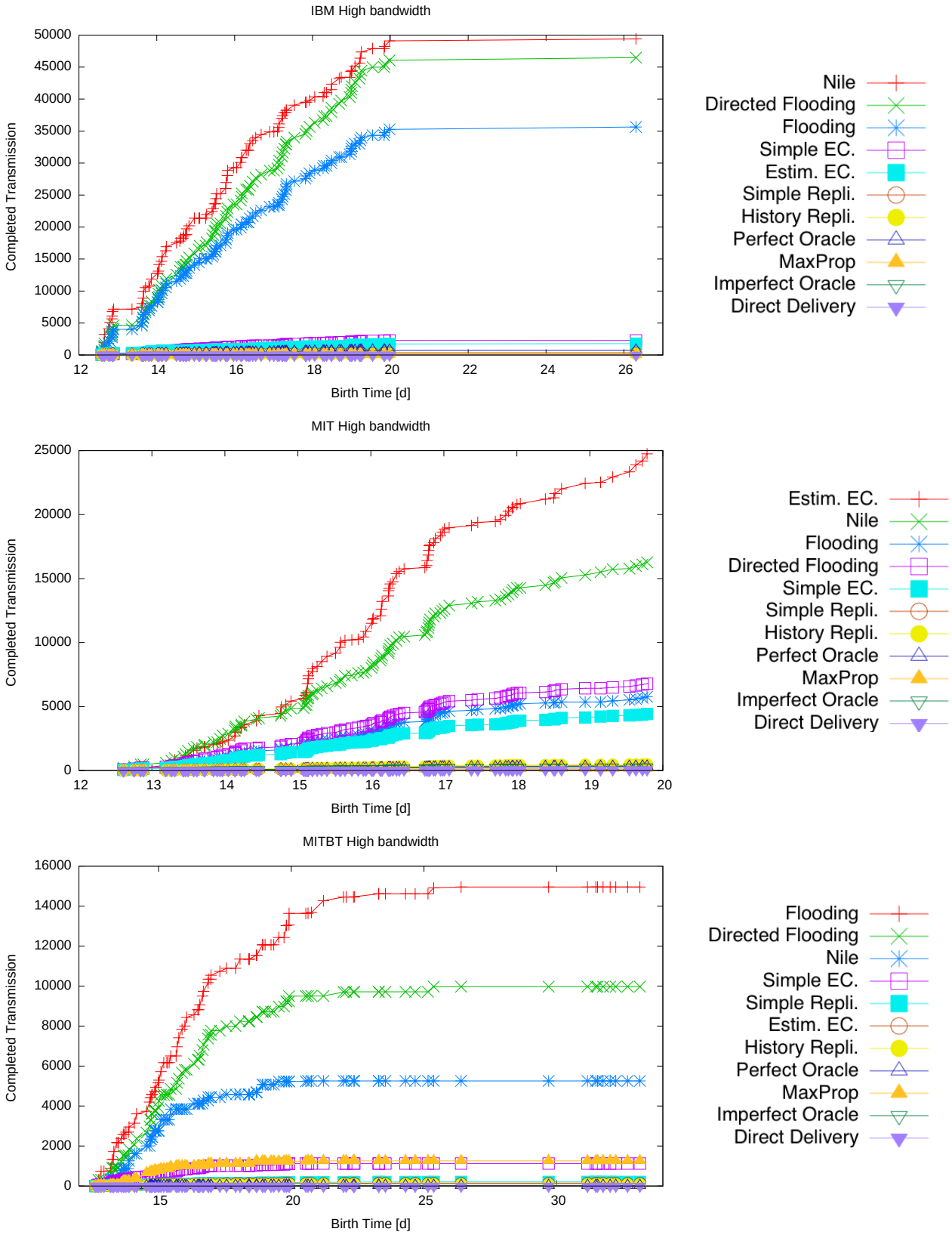


Figure 7.11: Transmissions completed in high bandwidth

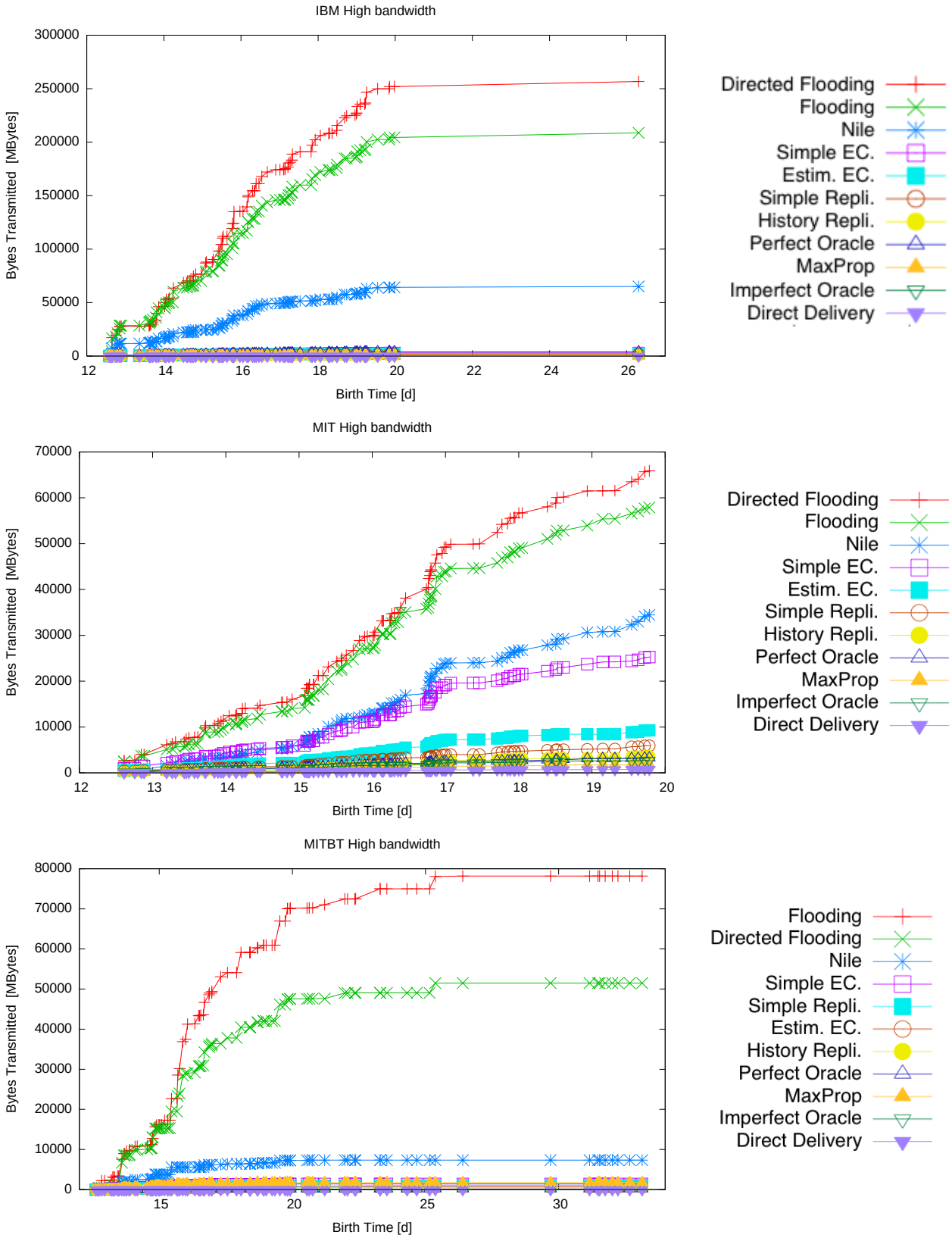


Figure 7.12: Bytes transmitted in high bandwidth

one device to reach destination improves; this situation reverses when the next contact takes place and the message hops back to the device from which it just hopped off.

Summary-Peripheral measures: The summary of all the measures presented in Fig. 7.13 shows a considerable shuffle among the positions between the three measures, i.e. transmission attempts completed and bandwidth consumed. The standings in the sparse networks are mostly the same, but when we look at the dense network of MIT, we can relate the transmission attempts to the bytes consumed but completed transmissions does not agree with the rest.

7.3.3 Queuing effect

In this section we will present the results of experiments described in section 6.2.4. Although, local queuing methods are applicable to all of the routing strategies, we have presented results related only to *Nile* in the low bandwidth scenario. These selected cases provide the proof of all of our arguments and thus, for the sake of simplicity, we have omitted the rest of results.

When looking at the routing performance of *Nile* equipped with queuing method (Fig.7.14), we observe that local queuing pushed the performance of *Nile* a little further but it was not able to affect all of the messages in a uniform manner. Local queuing did help some of the messages to arrive earlier at the destination but also delayed the delivery of a few others. Specifically in the case of bluetooth trace, the difference is very much in favor of local queuing whereas access point as well as cell tower have shown a random behavior. This random behavior can be explained by the fact that every device, using *Nile*, being in a distributed environment, can make a routing decision that can have a canceling effect by a decision from another device.

An analogy with ocean waves can help us understand this phenomenon even better. Due to all the nature activities, the water waves are generated randomly in an ocean. In a rare case, a rogue wave can occur even in normal weather conditions, when several waves coincidentally merge together and form an enormous wave [JA06]. In the case of opportunistic networks, timing of a message (wave) can be critical for reinforcing or totally canceling out propagation of another message.

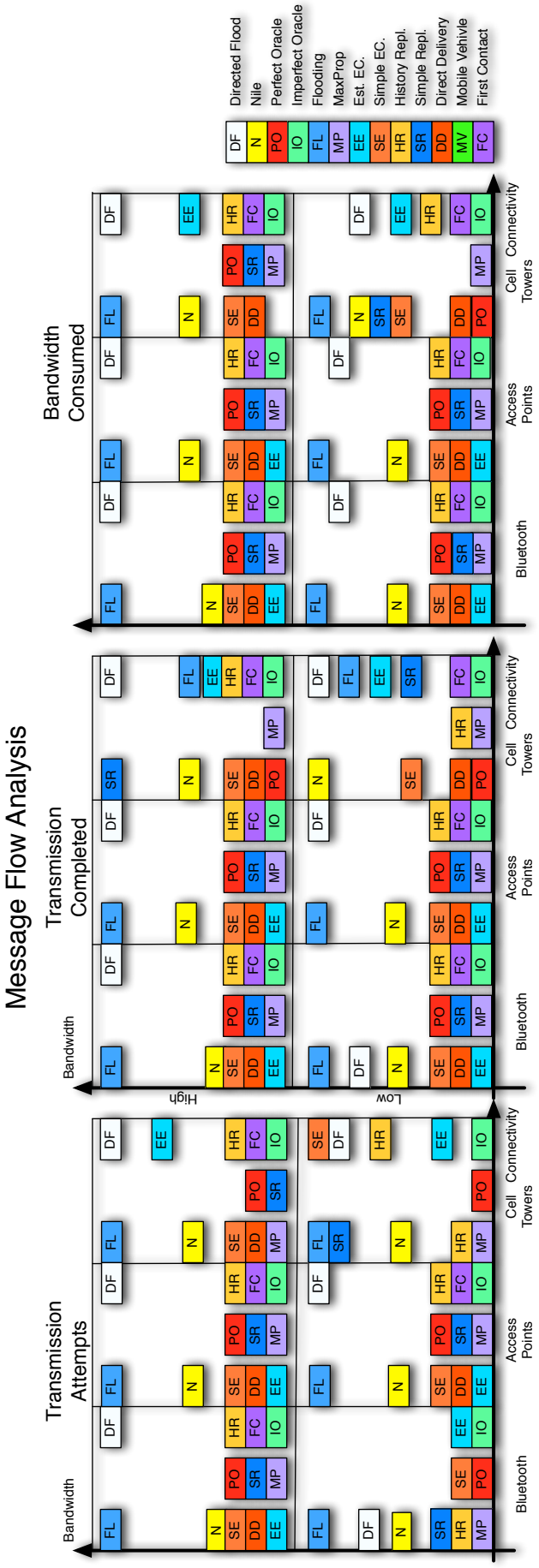


Figure 7.13: Standing of routing algorithms w.r.t. transmission attempts (left) transmission completed (center) bandwidth consumption (right)

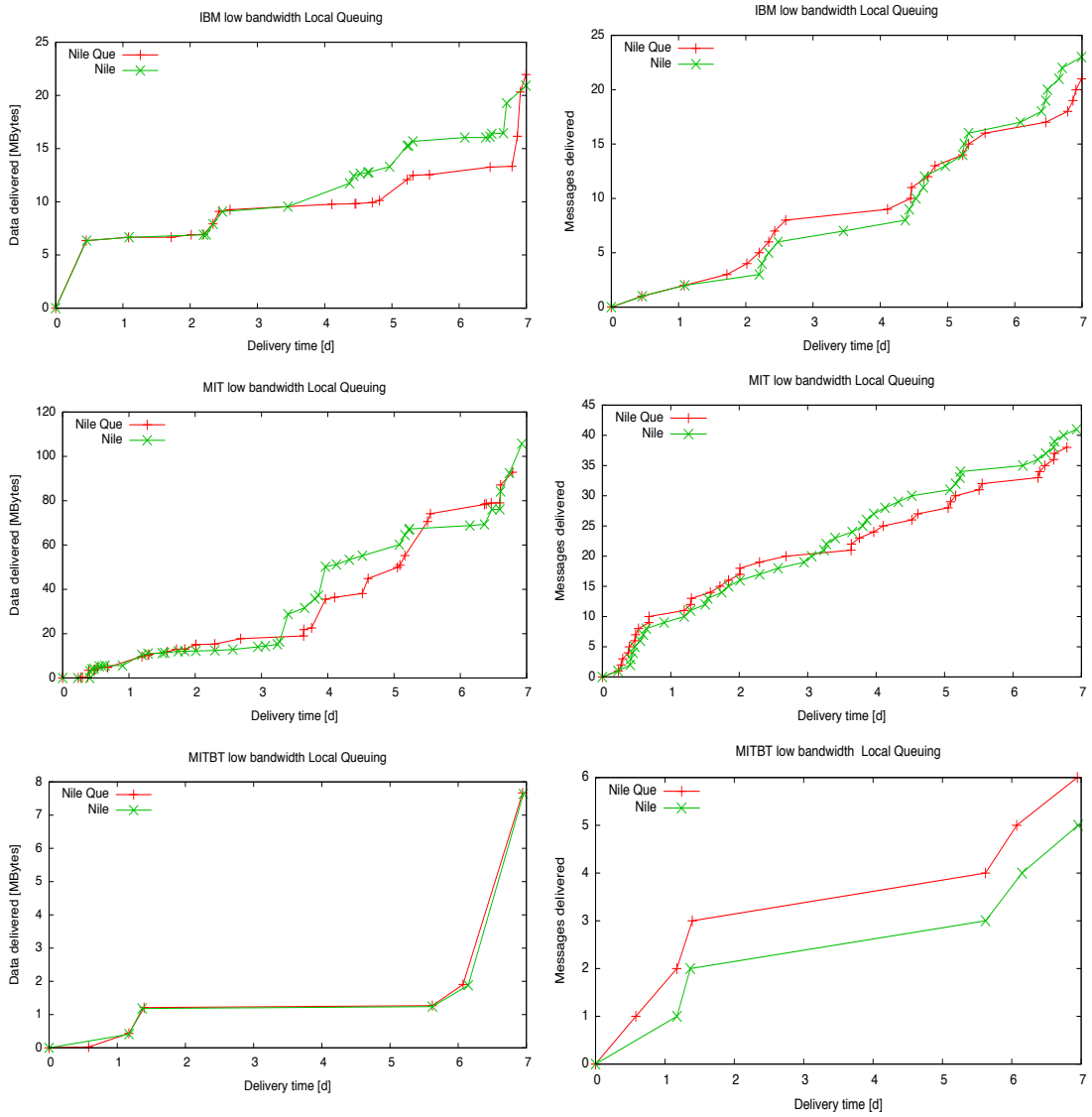


Figure 7.14: Message transmitted with local queuing in low bandwidth

7.4 Local storage overhead

In this section, we present the graphs that show how much storage memory has been consumed by different protocols, in the whole network, in the low bandwidth scenario Fig. 7.15, as well as the high bandwidth scenario (Fig. 7.16), the X axis shows the point in time during one month period and the Y axis shows the amount of data in bytes.

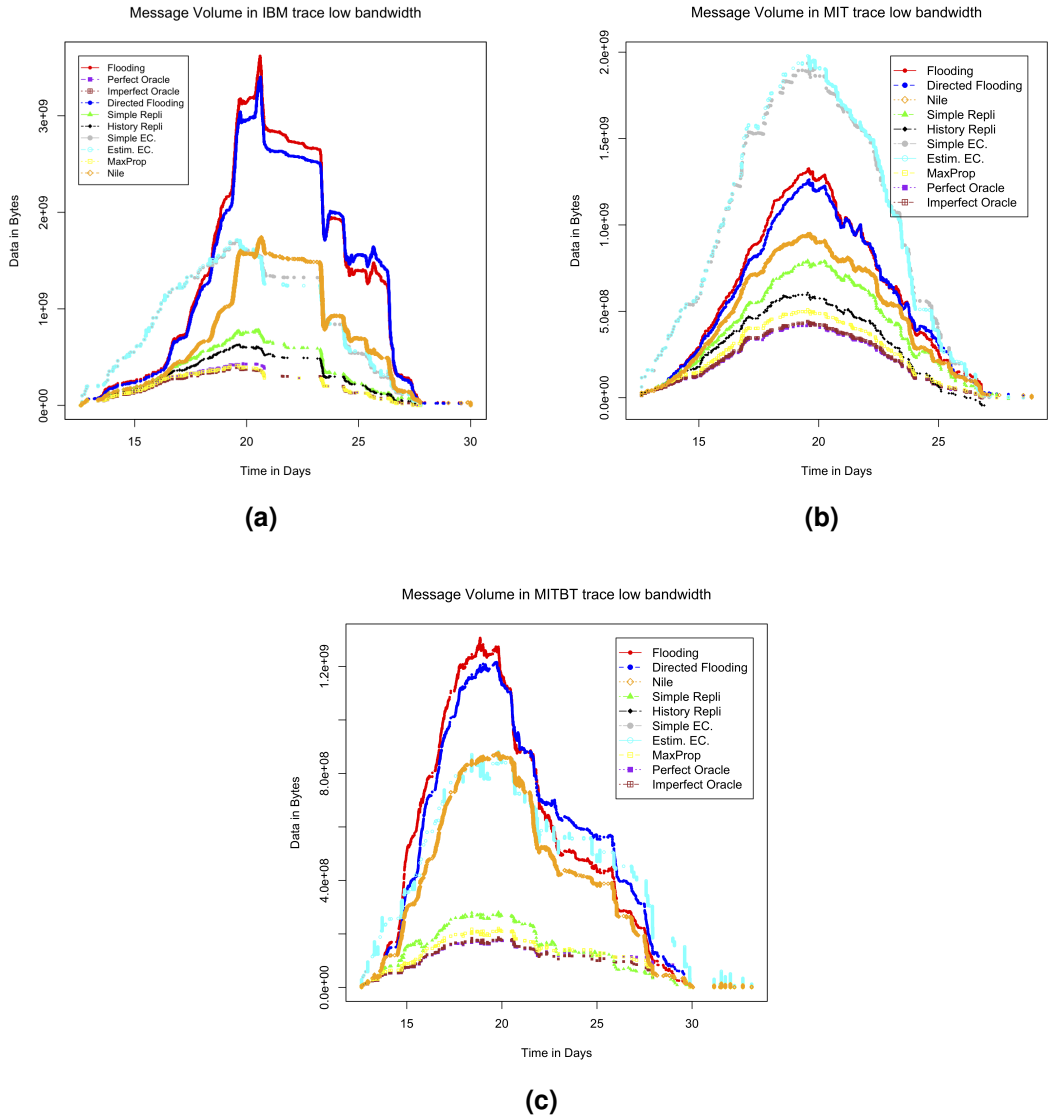


Figure 7.15: Storage overhead in the (a)access point (IBM), (b)cell tower (MIT) and (c)bluetooth (MITBT) trace with low bandwidth

7.4.1 Low bandwidth

When we look at the traffic overhead encountered in low bandwidth scenario of all three traces (Fig. 7.15a, 7.15b, 7.15c), we observe that *flooding* is not one of the highest consumers of local storage in the cell tower (MIT) trace (Fig. 7.15b). This confirms our point that *flooding*, in scarce bandwidth was not able to repli-

cate messages to the same extent as other protocols did. It can be possible that messages fully choked several bottleneck links and thus those over-chocked links prevented *flooding* from rampaging the local storage of devices that were present after the bottleneck on corresponding paths. As we will show in the next section, *flooding* did consume the maximum bandwidth in this scenario, proving the point that *flooding* was not able to replicate according to its will. In the access point (IBM) scenario, where the contact duration has been longer, flooding not only performed among the best but also consumed a lot of local storage as well. In both traces, *directed flooding* has performed better than *flooding* without consuming corresponding local storage.

The erasure coding based methods have been the biggest culprits of local storage consumption in the cell tower (MIT) trace. The explanation being that erasure coding fragmentizes the messages, and thereby reduces the size of the messages that makes them pass through bottlenecks links. We have tested this hypothesis by increasing the number of fragments per message, thus reducing the size of each fragment. The performance results improved with extra consumption of local storage.

As already stated, *Nile* has adopted the fragmentation from erasure coding based techniques but no erasure coding has been done. *Nile* has been clearly economical in both the access point (IBM) and the cell tower (MIT) traces. In the access point (IBM) case, the storage consumption has shown pattern similar to *flooding* and *directed flooding* but the magnitude is approx. 1/3. In the cell tower (MIT) scenario, *Nile* has consumed more local storage than single copy and non erasure coding two-hop replication strategies.

The lowest storage consumption cluster consists of techniques that are either single copy or two-hop. Interestingly, when we compare the magnitude in both traces, this cluster has almost similar values. The reason is that these method have defined a small number of replicas and these methods are able to create them successfully.

7.4.2 High bandwidth

The high bandwidth case shows the results expected by everyone. Fig 7.16, show that *flooding* and *directed flooding* have been the dinosaurs of local storage consumption in the access point (IBM) as well the cell tower (MIT) trace. The magnitude of consumption is up to approx. 5 times more than the lowest consuming cluster. It is interesting to notice in the sparser case of the access point (IBM) that *directed flooding* was able to replicate more than *flooding*, showing

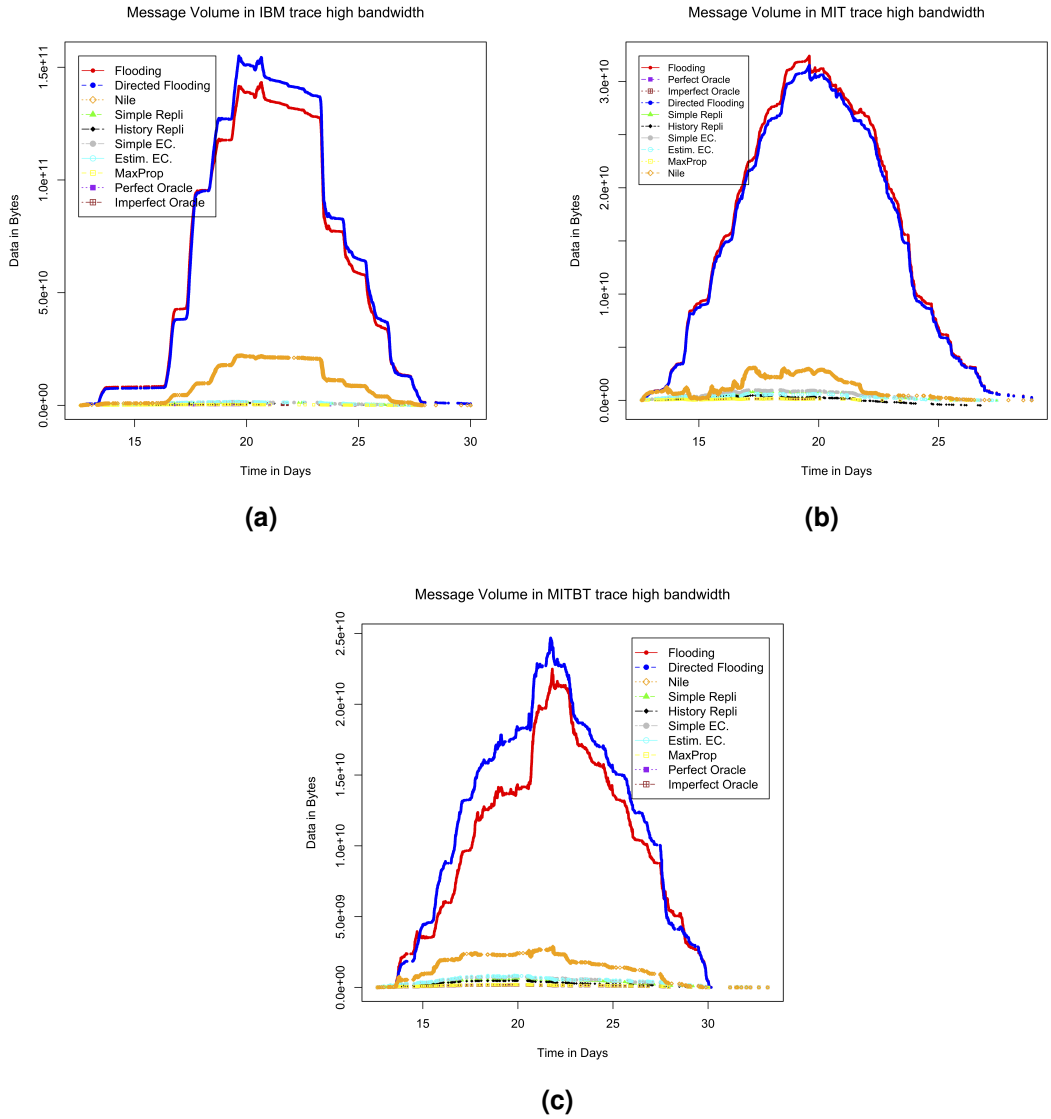


Figure 7.16: Storage overhead in the (a)access point (IBM), (b)cell tower (MIT) and (c)bluetooth (MITBT) trace with high bandwidth

that *directed flooding* forwarded the messages only in the direction of destination. In comparison to flooding, which attempted to replicate every message in every "direction", directed flooding was thus giving an opportunity to replicate a subset of messages.

Nile has been the second biggest consumer of local storage but the magnitude is

clearly far lower than flooding. Keeping in mind the performance of *Nile* in the high bandwidth case, it shows that our method of building the "dams" to stop the data to flood has worked very well. Particularly in the case of the cell tower (MIT), we can see an irregular pattern of *Nile* storage consumption, stopping the leakages in a high density scenario. Another reason of this irregular pattern is the feedback system explained in Section 6.2.3 that *Nile* uses to notify the devices about delivered messages. As in the previous case, the lowest storage consumption plot cluster consists of techniques that are either single copy or two-hop. The reason being that their replication factors have been pre-defined, i.e. the replication factor of 4 is assigned to replication methods and an encoding factor of 4 is assigned to erasure coding techniques.

7.5 Transmission overhead

Transmission overhead is described as the bandwidth consumed in the network during propagation of the messages. The results presented in this section are irrespective of the messages that have caused them, and each bandwidth consumption event is tagged with timestamps. For the sake of understanding, these graphs are not decaying because they represent the consumed bandwidth that cannot be recovered. Local storage graphs were decaying because storage is recovered once the message is either delivered or dies after its lifetime expires.

7.5.1 Low bandwidth

When we compare the plots of all the three networks in the low bandwidth scenario, shown in Fig. 7.17, flooding based protocols are the biggest consumers of bandwidth. With the exception of dense the cell tower (MIT) network, other protocols do not scale to the available bandwidth (Fig. 7.17a, 7.17c). In the cell tower (MIT) case (Fig. 7.17b), we can see a uniform spectrum with respect to bandwidth consumption where flooding based protocols being on the top, replication based in the middle and single copy protocols at the bottom. In all three cases, *Nile* can be seen as a compromise between flooding based and replication based protocols pictorially shown in Fig. 5.8. The important difference in the cell tower (MIT) low bandwidth scenario between local storage (Fig. 7.15b and Fig. 7.17b) shows that flooding based algorithms, although they have consumed large amount of bandwidth. They have not been able to consume local storage space. This means that the bandwidth consumption created several local bottlenecks and these bottlenecks blocked the propagation of messages to reach the rest of the devices in the network.

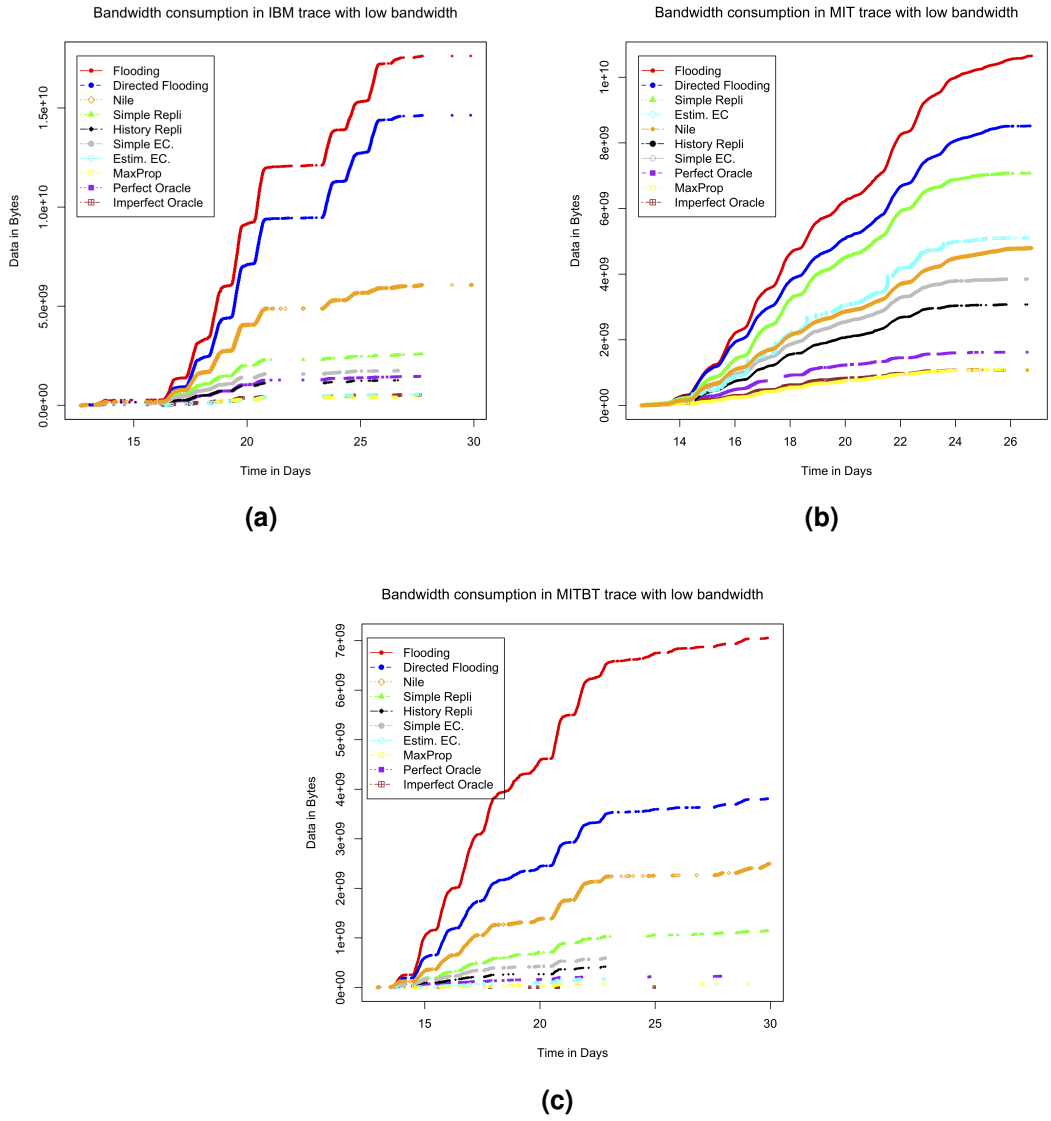


Figure 7.17: Bandwidth overhead in the (a)access point (IBM), (b)cell tower (MIT) and (c)bluetooth (MITBT) trace with high bandwidth

7.5.2 High bandwidth

When we look at high bandwidth scenario in all three networks Fig. 7.18, no surprises are noticed. Flooding based protocols have exploited the presence of bandwidth available and thus performed far better than others in sparse networks, whereas the metrics used by the rest of the protocol did not make them aware of

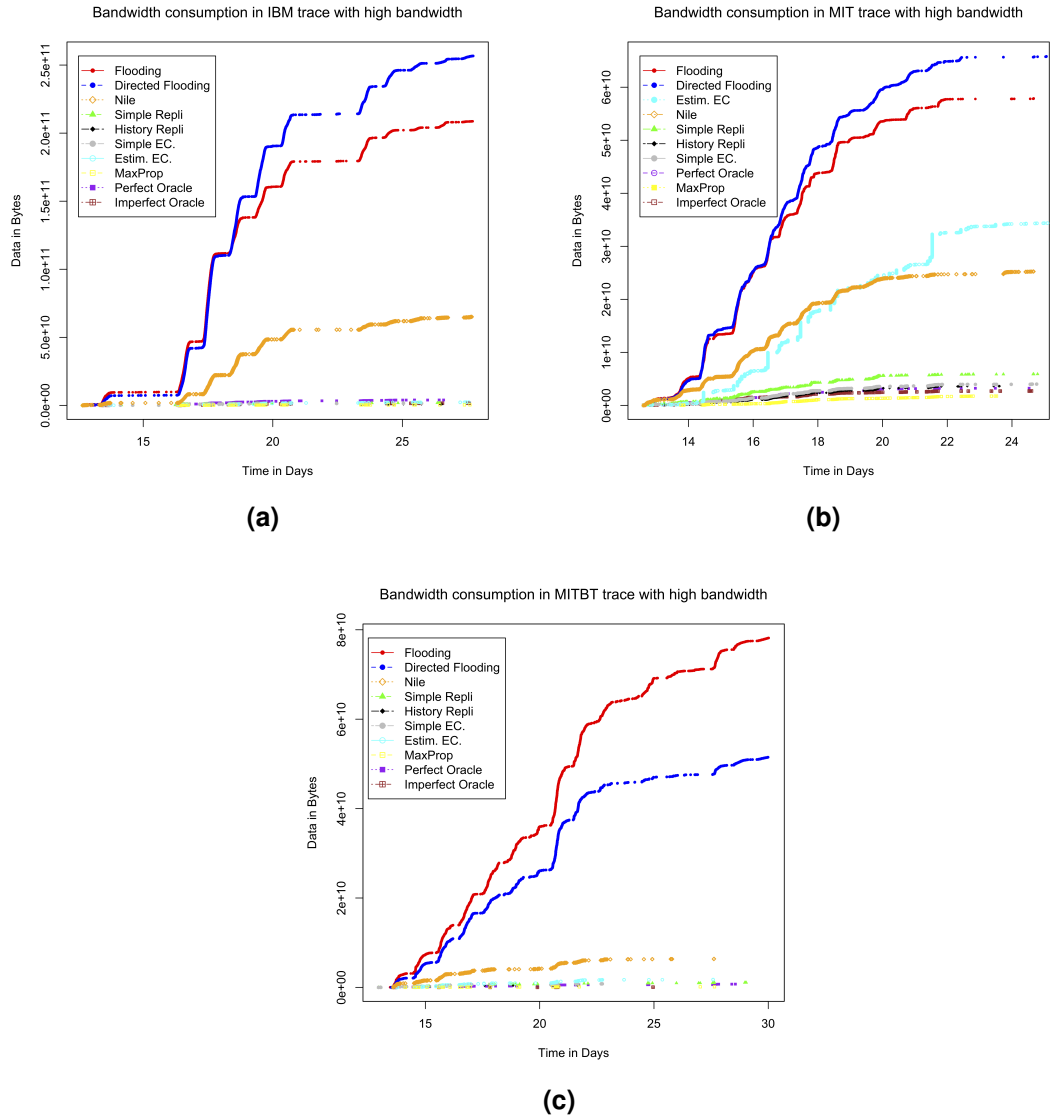


Figure 7.18: Bandwidth overhead in the (a)access point (IBM), (b)cell tower (MIT) and (c)bluetooth (MITBT) trace with high bandwidth

bandwidth availability, and thus, they failed to take advantage of this situation.

Summary-overheads In Fig. 7.19 we have placed the routing protocols according to both types of overheads and we can see the difference between the two. Not surprisingly, bandwidth consumption block is almost the same as observed in Fig. 7.13. One key aspect of Nile is that it is putting medium strain in all the net-

works with all bandwidth configurations. *Flooding* is the top consumer in all cases as far as bandwidth consumption is concerned but in dense-bandwidth deficient environment, i.e. cell tower (MIT), erasure coding techniques have consumed more local storage. Depending on the nature of the network and available bandwidth, *directed flooding* shares its overhead characteristic with either *flooding* or *Nile*. All the single copy methods are very economical in all the cases followed by two hop relay methods like *simple replication* and *history based replication*.

7.6 Network participation

In this section we present the behavior of devices with respect to handing of traffic in all the three networks. We have used 4 metrics to differentiate device behavior.

- **Bytes received** is the traffic volume that has been received by a device from all of its direct neighbors during the simulation period. This measure includes partial transmissions.
- **Bytes sent** is the traffic volume that has been sent by a device to all of its direct neighbors during the simulation period. This measure also includes partial transmissions as well.
- **Bytes traversed** is the traffic volume that has been propagated by a device from its direct neighbors to other direct neighbors during the simulation period. This measure does not include partial transmissions as a device cannot forward a message unless it has the complete message. If messages are fragmented then the complete fragment of a message must be received by a device before its is attempted to be propagated to the next hop.
- **Bandwidth available** is the traffic volume that has been reported by a device that was involved in a transmission of any message during the simulation period. Although, this is not a very accurate measure since it is recorded only for those devices that have been involved in (un)successful transmission during the simulation of a particular protocol Nevertheless, it provides an estimation of available bandwidth.

As already stated, the plots presented in the following show the extent of device participation where devices are sorted according to the measure in each plot. To improve the presentation and clarity in the figures, we have plotted the log of respective device activity on Y-axis, while X-axis represent the corresponding rank

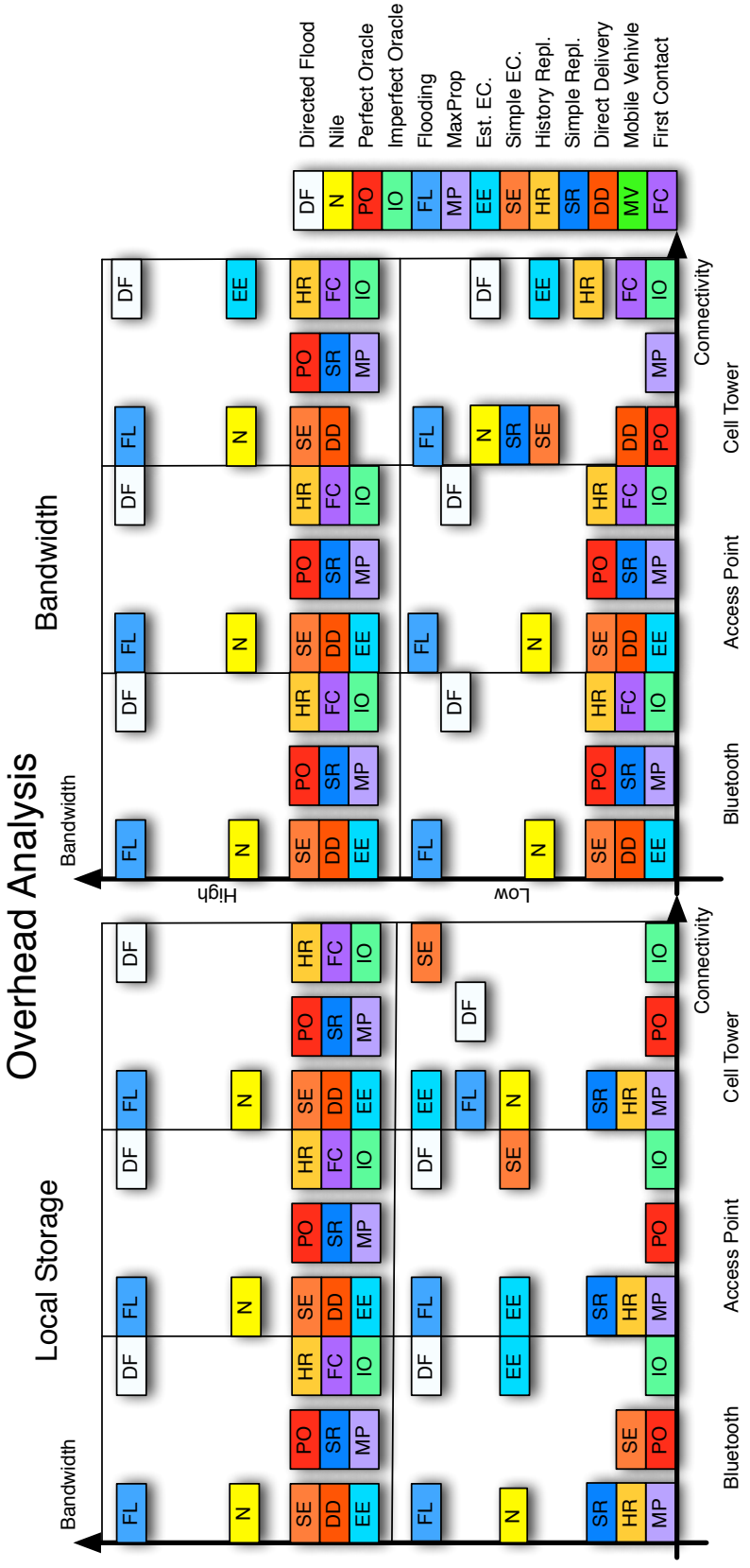


Figure 7.19: Overhead of routing algorithms with respect to local storage (left) bandwidth consumption (right)

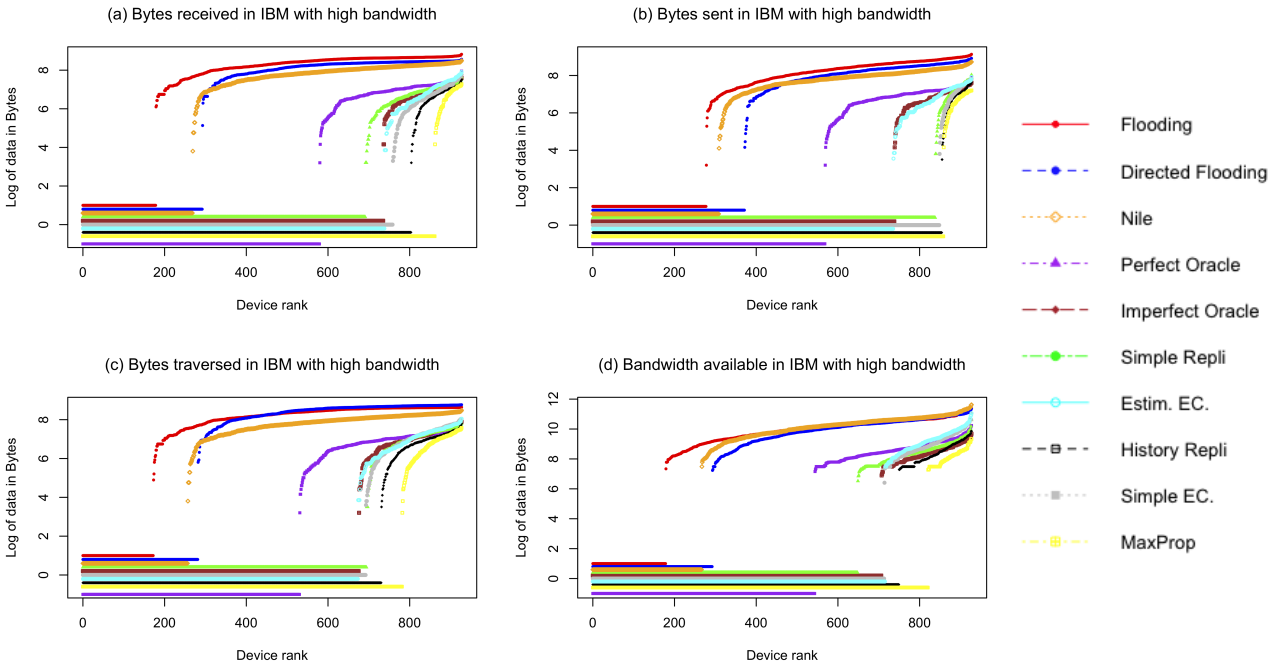


Figure 7.20: Device behavior in access point trace with high bandwidth

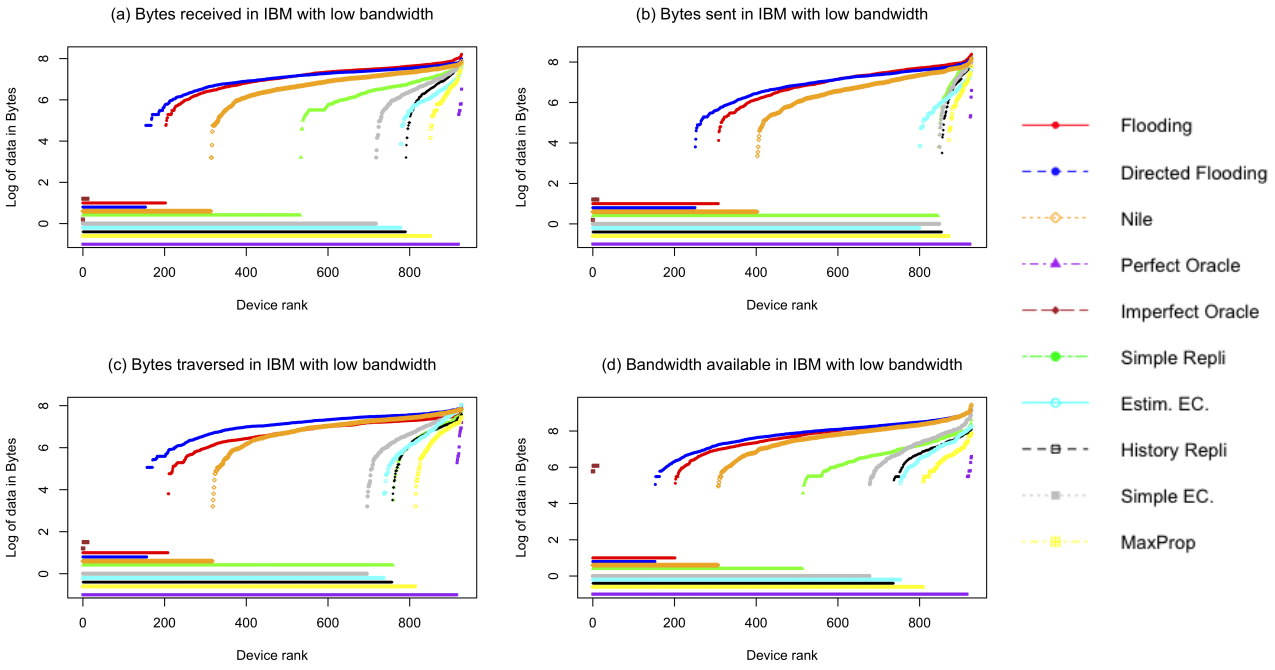


Figure 7.21: Device behavior in access point trace with low bandwidth

of the device. Furthermore, we have plotted all the methods using different hypothetical values in the range of [-1,1], for corresponding undefined values of log. These values represent inactivity of devices shown by straight lines at the bottom of every figure.

7.6.1 Access point - IBM

In the scenario of access points, *flooding* is utilizing the maximum number of devices with high bandwidth in all measures followed by *directed flooding* and *Nile* (Fig. 7.20) represents the big diameter of the network. As far as the aspect of bytes traversed is concerned, *directed flooding* employs fewer devices than *flooding* but the throughput volume is higher than that of *flooding*. If we look at device ranking with respect to available bandwidth, *Nile* uses maximum number of devices for propagation because *Nile* fragmentizes messages where replica are propagated on disjoint paths.

In the low bandwidth scenario (Fig. 7.21), the ones standing at the podium are very similar to what we observed in the high bandwidth scenario as far as bytes sent and received are concerned. *Directed flooding* and *Nile* have utilized services of the majority of the devices as we look at the bytes traversed, showing the better propagation capabilities of the devices. In the case of available bandwidth, *Nile* has been able to prune the bandwidth deficient paths and is at the lower end among the three. Clumsiness of *simple replication* is evident if we see the magnitude of the sent bytes, that represents high number of unsuccessful attempts.

Single copy and two hop relays have, expectedly, utilized a tiny fraction of devices for the routing purposes in both high and low scenarios. Effectiveness of fragmentation is visible by the fact that erasure coding based methods have utilized fewer devices in comparison to others for propagation in low bandwidth but the throughput volume is higher than *flooding*. Estimation based erasure coding being the gradient based method, has a propagation volume close to directed flooding and Nile.

7.6.2 Cell tower - MIT

As we know, cell tower is a dense network where all devices are accessible from each other within a day, therefore, device participation looks different than the access point network. Let us keep in mind that contact durations in the cell tower (MIT) network are not only rapid but also very short compared to the access point (IBM) network.

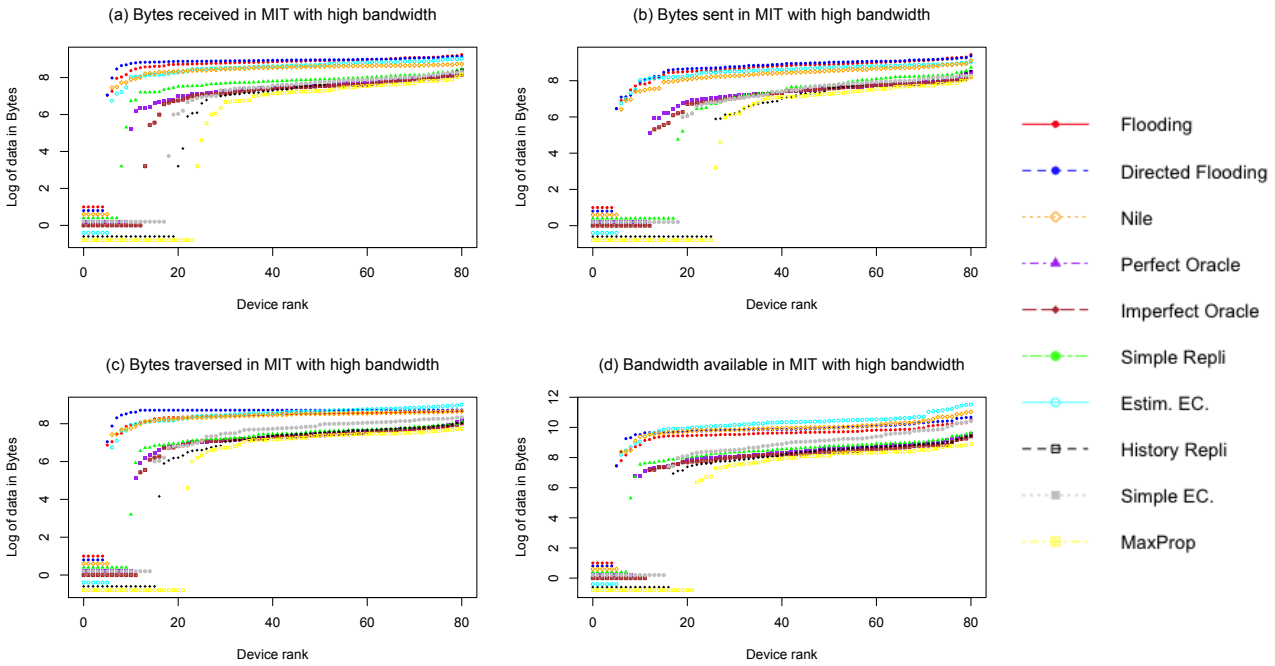


Figure 7.22: Device behavior in cell tower trace with high bandwidth

Although, there is not a big difference between behavior of *flooding* and *directed flooding* in Fig. 7.22(a) and (b), it is noteworthy that *directed flooding* is on the top of *flooding*. This shows that *flooding* is slightly suffering from lack of bandwidth even in the high bandwidth scenario and struggles to replicate messages as compared to *directed flooding*. Interestingly in Fig. 7.22(c), *directed flooding* converges to one value very early as it is able to replicate every message to each device with *flooding* attempting to achieve this aim untill the end. The unusual behavior is shown by *estimated erasure coding* method shows that it suffers from *reverse routing* and a few devices carry on exchanging a message due to the rapid change in their utility value to reach destination. This behavior is also visible in Fig. 7.22(d), whereas *Nile* is slightly more above compared to the rest of the group.

In the low bandwidth scenario Fig.7.23, *directed flooding* faces the same replication challenge as flooding due to low bandwidth. Therefore, it behaves more or less similar to flooding. Protocols that fragmentize the message utilize longer paths of the network as far as propagation is concerned without putting a flooding like burden with regard to the magnitudes of bytes sent and received.

The rest of the protocols show significant activity in the network proving the dense

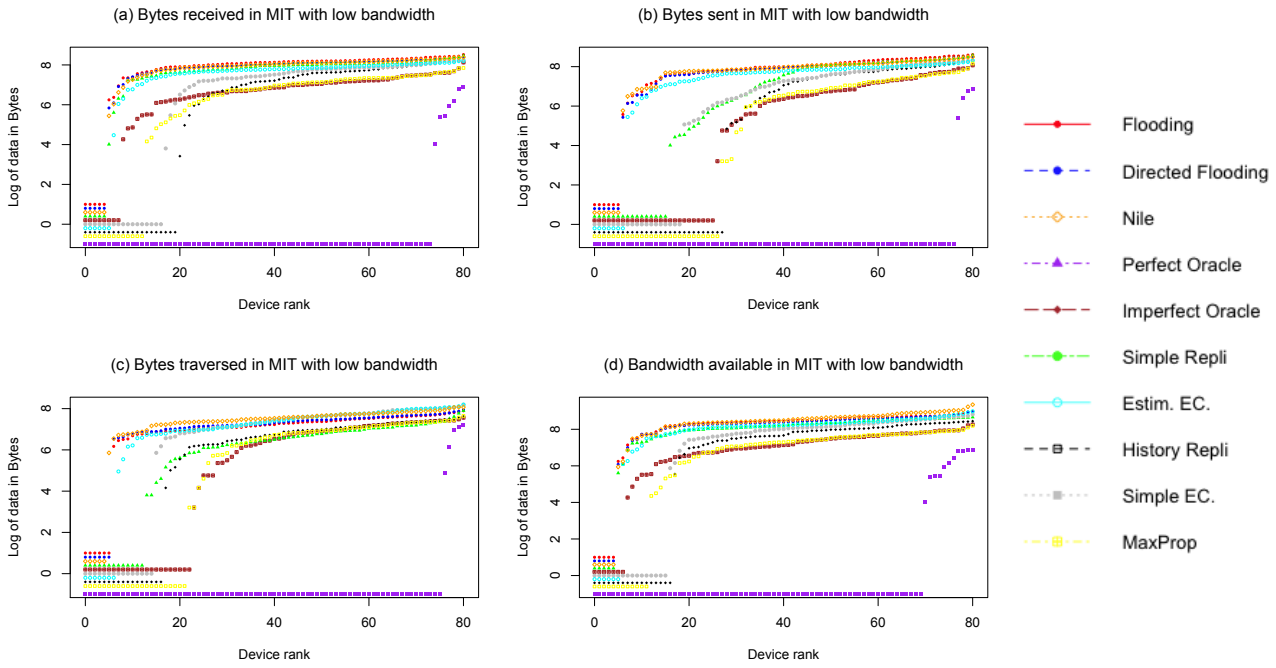


Figure 7.23: Device behavior in cell tower trace with low bandwidth

nature of the underlying network.

7.6.3 Bluetooth - MITBT

When we look at the bluetooth trace behavior in Fig. 7.24, 7.25, we observe a pattern similar to what we have seen in the access point (IBM) trace. Here, *flooding* is utilizing the maximum number of devices with high bandwidth in all measures followed by *directed flooding*. *Nile*, although it follows flooding (Fig. 7.20), shows that it is more or less part of the non-flooding group. This proves that *Nile* has been pretty choosy about selecting the next hop for messages. If we recall the performance of *Nile* in the bluetooth (MITBT) scenario in Fig. 7.5, 7.4, we can conclude that there exist a very *active core* of the network in bluetooth trace that *Nile* has relied upon to deliver performance close to *flooding*. As far as, bytes traversed is concerned, *directed flooding* employs fewer devices than *flooding* but unlike the access point (IBM) trace the throughput volume is less than that of *flooding* due to the extreme sparse nature of the network.

In the low bandwidth scenario (Fig. 7.25), the standings at the podium are very much similar according to bytes sent and received. Both, *flooding* and *directed*

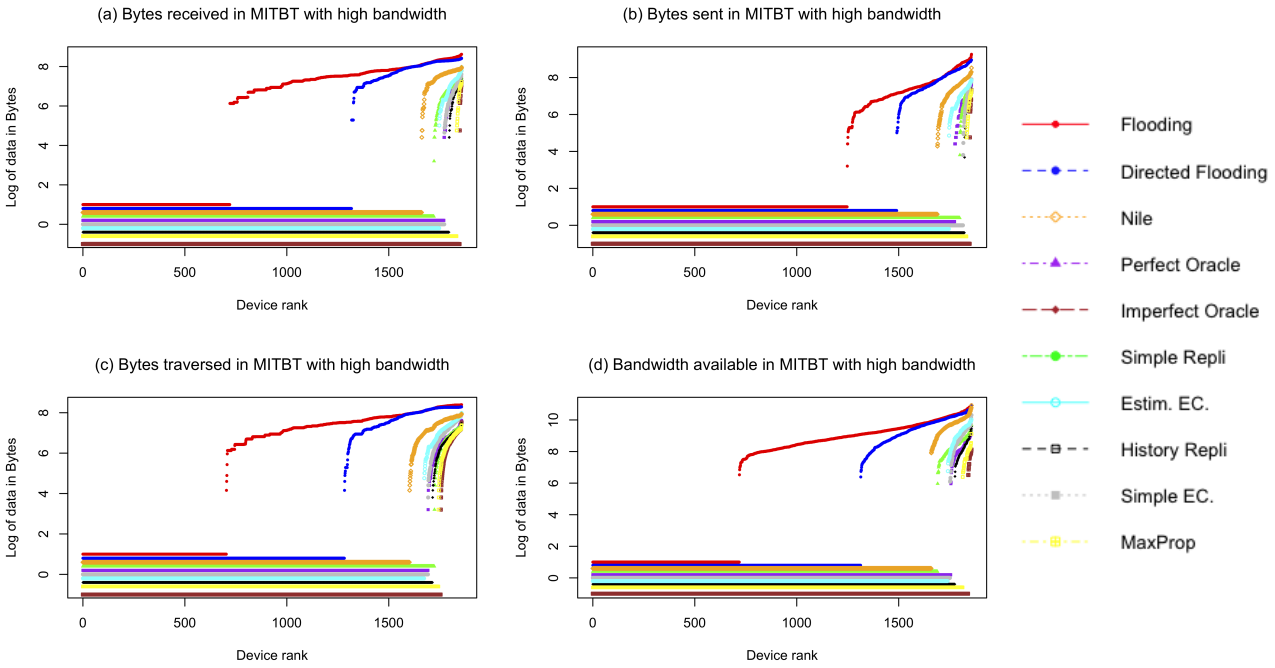


Figure 7.24: Device behavior in bluetooth trace with high bandwidth

flooding, have been pushed to their minimum activity due to the resource deficient and extremely sparse nature of the network. Yet again, *estimation based erasure coding* has suffered from reverse routing and hot potato phenomena in the active core of the bluetooth network and has shown the maximum throughput among the devices.

As we can recall from performance comparison of Fig 7.3, 7.2, MaxProp is the only gradient based method that delivered a few big messages. Thus, there is hardly much activity from the rest of the methods.

7.7 Summary

We have presented a multi-criteria performance and overhead analysis in this chapter. The variance shown by all the protocols in three networks reinforces the point that it is not easy to find one solution that fits all the scenarios. These results have not only shown the difference between finding the path and delivering the message to destination but also highlighted the conditions that turned the path calculations unreliable. The peripheral measures show those internal hidden aspects of routing protocols, which can be easily modified to reduce the overhead

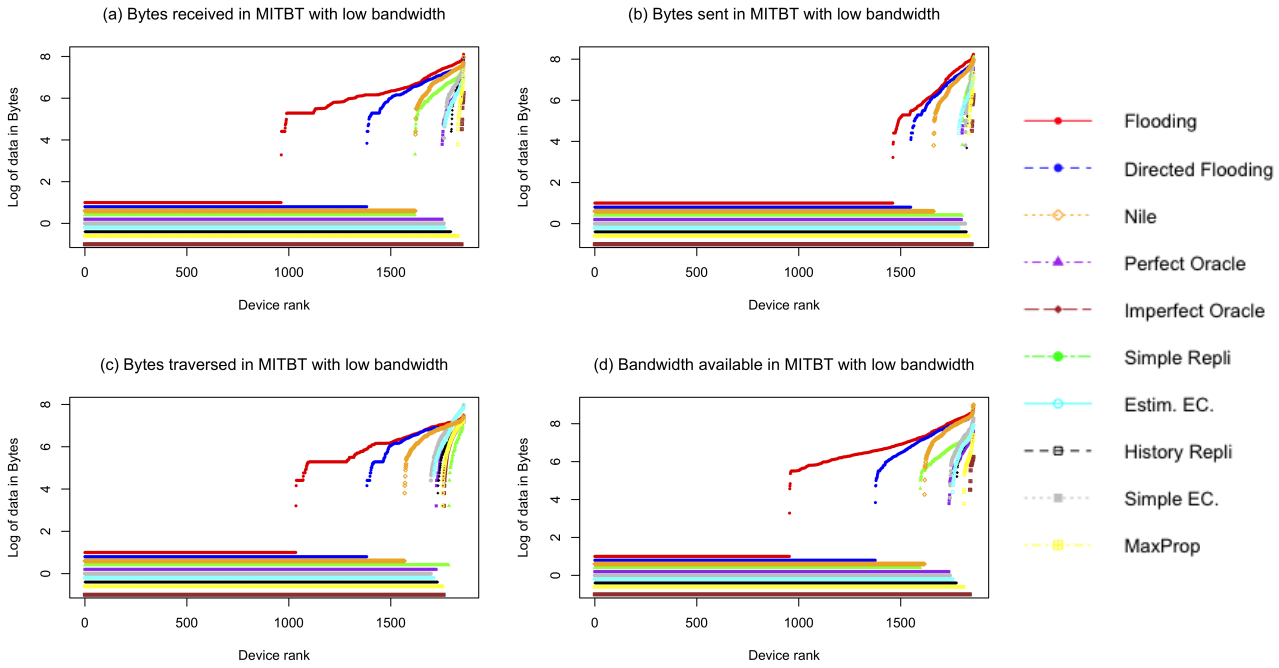


Figure 7.25: Device behavior in bluetooth trace with low bandwidth

without adversely effecting the performance. Moreover, these measures can also be beneficial for cross layer optimizations in a presence of a cluster cloud.

The overhead measurements presented in the last sections provide us with device as well as network level resource consumptions. It is interesting to relate the performance gained with the commutative network overhead. Protocol designers must keep in mind issues such as the extent of resources invested by the whole network to gain a performance edge of up to a certain threshold.

Chapter 8

Performance benchmarks in opportunistic networks

This chapter builds upon the results presented in Chapter 7 by identifying the shortcomings of flooding. It presents the reason behind the selection of flooding as a benchmark and then challenges the logic behind those reasons. This chapter also sheds light on those aspects, that in our opinion, must be considered to establish a universal and scalable benchmark.

8.1 Introduction

It is mostly argued that *flooding* can deliver optimal performance if we ignore the overhead. We argue that routing algorithms, including *flooding*, may scale differently with variation in network resources as discussed above. To asses the performance of routing algorithms in any opportunistic networks accurately, it is necessary to have such performance benchmarks that scale uniformly to the available resources and have the capability to perform better than other protocols, thus deserving to be an upper-performance bound. We identify the situation where flooding may fail to perform and based on its deficiency, we devise a new algorithm known as *EPO* that may be seen as a first step towards more accurate performance benchmarks. *EPO* utilizes concepts of oracles as presented by Jain, Fall and Patra [JFP04] but deploys them in its own peculiar way. These subtle changes ensure that *EPO* will scale appropriately with small changes in available bandwidth. Furthermore, we argue that as the networks in question are constituted from heterogeneous devices with huge variations in communication and

storage capabilities, therefore its is not easy to ignore the resource aspect while benchmarking in such networks.

8.2 Flooding as upper-bound

As discussed previously, to assume that all the participating nodes in opportunistic networks have the same communication capabilities is impractical. In wireless scenarios, traffic volume plays a very critical role [CXSN03], and thus, scalability (with regard to traffic volume as well as network size) of a routing protocol must be considered. If we assume that all the transmission among the devices is comprised of the same volume, and network size does not go through major changes, it may lead to unreliable conclusions. Ignoring the traffic volume, specially in the case of shortest path or single copy strategies may also lead up to impractically optimistic results. Lee and Geria [Lee02, PRSR06] have presented arguments that repeated utilization of the shortest path will amplify the congestion: in a random communication pattern, the nodes in the center of the network carry a disproportionately large amount of the entire traffic drastically decreases the throughput of the flows they forward or which even temporarily results in a disconnected network. This affects most long-range flows, as they have a higher probability of intersecting the central hot spot.

Given the myths about the performance of flooding in opportunistic networks, it has been customary to benchmark new protocols against flooding [WJMF05, LTZG06, MHM05]. In all the cases, a protocol is declared to be a winner, efficient or better if it performs close to flooding. As discussed in section 3.2.1, flooding is notorious for its overhead pertaining to bandwidth and local storage consumption. We would like to present a few contradictory examples where flooding fails to perform as expected and thus losing its credibility as an upper bound.

1. Let us assume that a device at the edge of a network that has very few links, wants to flood the network with a message but none of its links are good enough for the message to be replicated to any other device. If this device has other smaller messages, it has to deploy a smart scheduling technique so that those messages are given propagation priority to the next hop. Conversely, flooding has no availability of such an oracle that can predict the duration of the next contact. Another way of solving the issue could be to use the resume feature, i.e. the message is fragmented according to the available contact durations and transmission is resumed from that point onwards at which the two devices have the next contact. We have to face other questions and complications if we follow this approach, e.g. may the next device

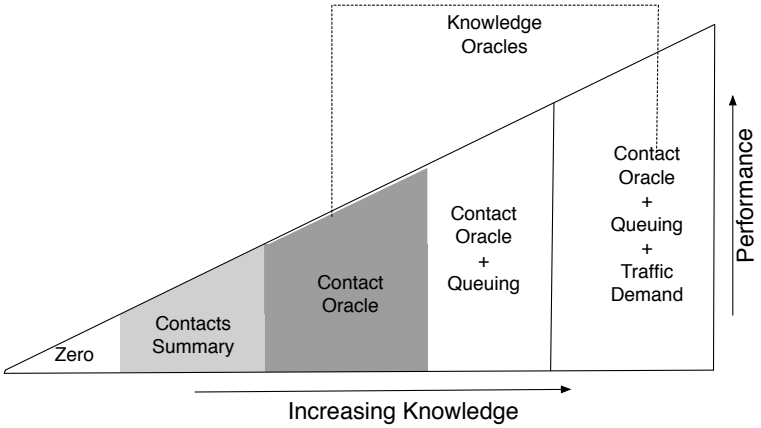


Figure 8.1: Conceptual performance vs. knowledge trade-off for different oracles. [JFP04]

propagate further the partial fragment any further? May the next device obtain the remaining fragment from another device? If yes, every device may end up with several non contiguous fragments of the message. To figure out an optimized solution, which fragment hole to fill first remains an open challenege.

2. In another scenario, if a node is located close to the center of the network and is equipped with state of the art communication hardware (large range of channels, good signal strength, high bandwidth etc), such a node can handle the throughput that flooding presents. If the premise of good hardware is non existent, even flooding may fail to reach the whole network in-spite of the availability of sufficient network resources, unless it is used in a better way as shown below. Such scenarios show that flooding can also fail miserably to perform even if the simulation account for more realistic conditions, it also shows that we need to improve on our performance benchmarks.

Based on the above arguments, we feel the need for a better benchmark for opportunistic networks that can adapt to underlying network. Jain et al [JFP04] have proposed several oracles based algorithms (Fig. 8.1), starting from *first contact* that has no knowledge to *linear programming* that utilizes the maximum knowledge including contact summary, local queuing, and traffic demand. [JFP04] has also presented a modified Dijkstra's algorithm presented in Alg. 2, which computes a path given a time varying graph. This algorithm assumes availability of the whole time varying graph that is equivalent to knowing the future patterns of devices in an opportunistic network.

Inspired by this hierarchy, we propose an algorithm, *EPO*, that assumes an oracle that has information about future contact timings, size of local queues at devices and future traffic demands of individual devices. *EPO* uses the same method as presented in Alg. 2 to find the shortest path but utilizes the computed shortest path in an unorthodox manner. The idea is to find the shortest path to the destination for one message and propagate that one message at a time through the network. Unless the message reaches the destination or no shortest path is available, propagation of other message are temporarily suspended. A question that may be asked here is, what if the shortest path is not good enough to deliver the message? It may be the case that the duration of overlap times of at least one hop are not long enough to transfer the message to the next hop and the whole chain of hops in the shortest path is disturbed.

8.3 EPO

We have experimented with two different methods in order to take care of this problem.

1. In the simple approach, the size of the message plays an important role in shortest path computation, whereby the message is propagated only on that particular shortest path that is good enough to transfer the the given message volume.

2. In the second approach denoted by suffix-X, we consider the possibility that there are several paths available but none of them is good enough to deliver the given message, at the same time their commutative throughput exceeds easily the requirements of the message. In this scenario, we propagate the message fragment on the first shortest path that is just big enough to be delivered by this path, and allocate the remaining fragment for the next shortest path. This fragmentation process continues until the whole message is delivered. If in the end, there are still message fragment left undelivered with no more paths available, all the bandwidth consuming transactions made by earlier fragments are revoked. This algorithm practically utilizes such an oracle that can foresee the commutative throughput of all the paths to any destination and propagation is attempted only when the whole message is deliverable, otherwise it is ignored.

8.3.1 How does it work?

Based on the two approached discussed above, we have implemented two different versions of the proposed benchmark, i.e. EPO and EPO-X, where both variations compute the shortest path using modified Dijkstra algorithm [JFP04]. EPO, for a message of a given size, finds such a shortest path that is good enough to deliver the amount of bytes of the message. If such a path is not found, that message is discarded, otherwise the message is propagated to destination and the next message is not propagated until first message has reached its destination. Whereas, EPO-X, presented as Alg 4, not only finds the path to the destination but also determines how many bytes can the first shortest path deliver. The message size is reduced to the threshold that the first shortest path can deliver, and then the fragmented message is delivered. *EPO-X*, then looks for the second shortest path to deliver the remaining fragment of the message and this process continues until the whole message is delivered. If it is not possible to deliver the whole message, all the transmission adjustments made for the earlier fragments of the current message are revoked. This step ensures that no unsuccessful transmission effects the delivery of the messages that are created later in time.

```
Input: Message M of size M_S and destination D_M
deliveredBytes = 0;
edgeBuffer = ∅;
while deliveredBytes != M_S do
    if shortest path P exist to D_M then
        /* Calculating Bytes deliverable by Path P          */
        B_p = MAX;
        foreach Edge E ∈ in P do
            B_p = Min(B_p, capacity(E));
            Insert (edgeBuffer , E);
        end
        Propagate M on path P to D_M;
        deliveredBytes = deliveredBytes + B_p;
    else
        /* restore the edges consumed in unsuccessful
           transmission                                    */
        restoreAll(edgeBuffer);
        exit();
    end
end
```

Algorithm 4: EPO-X propagation for a given message M.

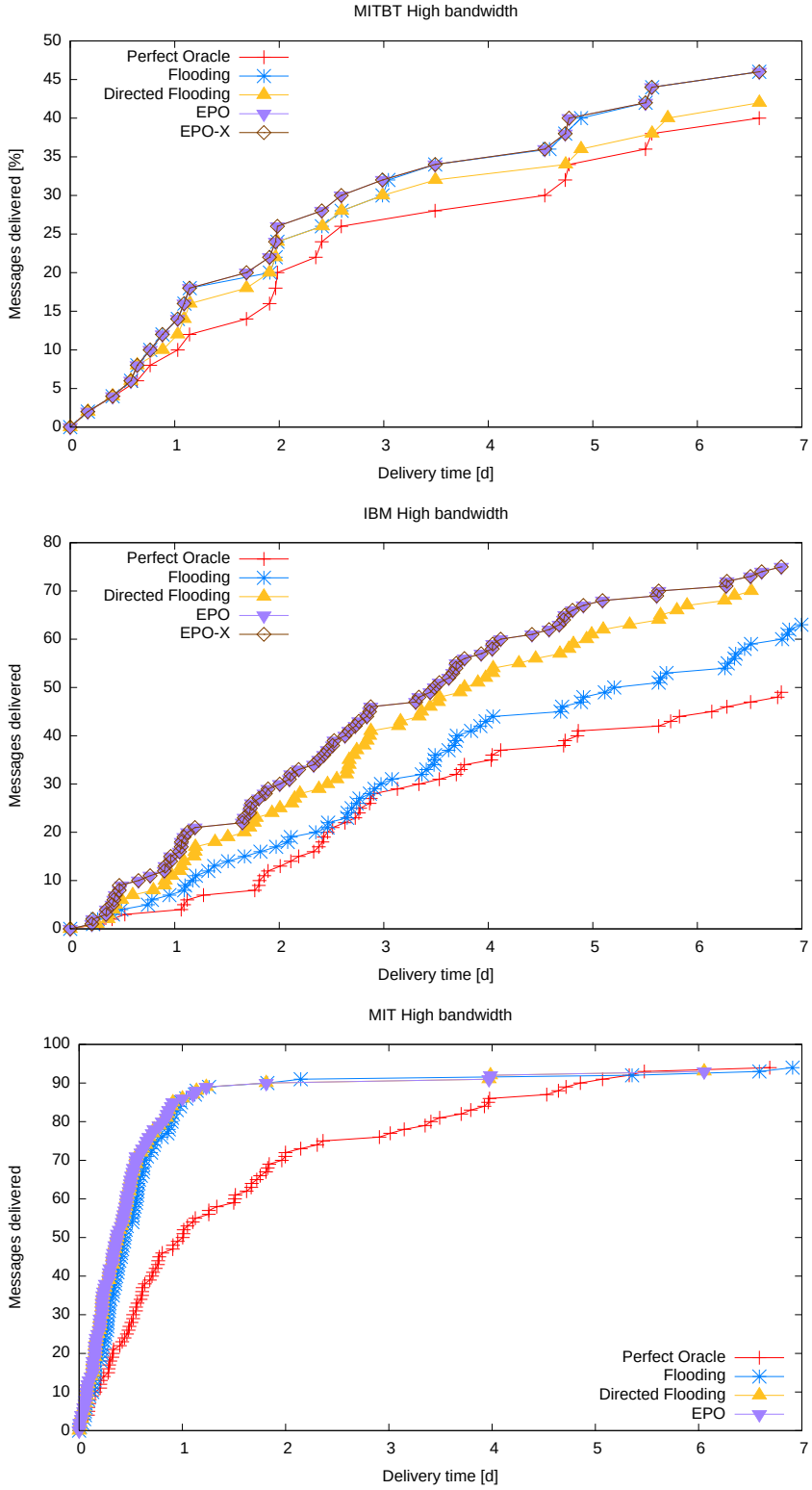


Figure 8.2: Performance in high bandwidth scenario in MITBT, MIT and IBM trace

Both *EPOs*, of course, have a greedy aspect such that they may sacrifice a vital link for a big message that is created later than a small message that happens to be using the same link. In our opinion, *EPO* is not only a better replacement for the upper bound, but will also deliver accurate information about the network characteristics, such as contact frequencies, contact durations, number of paths, edge stability, etc.

8.4 EPO results

When we analyze the two sets of results presented here in Fig. 8.2 for high, and in Fig. 8.4, Fig. 8.3 for low bandwidth, the difference in the delivery ratio in size as well as the number of messages prove the hypothesis that traffic volume does play a significant role in the case of opportunistic networks. Fig. 8.2 shows that flooding is not always the winner even in the presence of abundant bandwidth, whereas a slightly intelligent scheme like *directed flooding* can outperform flooding in both low and high bandwidth scenarios. Moreover, in the case of high bandwidth (Fig. 8.2), plots for both *EPO* are similar, which suggests that almost all the first shortest paths had delivered the messages and the effects of transmission overlaps (two messages using the same link) have been minimal. *Perfect oracle* ended up to be last because there is no guarantee that a second shortest path exists by the time propagation of the message is halted on one of the hops due to lack of bandwidth. The sparsity of the MITBT network is visible by the fact that Flooding delivered a mere 24 messages in MITBT and approx. 70 messages in the IBM trace while the MIT being the dense of all the traces, shows that flooding delivers 90% messages within a day.

We have not presented the results related to *EPO-X* in the MIT trace in Fig. 8.2 because all of the participants have performed so close to each other that it was not interesting to include *EPO-X* in that plot.

In low bandwidth plots (Fig 8.4 and Fig. 8.3), resulting differences between *EPO* and *EPO-X* are more distinct with *EPO-X* being superior. Although, *EPO* has performed better than flooding since it has not been able to find paths for several huge messages. In contrast, *EPO-X* has exploited the cumulative available bandwidth through all the path to the destinations. It is interesting that if we compare performance of *EPO-X* in high and low bandwidth cases, the cumulative delivery ratio is somewhat similar. The only difference is the delay incurred by the messages as the buffer duration of messages has been long. The bigger the message, the more round trips are needed to complete the full message transmission to the next hop. In the sparse case of the MITBT trace, we notice the greedy nature

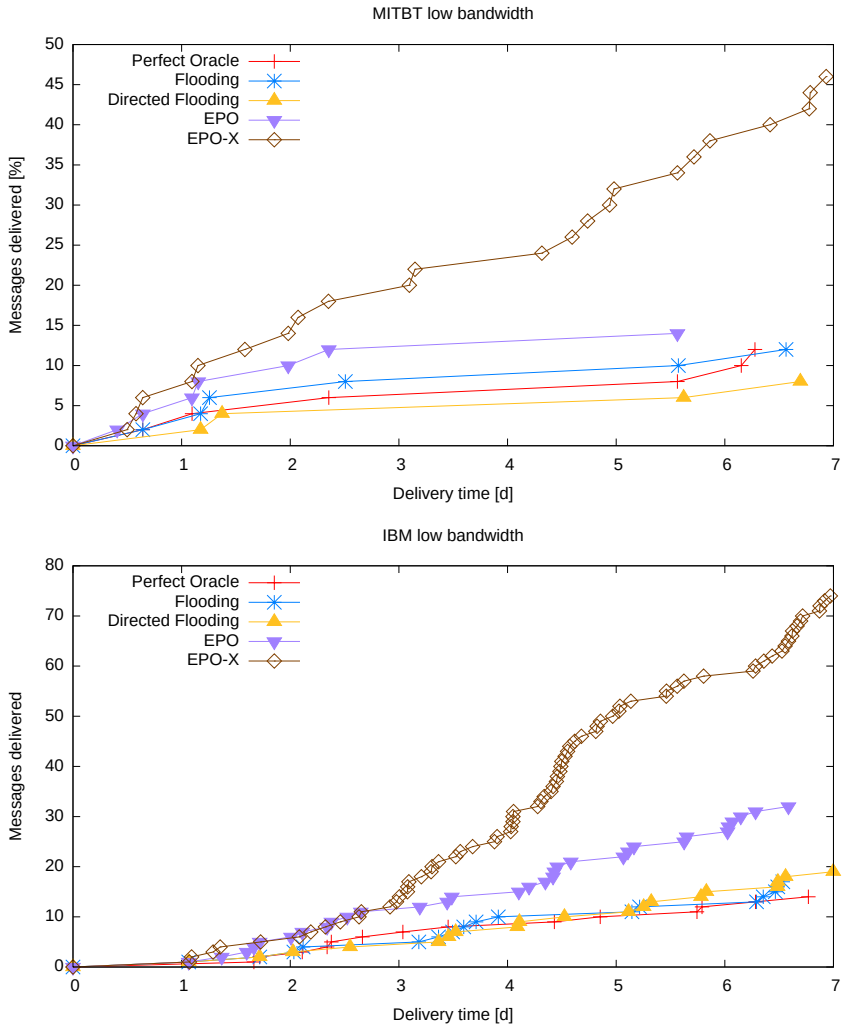


Figure 8.3: Performance in low bandwidth in MITBT and IBM trace

of EPO as it has not performed better than *directed flooding*. The explanation is that a few larger messages that are delivered early on by *EPO*, have consumed the links that were pivotal for the delivery of some messages that are generated later.

8.5 Conclusion

In this chapter, we have presented a different approach to simulate flooding that emphasizes its shortcoming by using messages of variable sizes. The above discussed results show that the behavior of opportunistic networks varies with the bandwidth variations. Moreover, *flooding* cannot be the winner whenever bandwidth is scarce and thus it is not a suitable upper bound for such networks.

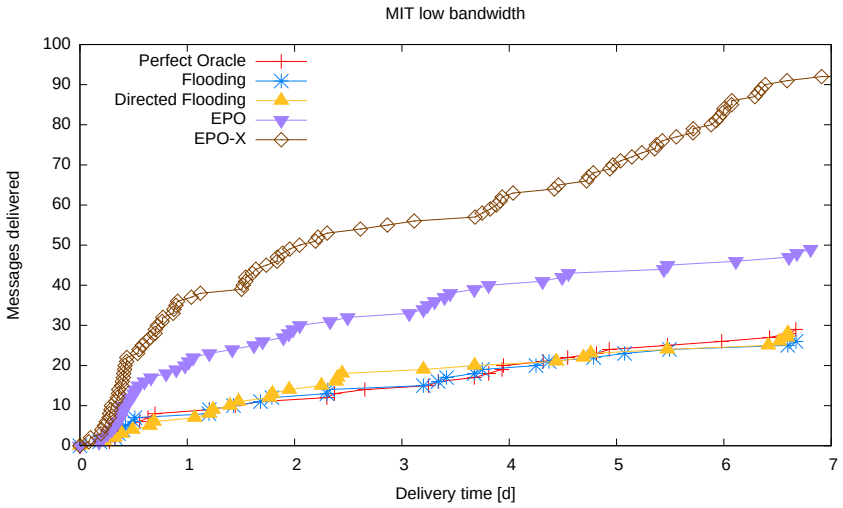


Figure 8.4: Performance in low bandwidth scenario in MIT trace

Furthermore, we have provided insight into two variations of a technique that cannot only be considered as a better upper bound (and can be improved further on) but that also highlights the hidden aspects of the underlying network. Strategies like *EPO* exploit all the available resources and help identify the bottlenecks in opportunistic networks, paving the way for the development of realistic and efficient routing protocols. Aspects, such as commutative throughput on the diameter of the network and identification of bottlenecks in clusters, will assist us to better understand such networks. In the future, our knowledge enhanced by these finding, will play an important role in developing suitable routing protocols.

Chapter 9

Prediction quality of history

> "A process cannot be understood by stopping it. Understanding must move with the flow of the process, must join it and flow with it." – The First Law of Mentat, in Frank Herbert's Dune (1965)

This chapter analyzes the effects of aggregation on the precision and effectiveness of routing in opportunistic networks. As we have seen in earlier chapters, opportunistic networks suffer from the lack of information on which suitable routing decision can be made. Even if this information is available, we are not sure about its accuracy. We create several hypothetical scenarios to identify the effects of routing decisions based on potentially imprecise and inaccurate information on the performance. We learned that history can predict paths accurately in dense networks, but we need to improve our methods for path quality. In sparse networks, statistical methods must be employed to improve the history gathering process because predictions provided by simple history are not reliable.

9.1 Introduction

In opportunistic networks, individual devices do not only have variable capacities but also the traffic is less predictable. It is well known that wireless communication suffers severely from traffic congestion and that in case of a bottleneck, path recalculation can be a resource expensive process. As already stated, traffic congestion in opportunistic networks, does not only create problems for those messages that are directly involved but it also reduces the delivery probability of those messages that are sharing the same path. It is therefore even more important for oppor-

tunistic networks to have traffic metrics that are as accurate as possible, so that congestion can be avoided. The delay encountered by the messages in opportunistic networks ranges from a few minutes to several days depending on the dense/sparse nature of the network. As the wireless communication is inherently dependent on the external environment such as distance, obstacles, interference, etc, we cannot assure the accumulated traffic measures to be precise in such scenarios.

Every routing protocol deploys its own way of collecting the history that is distinct with respect to several aspect including (a) what kind of history information is collected, (b) how frequent is it collected, and (c) what measures are taken to maintain the minimum device storage consumption. Moreover, due to hardware limitations, the size of routing information must be limited, which introduces inaccuracies in the measures. Consequently, obtaining accurate and precise traffic measures for participating devices is a great challenge. One may expect that more accurate paths and traffic measures will lead to better message delivery. The information that is considered relatively useless due to less frequent use or expired lifetime, is discarded to keep the consumption of computation and storage resources to an acceptable level. The delay encountered by the messages in opportunistic networks ranges from a few minutes to several days depending on the dense/sparse nature of the network.

In addition to the above, delays and device mobility make the access to information like network topology and traffic volume, very difficult. When we look through the available routing protocols in the research arena, we can find big variances in the motivations, approaches and methodologies used. Jain, Fall and Patral [JFP04] have proposed several oracles with future insight. Although, such methods are unrealistic, they can help to understand the nature and behavior of underlying networks, such as to reveal the hidden complexities of the propagation of messages that are not perceivable otherwise. On the other hand, the realistic protocols mostly consist of history based methods that utilized the past behavior shown by devices to predict their future pattern. Motivated by work in [JFP04, IMA11b] we have performed max-flow computations on three different opportunistic networks and then compared the results by performing max-flow computations on history generated by the respective networks. We have varied the history information available to devices in the network to observe its effects on the max-flow throughput.

9.2 Why max-flow?

Typically, network analysis requires finding a maximal-flow solution to identify bottlenecks when there are capacity constraints on the arcs. The maximum flow

problem is structured on a network, however, the arc capacities or upper bounds, are the only relevant parameters. Given a graph where one vertex is considered a source and another is the sink, some object then flows along the edges of the graph from the source to the sink. Each edge along the path is given a maximum capacity that can be transported along that route. The maximum capacity can vary from edge to edge, in which case the remainder must either flow along another edge towards the sink or remain at the current vertex for the edge to clear or to be reduced. Thus, the goal of the maximum flow problem is to determine the maximum amount of throughput in the graph from a source to the sink. Readers interested in background and theoretical proofs of problems related to max-flow may consult [AMO93].

Opportunistic networks can be seen as good examples of distributed systems [LKG06], which can be simulated and analyzed with the help of oracles that have the capability of delivering different kinds of network measures without delay, throughout the network. Mechanisms that provide information to predict the device and traffic behavior, and which are difficult or impossible to gather in realistic scenarios, are known as *oracles* [JFP04]. Provided that the information is accurate, strategies can make very efficient use of network resources by forwarding a flow along the best path. Jain, Fall and Patra [JFP04] have presented classification of several oracles based on the extent of information they can deliver. As depicted in the Fig. 8.1, a zero knowledge protocol could be one that forwards the messages randomly or to whomever receives it first. The contact summary gives insight into the past contact frequencies and the more frequent contacts receive priority over the others. The most complicated oracle is the one that can predict the exact timings of contacts, volume of traffic in local queue of devices, and traffic demand. One can safely assume that the higher the accuracy, the less likely it is to actually construct such an oracle in the real world. The information that is necessary for making intelligent routing decisions, and which can be constructed in the real world, lies between the two extremes, the zero knowledge of network and the full knowledge of timings of node contacts with future traffic demand.

One can safely assume that the higher the accuracy, the less likely it is to actually construct such an oracle in real world (Fig. 8.1). The information that is necessary for making intelligent routing decisions and that can be constructed in the real world case lies between the two extremes of zero knowledge of the network and full knowledge of the node contacts as well the traffic volume. As already stated, we can also find several other opportunistic network routing protocols that are based on some type of utility function [RK02]. Such mechanisms assume that nodes in an opportunistic network tend to visit some locations more often than others, and that node pairs that have had repeated contacts in the past are more likely to have

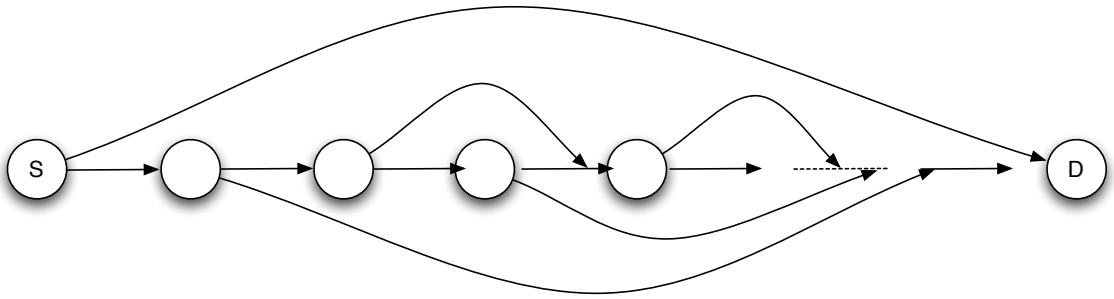


Figure 9.1: Propogation method [WH01]

contacts in the future. A probabilistic metric called *delivery probability* P(A, B), estimates the probability that node A will be able to deliver a message to node B. Additionally, we can find other examples where geographical location and time is also considered part of the history taking advantage of spatial and temporal factors in routing [RK02, BBL05]. Whatever may be the metric of history, most of these routing protocols work on the realistic premise that node mobility is not truly random.

One of the issues with history gathering is to define the granularity as well as the timespan. Opportunistic networks consist mostly of mobile phones or other portable devices. Several methods have therefore been proposed to summarize the history in such a way that it is does not pose a burden on computational and storage capabilities of the device. Moreover, it is argued that better prediction can be obtained by recent history. Lindgren, et al. [LDS03] uses an aging factor associated with probabilistic computations where old values are weighted less than the new ones. Other examples like [JLS07] use a mechanism similar to the sliding window to discard the "obsolete" history. Wang [WJMF05] has argued in favor of using the most significant r history readings where significance can be defined on a case-to-case basis. We can also find examples where Kalman filters have been used to reduce the noise (irrelevant history) to assure better prediction [CMMP08].

9.3 Adaption of max-flow computation for an opportunistic network

One may think of several ways to adapt the maximum flow problem for opportunistic networks. The idea here is to obtain maximum traffic that can flow in an opportunistic network among random source and destination pairs. We have used the contact oracle [JFP04] to obtain the *maximum flow* between two nodes in the network because *maximum flow* solution can only be obtained, if we have

the knowledge about the whole network. Since the contact oracle can predict the timings of future contacts, the *contact oracle max-flow* solution will give us the maximum volume transferrable between these selected source and destination (src,dst) pairs. As the contact oracle can deliver the exact contact timings of devices in an opportunistic network, we only need a modified Dijkstra's algorithm citeJain2004Routinga to adapt the computation to obtain *maximum flow*.

When we think about computing *maximum flow* with the help of history, we have a choice to obtain the path to destination in two different ways.

1. **With contact summary:** First, we rely on the contact summary oracle to compute *maximum flow*. As the contact summary oracle does not posses accurate node contact timings in the network, *maximum flow* obtained by contact summary oracle is expected to be less than that of the contact oracle. As stated earlier, since several methods are used to limit the size of information provided by the contact summary oracle, we analyze how these methods reduce the traffic flow with the gradual decrease in network information accuracy. The information size can be reduced by aggregating the measures over a defined timespan. Here aggregation refers to the mean values of the contact delays and their durations, thus *maximum flow* can be computed by extrapolating these aggregated measures. We use these extrapolated measures to forecast for the upcoming timespan.
2. **Without contact summary:** The second way of computing a path for computing maximum flow is by removing the contact summary oracle. Thus, nodes themselves are responsible for sharing the aggregated delays and other measures among each other. This means that information accuracy is further decreased in contrast to the previous scenario of the contact summary oracle because the information flow through the network is dependent on node mobility. A source can calculate the shortest path using this shared information only if another network node has already shared the measures about the destination with the source.

Thus, we *reformulate* the max-flow problem in an opportunistic network such that we compute the maximum traffic that can flow between a source and a sink using the mean of the variant capacities of all the paths over a defined timespan. Following are the issues that must be addressed to adapt the maximum flow computation for time varying graphs like opportunistic networks.

1. We must define the terminal condition of *maximum flow computation al-*

gorithm for opportunistic networks. The maximum flow algorithm does not terminate until the residual capacity of all the paths to the sink is reduced to zero. As we are dealing with time varying graphs, we have limited our analysis by setting up a timespan during which the destination must be accessed directly or indirectly by the source.

2. Another interesting question is how to assure this terminal condition when we do not posses accurate information about the network. As stated earlier, both history variants of *maximum flow* computation have to predict the paths and these computed paths cannot guaranteed to be correct. In the case, we have an incorrect path, one of the edges on the incorrect path must be either labeled *invalid* or *disconnected* temporarily so that Dijkstra's algorithm can deliver the next best path. At the same time, the residual capacity of the remainder of this path must be kept intact so that it may still be utilized for further maximum flow computation.

3. Another issue is how strictly a path may be followed when we have inaccurate information. We have used a forwarding strategy that does not follow the computed path in a very strict sense. The flow can be attempted to be propagated to any hop that may bring it closer to destination as shown in Fig. 9.1. Due to this aspect, the labeling of an edge as *invalid* is further complicated because we cannot identify the most suitable edge to be declared *invalid*, unless we use another oracle that can tell us which edge was either mistimed or failed to be realized at a particular point in time. This factor introduces more inaccuracy to history based computation of maximum flow and remains an open issue.

9.3.1 Analyzed Strategies

Based on the above discussed issues, we have simulated the following three different max-flow strategies for the sake of comparison. However, it is important to mention that in the following experiments, maximum flow computations are not performed simultaneously for all of the source and destination pairs. We compute maximum flow between the first src,dst pair and then we move back in time to select the next pair iteratively. Similarly, we go back in time to compute a new path if the given path provided by the contact summary oracle is not correct. It is also important to mention that our implementations of the below described strategies are of greedy nature with respect to delay (quickest path first). Therefore, it is possible that a better path with longer delay is sacrificed for a quicker path. We have decided for the greedy implementation because one of the terminal condition of the algorithm

Input: A source, destination pair, (src,$dest$)
Output: Maximum flow based on available oracle
$deliveredBytes$ = 0;
$edgeBuffer$ = $\emptyset$;
while *Contact oracle has shortest path* P *to* $dest$ **do**
    `/* Calculating Bytes deliverable by path` P `*/`
    B_p = MAX;
    foreach *Edge* $e \in$ *in* P **do**
        B_p = Min(B_p, capacity(e));
    end
    Propagate B_p on path P to $dest$;
    $deliveredBytes$ = $deliveredBytes$ + B_p;
end

Algorithm 5: Max-flow calculation with help of contact oracle for a given src,dest pair.

is to find all maximum flows that can access the destination within the described timespan. Without this condition the computation overhead is too high.

Contact oracle max-flow This max-flow computation is based on contact oracle [JFP04] where paths provided by oracle iteratively are consumed as long as there exist no such path to the destination that can deliver the traffic within the desired timespan. Paths are computed between 100 pairs of source and destinations with the help of the variant of the Dijkstra's algorithm. Once the src,dst pair is selected, we follow the mechanism similar to what has been presented in [IMA11b] described here in Alg. 5. This way, max-flow computed between the first pair of src,dst will affect all the next max-flows to be computed later if they share at least a common edge.

History-based max-flow with path oracle This strategy is a variant of the contact summary oracle where we are aware of the delays as well as the mean throughput of the obtained paths. It is imperative to recall that the obtained paths with these measures are used for prediction purpose. These measurements of delays and throughput are accumulated over a defined timespan in the following way. For each encounter between two directly connected nodes X and Y as shown in Fig 9.2 as *direct link*, the history information is shared as follows.

a. Node ID of Y if it is a first contact between X and Y.

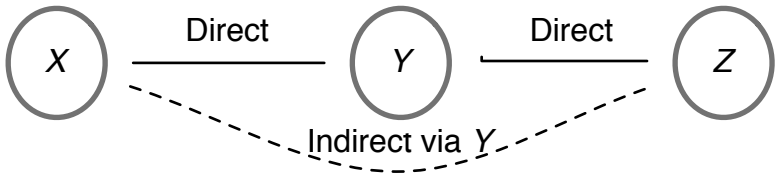


Figure 9.2: History calculation

b. Update or insert mean delay ϵ_{XY} encountered for transmission from X to Y. ϵ_{XY}, includes the time durations, data stay in the local queues as well as inter-hop transmission times.

c. Update mean number of bytes that could be delivered B^v_{XY}, i.e. link throughput.

As discussed earlier, contact summary oracle may find a path to the destination but cannot ensure the realization of this path in the network trace. In case this oracle predicts the path correctly, the probability of correct path throughput prediction is low. Thus, a flow is initiated with the volume that is prescribed by the oracle and this volume is decreased during the propagation if it is realized that the path throughput is less than the volume. This way, different edges on the path may suffer different bandwidth consumptions, which may bring inconsistencies to the results. As there may be several instances of a contact between two devices, we must extrapolate the edges of contact summary oracle proportionally to the frequency of contacts between two devices. These extrapolated edges are consumed in a fashion similar to the mechanism presented in [FF56] where aggregated residue capacity on each edge is reduced according to the minimum aggregated capacity of the whole path.

History based max-flow without path oracle This strategy has no access to any oracle. Devices themselves gather and share history information for a defined timespan. This means we must extend our history gathering method so that direct as well as transitive contacts can be preserved. In addition to the sharing of direct contact history information as described previously, information about indirect contacts is shared in a transitive manner as follows:

For an indirect neighbor or *twohop link* as shown in Fig 9.2, device X inserts augmented information in its routing table about the device Z accessible via Y,

a. Node ID Z if X has no earlier record of Z via Y.

```
Input: G = (V,history(E)), src, dest, w(e,t)
Output: L
Q ← {V} L[s] ← 0, L[v] ← ∞ ∀ v ∈ V s.t v ≠ src.  while  Q ≠ { } do
    Let u ∈ Q be the node s.t L[u] = min_{x∈Q} L[x] ;
    if  u = dest then
        exit;
    end
    Q = Q ← {u};
    for each edge e ∈ E , s.t e = (u,v) do
        if ContactCount(e) > 0 then
            if  Using history oracle then
                if  L[v] > (L[u] + w(e, L[u] )) then
                    L[v] ← L[u]) + w (history(e), L[u] );
                end
            else
                if  L[v] > (L[u] + w(L[u], Dist(dest))) then
                    L[v] ← L[u] + w(L[u], Dist(dest));
                end
            end
        end
    end
end
```

Algorithm 6: Dijkstra's Algorithm modified to use history costs. s and d are the source and destination node respectively. The path calculation is performed on the edges provided by history(E). Using oracle, we can calculate cost calculated provided by history to every node but without oracle, we need a destination for which we need to calculate the costs.

b. Update or insert mean delay from X to Z,

$$\epsilon_{XZ} = \epsilon_{XY} + \epsilon_{YZ}$$

c. Path throughput X, Y, Z,

$$B^v_{XZ} = min(B^v_{XY}, B^v_{YZ})$$

A source computes maximum flow to the destination, based on these measurements of mutually shared mean delay among the devices. The key difference between *with* and *without path oracle* is that the history based max-flow without path

oracle does not assure a path, unless the source has been able to obtain the mean delay to the destination from the mutual contacts of the middle hops. Thus, the mobility of a network plays an important role for the dispersion of this information. Theoretically speaking, if nodes in a network move very frequently with the speed of light, we may end up having the same measure as we had in the case of *with path oracle* variant.

Another question is how to proceed when we have a false prediction. It is possible that the first path is a bad prediction but there still may exist valid paths that show longer delays. We have to somehow eliminate the current invalid path so that the algorithm can find the next best predicted path. For the sake of simplicity, we disconnect one edge on the invalid path that has failed to be realized. This means, we disconnect the edge between the last hop successfully taken and the next available hop in the path as shown in Alg. 7. As our forwarding mechanism does not follow *strictly* the given path (Fig. 9.1), disconnecting the last hop and next hop may give unexpected results.

9.3.2 Amount of history

As discussed in section 9.3, a variety of methods can that propose the use of history to predict the behavior of devices can be found in literature. In the light of these examples, we have introduced a window period variable w. This variable represents the timespan over which history is accumulated. The two values used for w are 10 and 14; that represents the corresponding timespan in days. To separate the two issues of *accuracy loss due to information aggregation* from *quality of prediction*, we have simulated both variations of history based max-flow computation on the same timespan from which history has been computed as well as for 7 following days after the history timespan has elapsed. This configuration is represented in Fig. 9.3, where two values of $w = 10$ and 14 are represented with the corresponding prediction time period.

We have used a threshold of path delay of $x = 7$ days, i.e. all paths showing longer than 7 *mean delay* are discarded. The remainder of the simulation parameters are summarized in table 9.1. It is important to mention here that we have utilized the same history for $x = 10$ as computed for opportunistic routing simulations presented in [IMA11a]. This aspect resulted in some unexpected outcomes that we are not able to address here due to time limitations.

```
Input: A source, destination pair, (src,dest)
Output: Maximum flow based on available oracle
deliveredBytes = 0;
edgeBuffer = ∅;
while History oracle has shortest path P exist to dest do
    /* Calculating Bytes deliverable by Path P                    */
    B_p = MAX;
    foreach Edge e ∈ in P do
        B_p = Min(B_p, capacity(e));
        Insert (edgeBuffer , e);
    end
    Propagate B_P on path P to dest;
    if delivered then
        deliveredBytes = deliveredBytes + B_p;
        foreach Edge e ∈ in P do
            if B_p = capacity(e) then
                Decrement ContactCount(e );
            else
                LinkCapacity(e) = LinkCapacity(e) − B_P/ContactCount(e);
            end
        end
    else
        /* restore the edges consumed in unsuccessful
           transmission                                            */
        restoreAll(edgeBuffer);
        /* disconnect the failed path                              */
        disconnectEdge(lasthop,nexthop);
    end
end
```

Algorithm 7: Max-flow calculation with help of history oracle for a given source, destination pair.

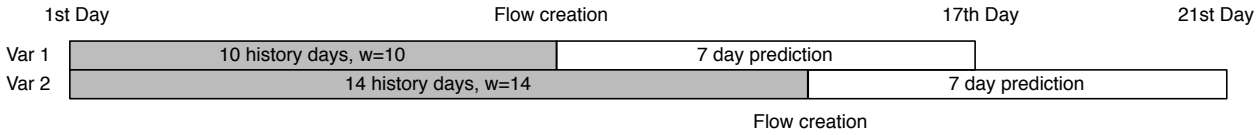


Figure 9.3: History window and prediction configuration

Source, sink pair count	100
Flow computation window length	7 Days
History window length	$w = 10$ (1st - 10th) Days
	$w = 14$(1st to 14th) Days
Valid path delay	$x = 7$ with path oracle
	$x = 10$ without path oracle
Bandwidth (low)	100 kiB/s
Bandwidth (high)	10,000 kiB/s

Table 9.1: Simulation parameters

Strategy	Denotement	Description
Contact oracle max-flow	CMF	Based on contact oracle
History based max-flow with path oracle	HMFP	Based on aggregated history and oracle is responsible for information sharing
History based max-flow without path oracle	HMF	Based on aggregated history and devices are responsible for information sharing

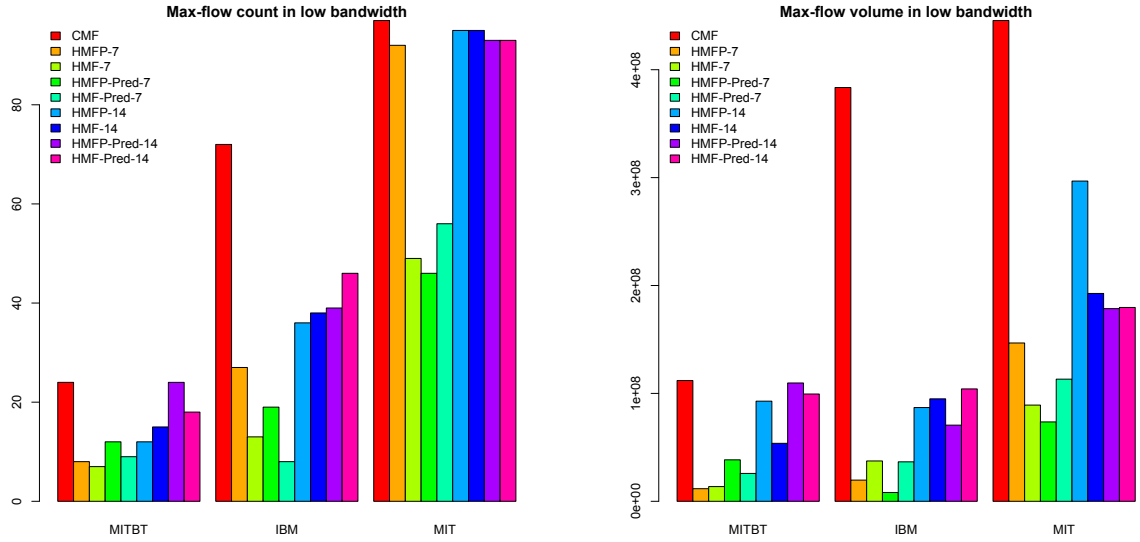
Table 9.2: Algorithm denotement and description

9.4 Results discussion

For the sake of presentation, we have assigned several denotements to our strategies that are presented in Table 9.2. We have used the suffix-*Pred.* to represent the *predicted* max-flow. This max-flow computation is performed on the timespan of 7 days that starts after the history timespan has elapsed; is shown as unshaded rectangle in Fig. 9.3. Moreover, each denotement is followed by a number that represents the value of history window size, i.e. w.

When we look at all the plots in Fig. 9.4a, 9.4b, 9.5a and 9.5b, CMF has delivered the maximum amount of bytes as well as connected the maximum number of node pairs. The sparsity of the bluetooth (MITBT) trace is evident here where CMF has connected a mere 30 node pairs as shown in Fig. 9.4a. In cell tower (MIT) and access point (IBM) cases, it has connected 98 and 70 src,dst pairs receptively. We also notice that the difference between high and low bandwidth is visible only in the traffic volume while, the src,dst pairs connected count is more or less the same.

Another observation that is true for all the 4 figures, is that an increase in window size w has a positive effect on almost all of the HMFs variants. Moreover, we see a decrease in both max-flow computations, i.e. volume and count, in almost all



(a) src,dest count in low bandwidth **(b)** volume in low bandwidth

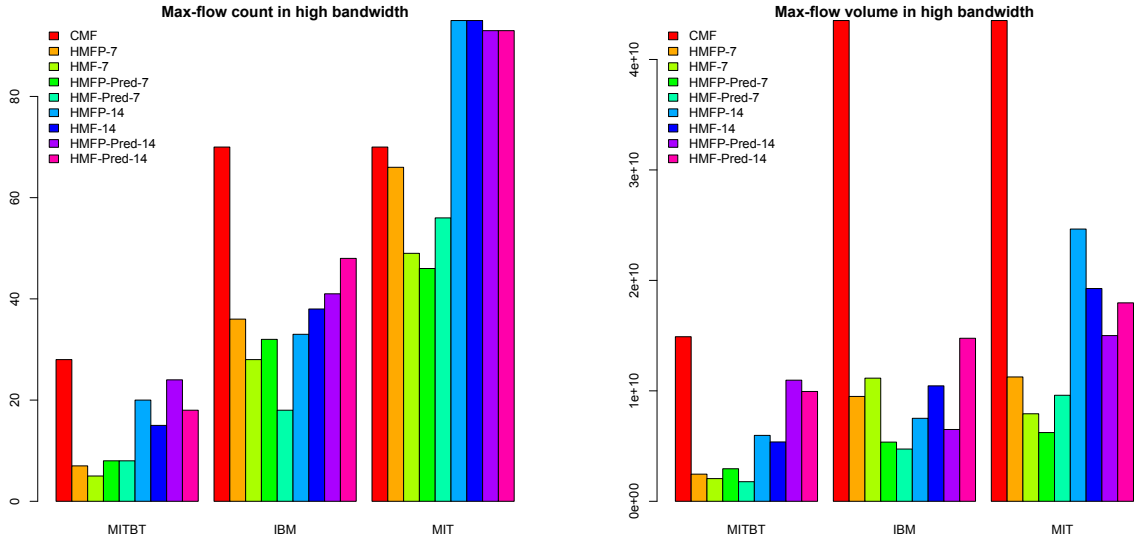
Figure 9.4: Maximum flow results for low bandwidth scenario

cases, when we look at *pred.* variants. There are a few exceptions to this statement that we discussing later.

When we analyze the two sets of results presented here in low bandwidth (Fig. 9.4a and 9.4b,) CMF has delivered maximum bytes as well established contacts between maximum pairs of source and destination in all three traces. When we look at all of the HMFs-7 and HMFP's-7, we see a drastic decrease in both aspect, i.e. volume and count. We did not expect such a drastic decrease, particularly not in the bluetooth (MITBT) trace. It is also interesting to observe that HMFP has delivered slightly fewer bytes than HMF but has established one more path. Although, HMF is supposed to have more inaccurate information in comparison to HMFP, due to artifacts explained later, it has delivered a few extra bytes in the bluetooth (MITBT) trace.

On the contrary, HMFP has delivered more bytes and connected more src,dst pairs in the bluetooth (MITBT) high scenario (Fig. 9.5a). Where HMFP has delivered more data as compared to HMF, the IBM trace has shown the expected behavior as far as volume is concerned (Fig 9.5b). This behavior is consistent for the IBM trace in the high bandwidth scenarios however, it is not true for the MIT high bandwidth scenario.

The reason behind these behaviors is that we used the same history for $w = 10$ as we have used for the routing protocol Nile [IMA11a]. Nile not only needs the



(a) src,dest count in high bandwidth **(b)** volume in high bandwidth

Figure 9.5: Maximum flow results for high bandwidth scenario

contact delays but also the traffic volume being transmitted on the paths. As this traffic load has been obtained by simulating *directed flooding* [IMA11a], mean path delays of both window sizes are not necessarily the same. Moreover, in high bandwidth scenario, *directed flooding* has propagated more messages farther in the paths. This traffic load has caused a variance in path delays between low and high scenarios because more replicas have reached farther in high bandwidth scenario. Both these factors alter the mean path delays in the network, and this difference causes different paths to be computed by HMF's as well as HMFP's in comparison to other scenarios, i.e. low bandwidth and $w = 10$.

The behavior of the dense MIT network has been as expected because the history window size of 14 has ensured that information about all paths is spread all over the network (Fig. 9.4a). All strategies with history window $w = 14$ have connected almost all the pairs that CMF has connected. As far as the volume is concerned, (Fig. 9.4b), we can see that the decrease in readings, when compared to CMF, is due to information loss to the aggregation.

A few inconsistencies that have been observed in the results are due to several reasons. As stated earlier, our implementation of the contact summary oracle is greedy with respect to delay (the quickest path first), which means that the utilization of a *quicker low throughput path* can affect the max-flow computation of a *slower high throughput path*. Moreover, the artifact of the forwarding scheme depicted in Fig. 9.1, has also resulted in interesting side effects. As discussed, the

forwarding scheme combined with the mechanism of removing the invalid path can cause a useful edge to be disconnected. This is due to the fact that we cannot exactly identify the edge on the traversed path that is responsible for the failure without a another oracle. Another reason is the variable consumption of edges during volume propagation, in which all history based max-flows may consume less bytes in later edges than the former ones because of the dynamic reduction of flow volume.

When we look at the volumes delivered by CMF in the high bandwidth scenario (Fig. 9.5b), volume delivered in the case of the IBM trace has surpassed all the others, whereas the contact count is higher in the MIT case (Fig. 9.5a). Surprisingly, the contact count for HMF is far lower when we compare MIT low and high bandwidth cases. This is because the two HMFs, i.e. low and high, have connected several same src,dst pairs through different paths because of congestion experienced in low bandwidth. Thus, the larger volume delivered by HMF has caused the history oracle to disconnect the network and the contact count has dropped (Fig. 9.5b).

9.5 Recommendations

These max-flow based experiments can been seen as a first attempt to analyze the variance in the behavior of three opportunistic networks while varying variable like bandwidth and window size. We observed the inconsistencies in the results caused by the use of two different approaches to compute history (mean path delays). We want to bring both variations in par with each othe,r however, computing max-flow in opportunistic networks is a time intensive process. Therefore, we have not been able to obtain timely results from a uniform way of history computation.

Nevertheless, we can conclude from these experiments that the variance of all variables may produce counterintuitive results depending on the underlying network. History can bring good quality prediction provided the network devices show regularly consistent behaviors. The validity of history decays with the passage of time and particularly sparse network suffer this decay in quality. It is also essential to keep the history size in check, although, history gathered over longer period does bring some advantages. In our opinion the history window size must be correlated with the mobility of individual devices, i.e. devices with low mobility should have longer history window than devices with high mobility. Moreover, these methods must be further developed by using advanced statistical methods to improve the prediction about path reliability.

Chapter 10

Multicasting and social networks

This chapter introduces an architecture (*Mergenet*) for mobile social networking, which is inspired by the similar motivation that resulted in *Nile*. *Mergenet* not only uses *Bloom filters* to store the profiles of users belonging to a social network but also defines a method that helps users to exchange ideas/information/knowledge without loosing their privacy. In the end, we present the prototype design, and its functionality in the selected traces.

10.1 Introduction

Nowadays, we face a novel kind of social networking, *virtual social networks* as enforced by well known platforms like *Facebook*, *MySpace* or *Friendster*. New social networks emerge each day, e.g., at a formal conference, at informal social gatherings, at family reunions, etc. Internet has already been playing an important role to let people socialize through online social websites. However, for many users this is still not an optimal way of interaction, since staying in contact with peers requires many steps, e.g., establishing contacts, completing user questionnaires, etc.

Technology simplifies to stay in contact with others, even in remote locations of our planet, e.g. by email or chat. However, the most prominent device used to stay in contact with others – and probably nowadays the most *personal* one – is the mobile phone. With the easy access to mobile devices, modern technologies have now started to adapt to more active socializing. As these mobile devices accompany their users almost at all the times, they can record and observe their

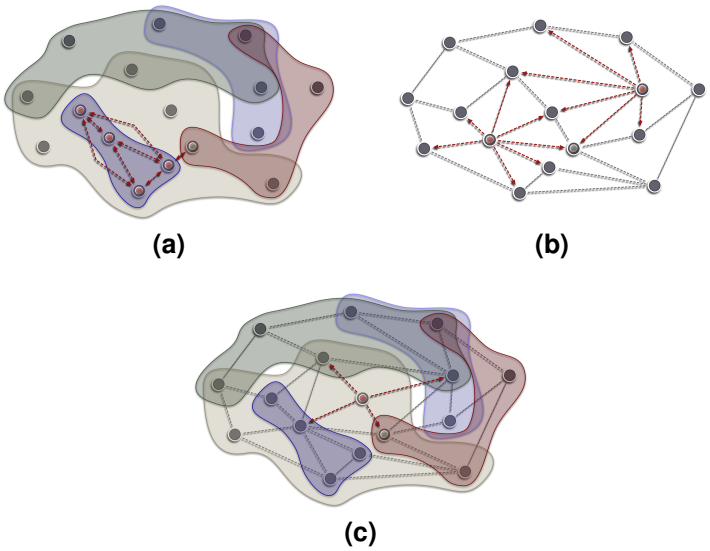


Figure 10.1: Simplified illustration of the concept of semantic group models fig.(a), centrality measurements fig.(b) and the enhancement of connections in a social network by exploiting social information about the nodes fig.(c).

users behavior as well as gather information about their social circle. Therefore, they can help users to receive information from contacts they may potentially not even know.

Modern social networking also reflects another aspect of today's life. Information is no longer only stored and found in databases or documents in the World Wide Web but also information, or to be more precise, *knowledge and sentiments* are shared by users worldwide. Hence, social networks become even more prominent when searching for information to solve a problem. Simplified, it all comes down to the well known adage: *"It's not what you know, it's who you know"*. Therefore, the question we face is less *how we find the information,* but more how we get in contact with *someone who has the information or expertise*. Such *group-forming networks*, as discussed in Reed's law [Ree99], have a utility which grows exponentially with the number of participants, unlike traditional networks whose utility merely grows linearly or quadratically. The traditional sense of "friend" is somewhat altered in the context of this new type of social network. These social networks alter the type of relationship between people, as personality becomes somewhat irrelevant, and for the most part the mere interest for a specific topic connects one person to another. Furthermore, modern social networking bypasses the geographic context in which people reside.

10.2 MANET in the context of social networks

Following, we give a brief overview of related work in the area of *Mobile Ad Hoc Networks* (MANETs) and *Delay Tolerant Networks* (DTNs) related to exploiting social information in these networks.

10.2.1 Exploiting social and semantic information

Researchers in social sciences have been debating several parameters like number of contacts, closeness to contacts, location in the network, etc., help classify the status of someone in his social environments. However, the concept of *Betweenness centrality* [GN02] has received the most of the attention (Fig10.1(b)). Using *Betweenness centrality* those nodes in the network are addressed for routing that have the highest number of connections to other nodes in the network. Daly and Haahr [DH07] propose a metric to identify characteristic nodes in a delay tolerant MANET that can serve as bridge nodes to pass messages to the intended receiver. To identify these nodes, they exploit social analysis techniques as a measurement of centrality in a MANET to enhance routing. Their idea was derived from the characteristics of the *small world phenomenon*, which states that individuals are often linked by a short acquaintances [Mil67]. Miklas, et al. [MGC+07] exploited the use of social information in mobile systems, conducting a simulation on the "Reality Mining" dataset of the MIT Media Lab [EP05]. In their simulation they classified the users into "strangers" and "friends" due to their number of encounters during the experiment. Their simulation has shown that this simple classification already yields a gain in the message delivery performance in DTNs in terms of the time a message takes to arrive at its destination. Zhao, et al. [ZAZ05] introduce semantic models for efficient routing in DTNs by defining group membership models (Temporal Membership, Temporal Delivery and Current Membership) for routing messages through the network. These models define the membership group, that may be interested in receiving a message from the network depending on the location and/or time. A similar semantic model was proposed by Gong et al. [GXZ+06].

In contrast to the approaches of [ZAZ05,GXZ+06], we follow the idea of [MGC+07] to utilize social information.However, we not only classify users into "friends" and "strangers" but add "virtual friends" a third class to the system. We also introduce transitivity in the social information of a user, simplified, we combine *Betweeness Centrality* and *Group Membership Models* to identify users that are either experts in a specific topic, or users that are connected to experts, respectively (c.f. 10.1(c)). We define this classification in more detail in section 10.4.

10.2.2 Mobile social networking

Mobile phones, or mobile devices in general, can help us to bring social networking to the next step, to mobile social networking. Therefore, we first define why it is important to rethink what we like to achieve with modern social networks, second, what difficulties we face when we like to integrate these aspects in the next type of social networks.

1. Connecting to experts that have some specific knowledge will facilitate problem solving as these experts can provide quick access to answers that are difficult to be solved by an individual.
2. Current online social networks work on a *pull* type approach, where users explore the available connections in the network. However, we argue in favor of a *push* type approach, as we know it from *real-world social networks* when we are introduced to others by colleagues of friends. This *push* approach can also be seen as some kind of recommendation to quickly connect to the experts.
3. In this context, *transitivity* is a very important aspect that is specifically inherent to virtual social networks. Transitivity is a powerful way to explore social connections by following links to n^{th} degree contacts of our direct contacts. However, due to the *pull* approach, utilizing transitivity is somewhat limited as the user is responsible to look for experts and does not get any references.

10.2.3 Issues involved

We propose a system based on mobile devices, thus, building an implicit infrastructure for mobile social networking. The proposed system integrates social information of its users, stored in Bloom filters [Blo70], to facilitate the look-up of other users that could be of potential interest. We define a two-way message protocol that exploits the social network's inherent transitive connections, enhanced with the social information of the users, to route messages to a suitable receiver.

We see following three aspects that have a pivotal role in a mobile social networking system.

1. Modeling an automatic recommendation system based on the *push* approach.

2. Modeling the transitive relationship in a system that facilitates the desired *push* approach.

3. Including social information (e.g. expertise in a specific topic) of the users in the social network.

Further, this message protocol should enable a receiver to reply to the initial sender using a pointed multicasting approach, while maintaining the responder's privacy as well as the privacy of the initial sender.

10.3 Bloom filters in networking

Bloom filters [Blo70] are simple space-efficient randomized data structures that represent a set of elements and that support membership queries while allowing false positives. A Bloom filter is a bit array of length m, with k hash functions, that can hold up to n, $n < m$ elements. To add an element into the Bloom filter, the element is hashed to k different indices of the Bloom filters bit array by applying the hash functions. Each of the k bits designated by the hash functions is then set to 1. If a bit would be set by different elements the bit stays set. The probability p_{err} of a false positive depends on the three parameters m, n, k of the Bloom filter and is calculated as::

$$p_{err} = (1 - e^{-kn/m})^k \tag{10.1}$$

Given a fixed size m for a Bloom filter as well as the intended (maximal) number of elements n, the optimal number of hash functions to uses can be obtained as[1]::

$$k = \ln 2 \cdot (m/n) \tag{10.2}$$

Bloom filters have been used to solve a variety of networking problems and different variations of Bloom filters have been proposed in the last years, and most of them are suited for a specific application area. Mitzenmacher [Mit01] introduced the concept of compressed Bloom filters and has shown that, by some modifications to the original concept, the size of a Bloom filter can be reduced without increasing p_{err}. Guo, et al. [GWCL06] proposed dynamic Bloom filters that can grow as needed by adding static sized Bloom filters to a list of Bloom filters as soon as the actual Bloom filter's false positive rate increases over a given threshold. Bauer, et

[1]Note that in practice k is rounded down to the next integer since generally this does not increase p_{err} dramatically and yields better performance.

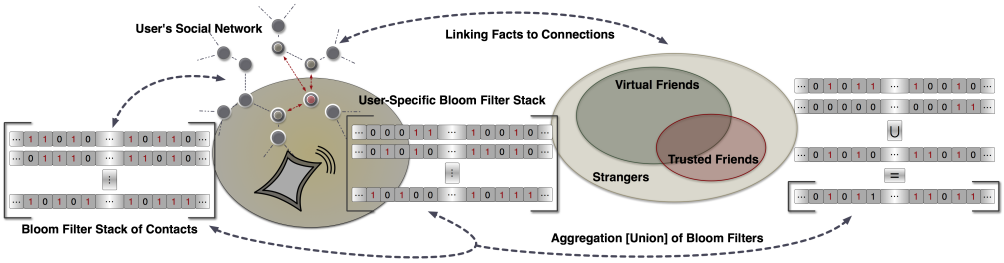


Figure 10.2: Illustration of the user profile as stored on the mobile device. The user's social contacts are classified according to the Venn diagram on the right.

al. [BHPW04] propose simple boolean set operations on Bloom filters to combine sub-queries for efficient queries on distributed hash tables. However, Bloom filters are by far not the only way to go. Hurley and Waldvogel [HW07] have shown that in the case that false negatives are admissible, Bloom filters can be outperformed by other techniques.

We use Bloom filters to encode the social information of a user as they allow efficient set operations as shown in [BHPW04]. As already briefly stated, we enhance our system with this user specific information and further utilize this information in the look-up procedure of potentially interesting contacts as well as in our two-way message protocol. Therefore, we can tolerate false positives, whereas false negatives are not preferable.

10.4 Mergenet architecture

We assume that users of our system are in possession of mobile devices capable of P2P communication. We aim *Mergenet* to be flexible enough to let a user establish a P2P connection through arbitrary techniques like WAN, LAN, Wifi, or Bluetooth. A user's profile is present on the mobile device, giving the responsibility to the user to keep his profile updated, especially if the user uses multiple devices. Details about the profile maintenance will be presented in section 10.4.1.

Mergenet takes into account the disparity among the users with respect to their social activities. Some users may like to be very social by introducing themselves to as many people as possible. In contrast, some may want to keep a very limited activity by maintaining a small number of close contacts. *Mergenet* classifies contacts as *users* and *virtual friends* similar to the idea of [MGC+07] depending on the frequency two devices see each other. However, in contrast to [MGC+07],

we let users add *trusted friends* manually, or, depending on recommendations of the system, let a user choose to add a virtual friend in his list of trusted contacts. For a user p the set of virtual friends C_{VF} is defined according to a threshold α_p as::

$$C_{VF} = \{c \mid c \in C_U,\ T_c \geq \alpha_p\}, \tag{10.3}$$

where C_U is the set of all the users p encountered, $T_c = M_c/(\sum_{i \in C_U} M_i)$, M_c is the meeting count of p with $c \in C_U$, and M_i is the meeting count of p with $i \in C_U$. The threshold α_p varies for all users and depends on the embedding of p in the social network. At the moment we are running experiments to identify the best matching α_p for a user p.

Following, we present two definitions we will use throughout the next sections:

- **Client:** a user interested in either introducing himself to the community or to enquire some specific information.
- **Responder:** a user interested in getting to know a new *client* or to respond to a query from a client.

10.4.1 Profile structure

$B_j,\ j \in \mathbb{N}_1$ in a stack $BS_i = \{B_1^i, B_2^i, \ldots, B_j^i\},\ i \in \mathbb{N}_1$. BS_1 denotes the Bloom filter stack for the current user while $i > 1$ denote the Bloom filter stacks for the users that came in direct contact with the current user. The first Bloom filter B_1^i in each stack is considered mandatory and contains the minimum information for the profile to exist. B_1^i will be populated with at least the identification information of the users, while the remaining Bloom filters correspond to a user's interests (e.g. music, movies, literature, etc.). For every Bloom filter in the stack we compute the check sum. Whenever the profile is updated, the changes will be hashed in the corresponding Bloom filter and the check sum is recomputed. Thus, other users in the network perceive the change of a contacts profile. Formally, let $CS^{B_j^i} = f^B(B_j^i)$ be the check sum for the Bloom filter B_j^i. Then, the check sum of BS_i would be computed as $CS^{BS_i} = f^{BS_i}(CS^{B_k^i}),\ k = 1, \ldots j$. This helps to shortlist the number of Bloom filters in the case of profile exchange.

Whenever two users come in contact with each other for the first time they create a new Bloom filter stack for each other, incrementing their corresponding values of i. At the time of profile exchange, *Mergenet* computes the union of all Bloom filters at one level of the stack before transmitting them. Formally, let $BS^T = \{B_2^T, \ldots, B_j^T\}$

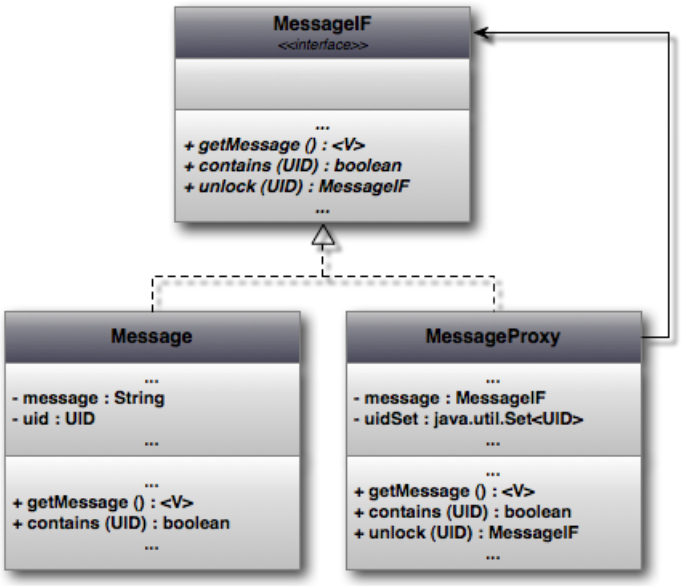


Figure 10.3: The UML class structure of the message design.

be the Bloom filter stack to transmit. BS^T is computed as follows, omitting the mandatory Bloom filter::

$$BS^T = \forall BS_i : \bigcup_j B_j,\ j > 1. \tag{10.4}$$

We can thereby ensure the transitivity property in the network. Meaning that, if user A shares her profile with B, and B shares her profile with C, C's profile also contains information about A without being connected to A directly. However, *Mergenet* permits the aggregation of profiles only for Bloom filter stacks $BS_i,\ i > 1$ of *virtual friends*. Bloom filter stacks associated with users $u \notin C_{VF}$ will not be aggregated into BS^T. These Bloom filter stacks will be kept separate, as they relate to users of the system that are not encountered frequently. Fig. 10.2 illustrates the relationship between an user's profile encoded in a Bloom filter stack, the Bloom filter stacks of trusted friends, and the aggregation of an user's profile and the profiles of her *virtual friends*. Note that the set of *trusted friends* is not necessarily a subset of *virtual friends*.

Due to the aggregation of the different Bloom filters in the Bloom filter stacks, the false positive rate p_{err} (c.f. Eq. (10.1)) can increase dramatically. In our design we focus on three different approaches to overcome this problem.

- Use large Bloom filters with a fixed size $m = 2^{31}$ that can be compressed before transmission as proposed by [Mit01].
- Use *Dynamic Bloom Filters* (DBF) [GWCL06]. Thereby, the size of any Bloom filter $B_j \in BS_i$ will be kept flexible and grows dynamically as needed. Whenever the existing Bloom filter gets heavy to sustain the false positive rate, another Bloom filter of equal size is created while the old one is sent to storage and treated as legacy Bloom filter. This enables our system to transmit only chunks of the profile that were updated instead of always transmitting a large Bloom filter. Furthermore, clients with low populated profiles will not be burdened by unfilled Bloom filters.
- Use *Attenuated Bloom Filters* [RK02] that only aggregate the information of user profiles up to a predefined depth of transitive links. Actually, this is the most promising approach so far.

10.4.2 Protocol definition

The transmission protocol consists of two kinds of exchange – *profile exchange* and *information exchange*.

Profile exchange

Mergenet identifies every user by an unique identifier (UID). Whenever two users encounter each other one user assumes the role of the *Client*, the other as the *Responder*. If one device acquires the role of the *Client* c, it connects to the *Responder* r. Following, the profile exchange is achieved as shown in Alg. 8. Method calls on the *Client/Responder* side are denoted as $[c|r].methodCall()$, respectively. This syntax will be used throughout the presented algorithms.

Information exchange

Information exchange includes any kind of queries users may ask among each other. *Mergenet* retrieves the suitable *Responder(s)* to a query by directing the query to only those users who have knowledge in the area the query belongs to. To route the message to a suitable *Responder*, *Mergenet* includes the topic of the query in the message as a Bloom filter that contains the relevant bits to identify the topic. Note that every query has a lifetime limited by t_{exp}, its *expiry time*.

Mergenet assumes that c maintains a list of all alive queries $Q = \{q_0, \ldots, q_k\}$ sorted by (1) the number of replies and (2) their expiry time. A query q is an object of type

1: $c.send(UID, CS^{BS_c^T})$ {//send initial information to r}
2: **if** $r.isNew(UID)$ **then**
3: $r.add(UID)$
4: $BS_c^T \leftarrow c.exchangeBS(BS_r^T)$ {//exchange stacks}
5: $r.addBFStack(UID, CS^{BS_c^T}, BS_c^T)$
6: **else if** $r.isUpdated(UID, CS^{BS_c^T})$ **then**
7: $BS_c^T \leftarrow c.updateBS()$ {//update stack of c}
8: $r.replaceBFStack(UID, CS^{BS_c^T}, BS_c^T)$
9: **end if**

Algorithm 8: Profile Exchange & Update

`MessageIF` (Fig. 10.4.2 & Fig.10.3). As a precondition for information exchange, *Mergenet* assumes that user profiles are up-to-date according to Alg. 8.

Message Design In *Mergenet* we utilise a proxy enveloping technique. Using this technique, the *Responder* will only "see" the set of forwarding users of the query (including the initial *Client*). The *Client* will only see the answer to his query but not the person who gave the answer. This technique preserves the privacy of the *Client* as well as the privacy of the *Responder*. We design this enveloping technique by utilising the *Decorator* design pattern [GHJVng] as depicted in Fig.10.3. The `MessageIF` exposes four methods to retrieve the topic of the message encoded in a Bloom filter, the message text, a method to check if a given UID is included in the set of forwarding nodes of the message, and a method to unlock the method up to an envelope created by the node with the corresponding UID. The initial *Client* creates the `Message` object adding the topic of the message, the message itself, and its UID. Thereafter, the message is wrapped in a `MessageProxy` object, adding an arbitrary number of optional message receivers from her from her list of *virtual friends*. These additional nodes are later used for passing back the response to a query.

Query & Response Forwarding Forwarding a query q to a suitable *Responder*, or routing the answer to a query back to its initial *Client*, involves the steps described in Alg.9. According to the previous definition, we write $q.messageCall()$ to represent a method call of the message object. Simplified, the current *Client* – either the initial creator of the query or an intermediate node – passes the message to the next *Responder* that is identified as an user that can either answer the query, or knows another person that can answer the query, respectively. Whenever c connects to a *Responder* r it checks if r is either an expert for the topic of any message $q \in Q$, or can serve as a forwarding node for the message. Therefore, c

```
1: for all q ∈ c.Q do
2:   if q.isQuery() then
3:     if r.isExpert(q.getTopic()) then
4:       r.handleQuery(q) {// c.f. Alg. 10}
5:     end if
6:   else
7:     if q.contains(r.UID) then
8:       temp ← r
9:       r ← c
10:      c ← temp
11:      c.handleQuery(q.unlock()) {// c.f. Alg. 10}
12:    end if
13:  end if
14: end for
```

Algorithm 9: Query & Response Forwarding

and r first exchange or update their profiles according to Alg.8. As r exposes his aggregated profile BS_r, the profile information of his n^{th} degree contacts will be included as well. Thus, c does not actually know if r can answer the query or if r simply serves as an intermediate node.

This forwarding and back-routing scheme is further illustrated in Fig.10.4. The *thick (red)* edges correspond to a query-forwarding from the initial *Client* on the left to the final *Responder* on the right. Thereby, the query is forwarded by the intermediate users, represented by the (red) nodes in the centerline of the figure. Each of the intermediate nodes encapsulate the message in another `MessageProxy` object (the message envelopes), inserting an arbitrary number of additional contacts that are identified as *virtual friends* of the active node (c.f. Eq. (10.3)). These additional nodes serve as proxies that can be used later for routing back the answer of the query to the initiating *Client*. The grey areas in the back of the figure correspond to the envelopes an intermediate node creates before forwarding the message. Every message envelope contains the information of the creating node, including the additional proxy nodes, and each envelope, including all envelopes created by nodes after the actual node, can be unlocked by the creating node.

During response forwarding the final *Responder* keeps the message as long as the query is alive and passes it to every node, including proxy nodes (beige), it encounters. As every forwarding node includes a number of proxy nodes, a forwarding node defines a "corridor" for the returning message. Thus, the network is not simply flooded with messages containing the answer to a query. Instead,

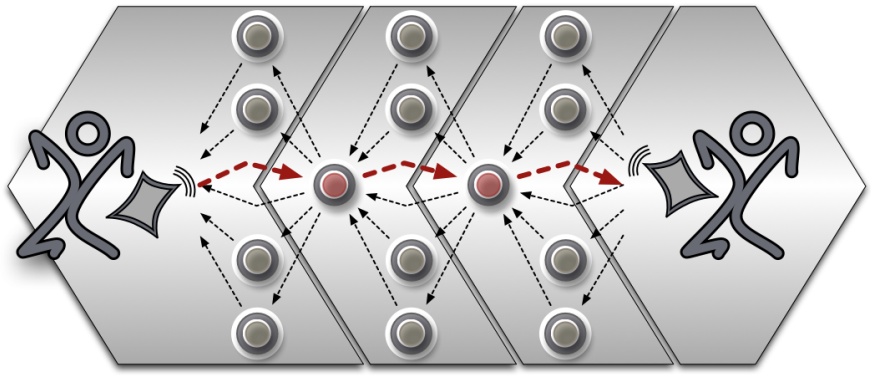


Figure 10.4: Illustration of the message protocol.

the route to the final *Responder* already defines a good route back to the initial *Client*.

```
1: if c.isExpert(q.getTopic()) then
2:     q.setAnswer()
3:     q.setIsQuery(false)
4: end if
5: c.Q.append(q)
```

Algorithm 10: Query & Response Handling

10.5 Results summary

We performed several simulations using this multicasting protocol with the point of view spreading a query in social networks. We have filled the profiles of the user with random interests and hobbies. Each query is dedicated to a particular kind of experts that have advertised their profiles in such a way that shows their interest in the subject of the query. This implies that the aim of the protocol is to reach as many as possible experts in a short span of time. In the context of *Mergenet* we can reduce the number of transmissions made, by limiting (a)the number of hops a query may traverse and the ratio of the (b)direct contacts a middle hop may forward the query to. These factors will not hamper the functionality of *Mergenet* because the aim here is to contact at least one expert that can answer the query of the client.

Therefore, we have introduced two parameters, i.e., depth and spread of the query. *Depth* corresponds to the hop count that a query is allowed to traverse while *spread* corresponds to the ratio of direct contacts, each hop may include in the recipient

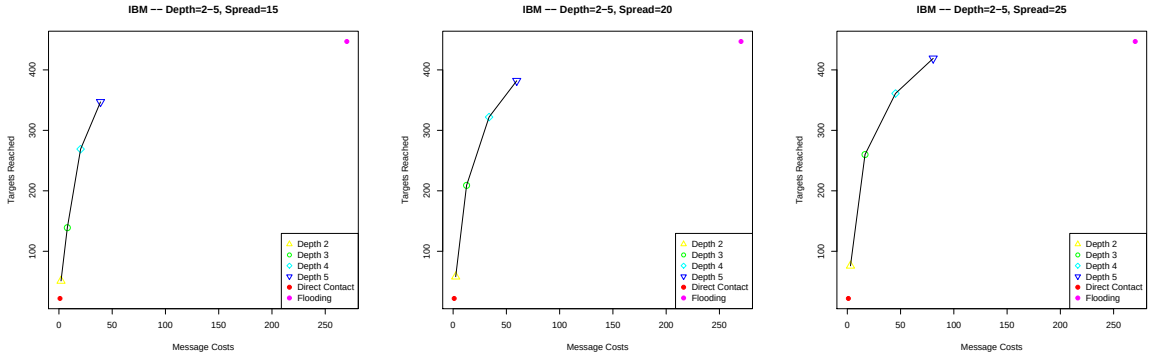


Figure 10.5: Target vs. costs in the IBM trace with (a)spread=15%, (b)spread=20%, (c)spread=25%

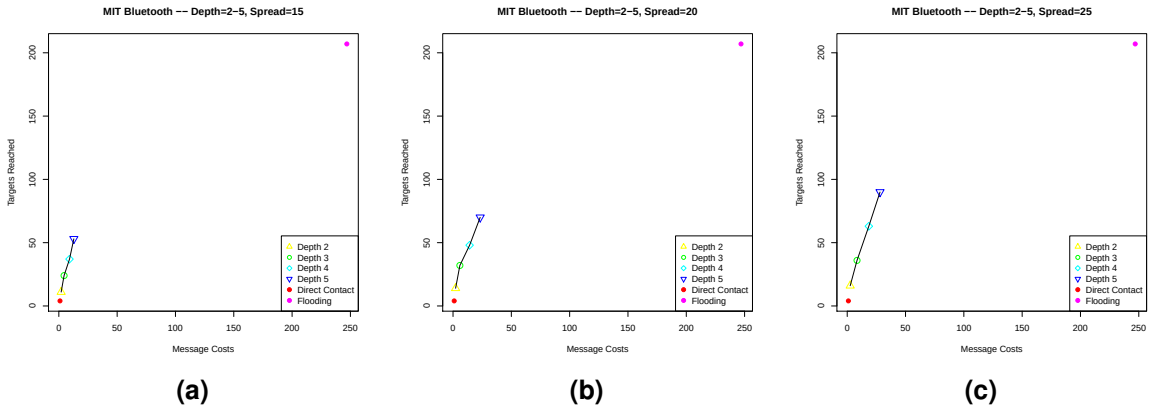


Figure 10.6: Target vs. costs in the MITBT trace with (a)spread=15%, (b)spread=20%, (c)spread=25%

list. The results presented here are presented with 4 values of *depth* (2,3,4,5) and 3 values of *spread* (15%,20%,25%). We have included two performance metrics (a)*cost* that represents the number of transmissions made by each variant whereas (b)*target* represent the number of experts that were accessible. The *depth* factor is only valid for our strategy and is represented by $depth = x$ in the legend while the spread factor is affixed with the corresponding title of the plot. The remainder of the implementation details and results obtained in other variations are available in [Zin10].

We have compared the results of *Mergenet* with *direct delivery* and *flooding*, As

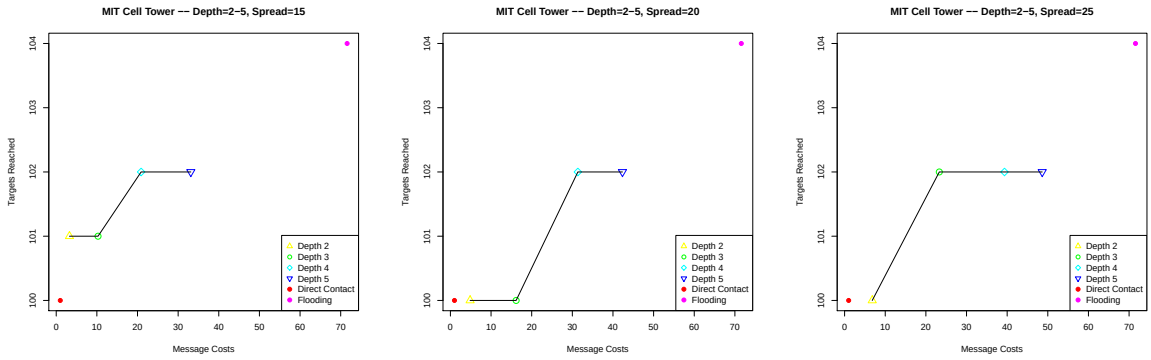


Figure 10.7: Target Vs Costs in the MIT trace with (a)Spread=15%, (b)Spread=20%, (c)Spread=25%

direct delivery and *flooding* are not dependent on either *depth* or *spread*, these varying parameters are not having any effect on their cost and targets accessed as shown Fig. 10.5, 10.7, and 10.6. In the case of access point (IBM) trace Fig. 10.5, we observe a gradual increase in the number of target that were contacted as the *depth* and *spread* factor rise. Although, there is a corresponding increase in the cost as well but it is far less than cost of *flooding*. In contrast, the number of targets contacted is almost as good as *flooding*. In the sparse case of the bluetooth (MITBT) trace, we see that *Mergenet* has failed to match the performance of *flooding*. We face a big challenge due to the sparse nature of this network. Moreover we believe that both the *spread* and *depth* factors must be increased to match the performance of *flooding*, while there will be a corresponding increase in the costs as well.

The dense cell tower (MIT) trace shows that we can easily tolerate a low *depth* as well as a low *spread* factor while keeping a *flooding* like performance. The $spread = 30\%$ factor has overridden the effects of *depth* due to the density of the network. In contrast, *direct delivery* has also contacted an impressive number of targets. As results related to delays incurred by these variants are not included, it is hard to see the advantage of *Mergenet* in the case of the MIT.

We are currently revisiting our implementation of the strategy in such a way that dispersion of the queries to find the experts is independent of any parameters. Furthermore, we intend to rely only on the shared Bloom filters to configure the protocol so that we can assure the query delivery while keeping the query fan-out as minimum as possible. We are investigating the effects on performance when a node

increases the *depth* factor if node-degree is very small. This will help the query to traverse farther in the network. Moreover, when node-degree is not small but the shared Bloom filters of confirmed friends are not showing access to experts/target, then a node can flood the virtual friends to obtain the desired results. We are also working on the results of our response routing strategy as depicted in Fig. 10.4. Thus, we believe that *Mergenet* has still a few steps to go to earn the status of efficient mobile social platform.

Chapter 11

Open issues and conclusion

In this chapter we have a bird's eye look at the work that we have presented earlier and identify the aspects that we have intentionally or unintentionally avoided or ignored. Based on our experience in this field, we intuitively argue on the effects of these aspects that we have not included in our analysis.

11.1 Future directions

The presence of mobile devices in a network opens several issues that one does not have to ponder about in wired networks. We have presented a broad overview of these aspects but there are still several issues that must be considered for routing in such scenarios. The issues such as power consumption or energy efficiency may not belong to a routing layer of the protocol but a cross layer optimization can help reduce their side effects. In this section, we present a brief summary of such issues that we would be concentrating upon in the future.

11.1.1 Message size

Based on the results in section 7.3, we believe that the messages should be propagated in a rather streaming behavior rather than chunk-wise. We define streaming as sending the messages in small pieces and partition them whenever a transmission is broken. The carrier of a partial message can forward the message in a same manner and in a result, the message may get divided into more unequal fragments.

It has been our intention from the beginning not to couple the size of the mes-

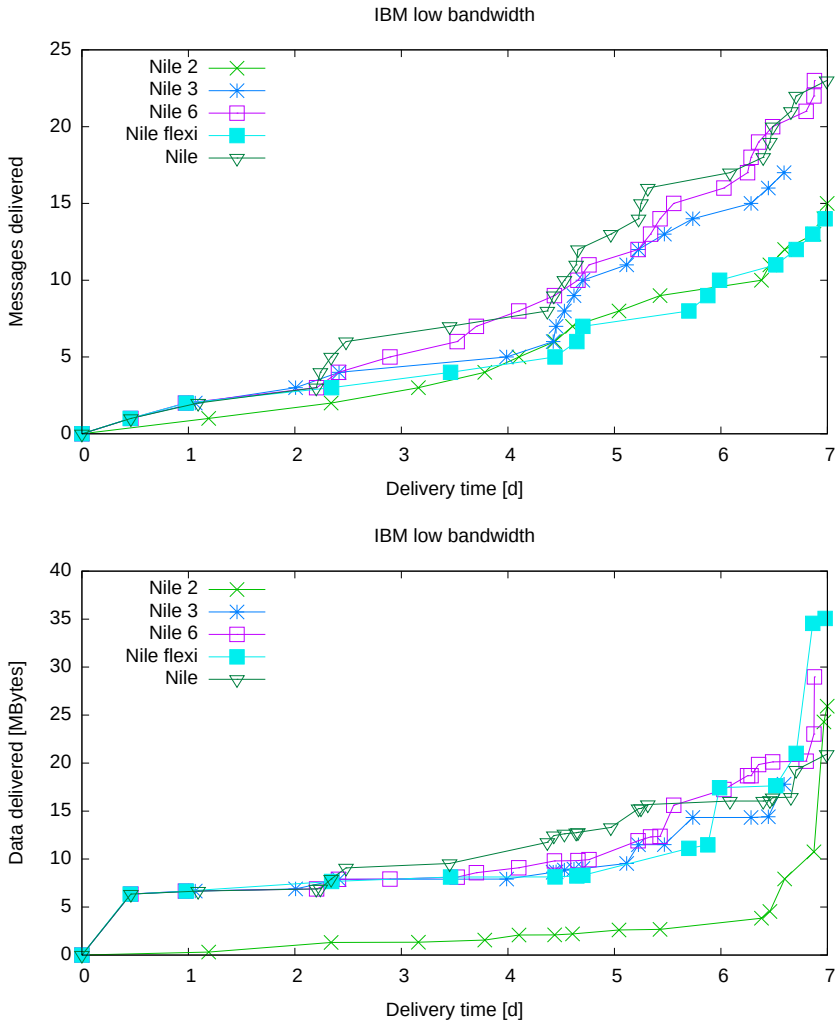


Figure 11.1: Plot for Nile using variable fragment count

sage to the path parameters and therefore we have not followed in this direction. Nonetheless, to confirm our hunch, we played around with different fragment sizes for *Nile* simulations. As shown in Fig. 11.1, we can see that as the number of fragments increases a couple of bigger messages that had otherwise failed to do so, have reached their destinations.

Following this intuition, we have experimented such that the number of fragments, depending on the path characteristics, are dynamically allocated to each of the messages. The message source makes this decision with he help of history and the message fragments are then created in such a way that the message size corresponds

to the capacity of the path that can deliver the least volume. This technique represented by *Nile flexi* in Fig. 11.1, has delivered a couple of big messages that rest of *Nile* variants have not delivered. In contrast, *Nile flexi* has sacrificed the number of messages as bigger messages have consumed the necessary bandwidth that was vital for these smaller messages. It is evident from these results that performance can be further improved, given more precise fragmentation and replica generation decision are made.

11.1.2 Path aggregation

As we argue in favor for a multi path routing strategy, we face the challenge of path aggregation. We will explain the problem with the help of Fig 11.2 where a source sees multiple paths to the destination. The first path is advertised by the device E while the other path is advertised by device D. As the source cannot see beyond its neighbors, it has no idea about the availability of multiple paths via neighbor E. The question is how E can advertise its metrics to the source that it knows E has the flexibility of path choice.

Lets assume that the path D,T is the fastest among all the available paths via E, then the source will prefer to push the message to D. If the path D,T is less congested than all the available paths via E, but their collective available bandwidth exceeds that of path D,T, it is not easily possible to aggregate the metric in such a way that it shows the commutative effect of these paths. We can, off course enhance the routing table with this extra information but this aspect will cause the size of routing table to explode exponentially and this is not feasible for mobile devices where every device acts as a router.

To make the problem clearer, we will revisit the strategy that we used in *Nile* to fill the routing table in section 6.2.1.

Assuming the scenario in Fig. 11.2, every device A,B,C,T is being advertised by the device E will be inserted into the routing table of the device X. If E has the minimum delay ϵ path to T via A then this path will be advertised to X. If the delay of path X,D,T is also ϵ, X cannot see the difference between the two paths. The path via E may have potentially more available capacity than the path via D, therefore it is a good idea to prefer the path via E over the other.

In our opinion the device E should offer an aggregate measure of all of its paths to T in such a way that the device X prefers to push messages destined to T via E. One option could be to discard all those paths that are not within one standard

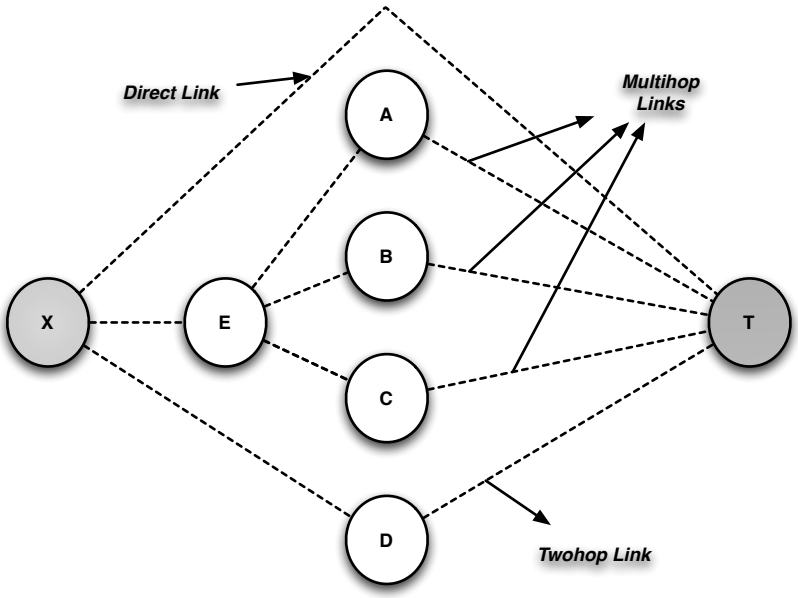


Figure 11.2: Fork aggregation in paths

deviation σ from the mean of all available paths and $\epsilon - \sigma$ should be advertised as the aggregated minimum delay of all the paths. The modified tuple sharing scheme could be as follows.

a. Node ID D if X has no earlier record of D via X.

b. Expected delivery time from X to T,

$$\epsilon_{XT} = \epsilon_{XE} + \delta_{ET}$$

where

$$\delta_{ET} = mean(\epsilon'_{ET}) - \sigma(\epsilon'_{ET})$$

and ϵ'_{ET} is the mean delay of all paths from E to T.

c. Congestion Indicator on the path X, E,... T

$$\eta'_{ET} = (b'_{ET} + \sigma(b'_{ET}))/(b'^{v}_{ET} - \sigma(b'^{v}_{ET}))$$

where b'_{ET} and b'^{v}_{ET} are the average number bytes delivered and average number of bytes that could be delivered, while

$$\eta_{XT} = \begin{cases} max(\eta_{XE}, \eta'_{ET}) & \text{if } \eta_{XE} < 1,\ \eta'_{ET} < 1 \\ \eta_{XE} \times \eta'_{ET} & otherwise \end{cases}$$

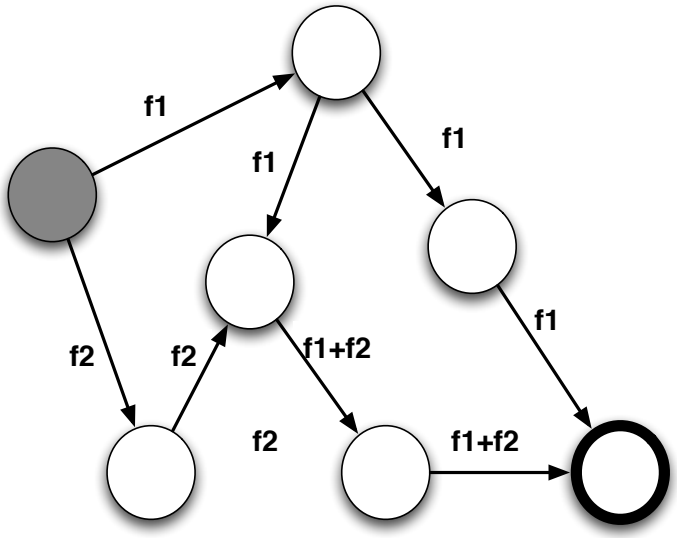


Figure 11.3: Network coding to increase capacity

11.1.3 Network coding

In our opinion, routing in opportunistic networks can improve further, if we propagate the messages with the help of network coding. Network coding is a generalization of the conventional routing (store-and-forwarding) method. In conventional routing, each intermediate node in the network simply stores and forwards the messages received from other nodes [Dim07]. In contrast, network coding allows the intermediate nodes to generate output data by encoding (i.e. computing certain functions of) previously received messages. In principle, during the propagation of messages from source to destination, network coding allows messages to be "mixed" (encoded) at intermediate nodes. An appropriate coding over the links in the network may render those links in effect error-free. Network coding can then be used to achieve capacity or recovery over an error-free network, with possible delays due to coding.

The issue of network capacity has generally been considered in the context of network links exhibiting ergodic error processes [KMM03]. Networks that do not experience ergodic error processes may be reasonable models for networks that in reality are built from links exhibiting ergodic failure processes. The potential advantages of network coding over routing include resource (e.g., bandwidth and power) efficiency, computational efficiency, and robustness to network dynamics.

This idea is illustrated in Fig. 11.3 where any of the middle hop in the network can

combine different fragments of one message that have been floated by source in the network. The coded fragments give the flexibility to the destination to decode the original message if any one of the original message gets through the network. We pursued this direction but we could not solve the challenge of selecting optimal fragments to encode. For, e.g., a source has n fragments, it can encode these n fragments in $2^n - n - 1$ different combinations. Theoretically, middle hops are also capable of encoding the fragments (including already encoded) that make the problem of optimal fragment selection a big challenge.

11.1.4 Modeling

Modeling opportunistic network has posed the biggest challenge for us during the course of this project. Opportunistic networks being a relatively new research area, still going through evolution process, makes it a challenge to adapt a mathematical model for it. The introduction of QoS factors make this challenge double fold as mathematicians are still in the process of improving their QoS based models for wired networks.

We have attempted to adapt a few existing models and we see the opportunistic network modeling problem has dual fronts.

- **Network view** can be interesting for the designers and the architects of the networks who would be interested to optimize the performance of networks by defining bandwidth allocation schemes.
- **Device view** can be used at the device level to make routing decisions where a device can forecast the delivery probabilities.

Network view As formulated by Wang and Luh [WL09], the challenge in integrated-services networks is to determine a general resource allocation and QoS routing schemes that have the following desirable goals.

1. Links and bandwidth must be determined in a way such that the network can assure fairness and maximum revenue.
2. Bandwidth must be allocated for each flow such that user requirements are met and at the same time such that the predefined condition of fairness is satisfied.

3. An appropriate effective route for each flow (between two nodes) must be found.

We find these goals very much coherent with the routing goals in opportunistic networks but as already mentioned in section 1.1, we lack the luxury of obtaining real time network metrics from opportunistic networks. Therefore such goals have to be re-defined to make them suitable for opportunistic network scenario. Apostolopoulos et al. [AGK+] concentrate on improving this trade-off by devising methods that increase the performance of QoS routing, when cost constraints such as, limitations on link state update traffic, affect the accuracy of the information on which it operates. This can also be an interesting avenue to explore, although, link-state protocols can expose memory limitations of mobile devices.

Device view Le and Hossain have used *tandem* [LH08] queues to model QoS routing in multi-hop wireless networks. They argue that the routing component of a QoS-routing algorithm essentially involves link and path metric calculation, which depends on several factors, such as the physical and the link layer designs of the underlying wireless network, transmission errors due to channel fading and interference. Using such a queueing framework, we can derive different performance measures, namely, end-to-end loss rate, end-to-end average delay, and end-to-end delay distribution. Although, we have ignored *Link layer* related issues in this project, we believe it is imperative to consider them to calculate the link metric and incorporate this into the route discovery process of any QoS routing algorithm.

We see the modeling problem in the context of *Nile* as a fork join parallel queue represented by Fig. 11.4. Assuming that we have obtained disjoint path, a source may see the network cloud to destination as set of parallel queues that *split* up at the source and *join* at the destination. The queues on the paths represent the message queue on each middle hop and the delays encountered in the queue is dependent on the waiting time and durations of the next hop. The message then jumps from one queue to next queue and in the end reaches the sink, i.e., destination. As the first queue will report the delays of all the queues that are in sequence also known as *tandem* queues [LH08], until destination, a source can estimate the probability of delivery.

11.1.5 Cooperative communication

A wireless network system is traditionally viewed as a set of nodes trying to communicate with each other. However, from another point of view, because of the broadcast nature of wireless channels, one may think of those nodes as a set of

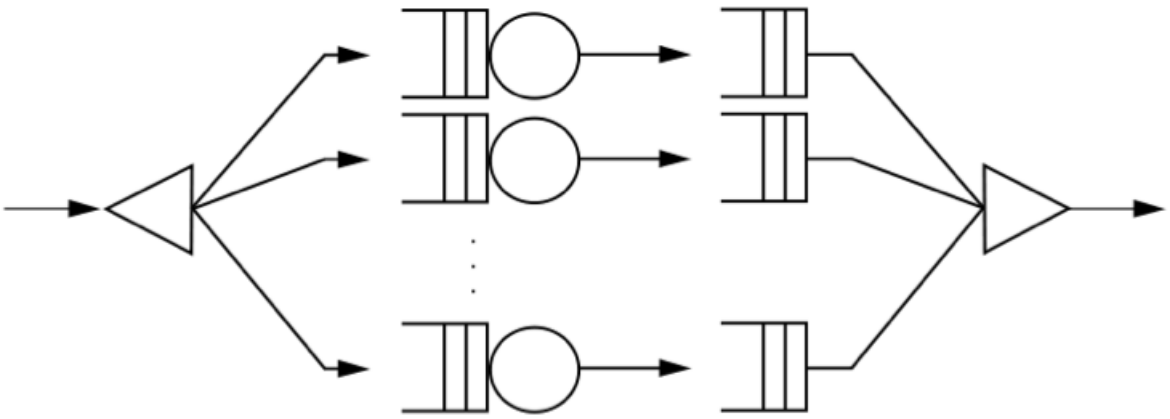


Figure 11.4: Fork/Join parallel queues [DG10]

antennas distributed in the wireless system. Adopting this point of view, nodes in the network may cooperate for distributed transmission and processing of information. Cooperative communications can generate independent multi-input, multi-output (MIMO) like channel links between a source and a destination via the introduction of relay channels where a cooperating node can act as a relay node for a source node.

In recent years, a new perception for ad hoc and sensor networks known as *cooperative communication* has emerged [KJRLK09] where a source has to rely on the assistance from other nodes to forward or relay information to a desired destination. Such a need of cooperation among nodes or users has inspired new thinking and ideas for the design of communications and networking systems by asking whether cooperation can be used to improve system performance. As a result, a new communication paradigm arose, which had an impact far beyond its original applications to ad hoc and sensor networks.

It is a new paradigm that draws from the ideas of using the broadcast nature of the wireless channels to make communicating nodes help each other, of implementing the communication process in a distribution fashion, and of gaining the same advantages as those found in MIMO systems. Such a new viewpoint has brought various new communication techniques that improve communication capacity, speed, and performance; reduce battery consumption and extend network lifetime; increase the throughput and stability region for multiple access schemes;

expand the transmission coverage area; and provide cooperation tradeoff beyond source–channel coding for multimedia communications.

Certainly, we have to answer several question such as what and how performance can be improved by cooperative communications and networking, why cooperative communications in wireless networks are possible? Note that the wireless channel is broadcast by nature. Even directional transmission is in fact a kind of broadcast with fewer recipients limited to a certain region. This implies that many nodes or users can "hear" and receive transmissions from a source and can help relay information if needed. The broadcast nature, long considered as a significant waste of energy causing interference to others, is now regarded as a potential resource for possible assistance. In contrast, Sharma et al. [SSL+10] has introduced a concept of that network coding noise (NC noise) showing that NC noise can diminish the advantage of NC in cooperative communication.

11.2 Conclusion

In this work, we gathered several delay tolerant routing techniques and tested them under a common umbrella. We have investigated the reason that may help one protocol better than the other under certain conditions while the later may perform better than former in a different scenario. We have investigated the factors that help *flooding* to perform better in sparse networks and those that degrade its performance in congested networks. We have also observed the conditions where non-replication based techniques may perform as good as their replication based counterparts. We have then, combined the positives of the two worlds to obtain a solution that not only scales nicely to the network size but also adapts to available network resources.

Furthermore, we see that the performance of existing algorithms fluctuate heavily among different scenarios, indicating that they may perform well in one scenario, but as soon as underlying networks change, their performance is strongly affected. We have proposed a novel protocol *Nile* that we have designed with the aim of a solution that can be adapted to every situation. The reason why *Nile* adapts very well to different bandwidth and network density scenarios, is the flexible control of replica creation. The relative simplicity of protocol and overall performance of *Nile*, combined with the ability to be further fine-tuned to specific environments, in our opinion makes it the prime choice for any mobile opportunistic network.

We have also highlighted those aspect of *flooding* that can adversely effect its performance to that extent that it cannot be considered to be the upper bound in

every scenario. We have presented insight into two variations of a technique that cannot only be considered a better upper bound (and can be improved further on) but also highlight the hidden aspects of the underlying network. Moreover, we have introduced the notion of how to verify the prediction quality of history in opportunistic networks. We have modified the concept of max-flow computations in such a way that it can be used to estimate the quality of history. We strongly believe that these max-flow computation methods must go through some rigorous statistical refinement. The strategies presented here, exploit all the available resources and help identify the bottlenecks in opportunistic networks, paving the way for development of versatile and efficient routing protocols. Aspects, such as commutative throughput on the diameter of the network and identification of bottlenecks in clusters, will assist us to better understand such networks. In the future, our knowledge enhanced by these finding, will play an important role in developing suitable routing protocols.

Nile has inspired us to create a social expertise sharing system *Mergenet*, that assists user to socialize in way that online social network cannot do. It helps them to make contacts in real life and lets them use these contacts to share as well as gather information. As data intensive applications like MMS and Mobile television are on their way, we intend to test this architecture in real world by focusing on gathering maximum information by transmitting the least amount of data in minimum possible time. This also involves *bloomfilters* compression without loosing their efficiency. As shown by the results, *Mergenet* quickly routes the information without losing privacy but there is much more to be done to call *Mergenet* very secure.

We conclude with the remarks, every opportunistic routing protocol is heavily dependent on the environment in which it operates. We also believe, it is not a bad idea to customize a protocol for one particular situation as the behavior of opportunistic networks varies with the bandwidth variations. We also stress upon the need of a versatile protocol that can be adapted to several imaginable scenarios.

Index

Bibliography

[AGK+] George Apostolopoulos, Roch Guerin, Sanjay Kamat, Ibm T. J. Watson, and Satish K. Tripathi. Improving qos routing performance under inaccurate link state information. In *Proceedings of the 16th International Teletraffic Congress*, pages 7–11.

[AGKT98] George Apostolopoulos, Roch Guérin, Sanjay Kamat, and Satish K. Tripathi. Quality of service routing: A performance perspective. In *Proceedings of ACM SIGCOMM*, Vancouver, BC, Canada, September 1998.

[AIGM93] E. Ayanoglu, Chih-Lin I, R.D. Gitlin, and J.E. Mazo. Diversity coding for transparent self-healing and fault-tolerant communication networks. *Communications, IEEE Transactions on*, 41(11):1677 –1686, November 1993.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, united states ed edition, February 1993.

[AS95] M. Sandilands A.R.H. Swan. *Introduction to geological data analysis*. Oxford ; Cambridge, Mass., USA : Blackwell Science, 1995.

[BBH04] Fan Bai, Ganesha Bhaskara, and Ahmed Helmy. Building the blocks of protocol design and analysis: challenges and lessons learned from case studies on mobile ad hoc routing and micro-mobility protocols. *SIGCOMM Comput. Commun. Rev.*, 34(3):57–70, 2004.

[BBHK10] Michael R. Berthold, Christian Borgelt, Frank Hppner, and Frank Klawonn. *Guide to Intelligent Data Analysis: How to Intelligently Make Sense of Real Data*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[BBL05] Brendan Burns, Oliver Brock, and Brian Neil Levine. Mv routing and capacity building in disruption tolerant networks. In *Proceedings of IEEE Infocom*, 2005.

[BC03a] Magdalena Balazinska and Paul Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 303–316, New York, NY, USA, 2003. ACM.

[BC03b] Magdalena Balazinska and Paul Castro. CRAWDAD data set ibm/watson (v. 2003-02-19). Downloaded from http://crawdad.cs.dartmouth.edu/ibm/watson, February 2003.

[BCGS04] Stefano Basagni, Marco Conti, Silvia Giordano, and Ivan Stojmenovic, editors. *Mobile Ad-hoc Networking*. IEEE press, 2004.

[BGJL06] John Burgess, Brian Gallagher, David Jensen, and BrianNeil Levine. Maxprop: Routing for vehicle-based disruption-tolerantnetworking. In *Proceedings of IEEE Infocom*, Barcelona, Spain, April 2006.

[BHPW04] Daniel Bauer, Paul Hurley, Roman Pletka, and Marcel Waldvogel. Bringing efficient advanced queries to distributed hash tables. *lcn*, 00:6–14, 2004.

[Blo70] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[BS08] Michael Baur and Thomas Schank. Dynamic graph drawing in visone. Technical report, 2008.

[BSH03] Fan Bai, Narayanan Sadagopan, and Ahmed Helmy. The important framework for analyzing the impact of mobility on performance of routing protocols for adhoc networks. *Ad Hoc Networks*, 1(4):383 – 403, 2003.

[CBD02] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[CHC+06] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. In *In Proc. IEEE Infocom*, 2006.

[CL05] Cha and Lee DK. Lee, 2005 Cha M. Split-n-save multiplexing in wireless ad hoc routing. In *Proceedings of the 24th annual joint conference of the IEEE computer and communications societies (INFOCOM),*, Miami, March 2005.

[CMMP08] P. Costa, C. Mascolo, M. Musolesi, and G. P. Picco. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 26(5):748–760, June 2008.

[CXSN03] Kai Chen, Yuan Xue, Samarth H. Shah, and Klara Nahrstedt. Understanding bandwidth-delay product in mobile ad hoc networks. *Computer Communications*, 27:923–934, 2003.

[DF07] Michael Demmer and Kevin Fall. Dtlsr: delay tolerant routing for developing regions. In *NSDR '07: Proceedings of the 2007 workshop on Networked systems for developing regions*, pages 1–6, New York, NY, USA, 2007. ACM.

[DG10] L.Kelly D.Byrne and G.J.F.Jones. *Multiple Multimodal Mobile Devices: Lessons Learned from Engineering Lifelog Solutions*. Amazon, 2010.

[DH07] Elizabeth M. Daly and Mads Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 32–40, New York, NY, USA, 2007. ACM.

[Dim07] *Network Coding for Distributed Storage Systems*, May 2007.

[EP05] Nathan Eagle and Alex (Sandy) Pentland. CRAWDAD data set mit/reality (v. 2005-07-01). Downloaded from http://crawdad.cs.dartmouth.edu/mit/reality, July 2005.

[Fal] Kevin Fall. Delay tolerant network research group.

[FBY00] Richardh. Frenkiel, B. R. Badrinath, and Roy D. Yates. The infostations challenge: Balancing cost and ubiquity in delivering wireless data. *IEEE Personal Communications*, 7:66–71, 2000.

[FCG+06] Stephen Farrell, Vinny Cahill, Dermot Geraghty, Ivor Humphreys, and Paul McDonald. When tcp breaks: Delay- and disruption-tolerant networking. *IEEE Internet Computing*, vol. 10, no. 4:pp. 72–78, 2006.

[FF56] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[GHJVng] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 2007 – 36th printing.

[GN02] Michelle Girvan and M. E. J. Newman. Community structure in social and biological networks. *PROC.NATL.ACAD.SCI.USA*, 99:7821, 2002.

[GT02] Matthias Grossglauser and David N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. Netw.*, 10(4):477–486, August 2002.

[GWCL06] D. Guo, J. Wu, H. Chen, and X. Luo. Theory and network applications of dynamic bloom filters. In *Proceedings INFOCOM 2006. 25th IEEE International Conference on Computer Communications.*, pages 1–12, April 2006.

[GXZ+06] Yili Gong, Yongqiang Xiong, Qian Zhang, Zhensheng Zhang, Wenjie Wang, and Zhiwei Xu. Anycast routing in delay tolerant networks. In *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, 2006.

[HCY08] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 241–250. ACM, 2008.

[HW07] Paul Hurley and Marcel Waldvogel. Bloom filters: One size fits all? *lcn*, 0:183–190, 2007.

[IMA11a] Waldvogel M Islam M. A. Optimizing message delivery in mobile-opportunistic networks. In *Baltic Conference on Futre Inernet Communication*, 2011.

[IMA11b] Waldvogel M Islam M. A. Questioning flooding as a routing benchmark in opportunistic networks. In *Baltic Conference on Futre Inernet Communication*, 2011.

[IW08] Arshad Islam and Marcel Waldvogel. Reality-check for dtn routing algorithms. In *ICDCSW '08: Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems Workshops*,

pages 204–209, Washington, DC, USA, 2008. IEEE Computer Society.

[JA06] Peter Janssen and Werner Alpers. Why sar wave mode data of ers and envisat are inadequate for giving the probability of occurrence of freak waves. In *Proceedings of SEASAR 2006, SP-613*, ESA - ESRIN, Frascati, Italy, 2006. ESA.

[JDPF05] Sushant Jain, Michael Demmer, Rabin Patra, and Kevin Fall. Using redundancy to cope with failures in a delay tolerant network. In *Proceedings of ACM SIGCOMM*, pages 109–120, New York, NY, USA, 2005. ACM Press.

[JFP04] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a delay tolerant network. In *Proceedings of SIGCOMM 2004*, pages 145–158. ACM Press, 2004.

[JLS07] Evan P. C. Jones, Lily Li, and Jakub K. Schmidtke. Practical routing in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 6(8):943–959, 2007.

[JM96] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[JOW+02] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *ASPLOS-X*, 2002.

[JW05] E. P. Jones and P. A. Ward. Routing strategies for delay-tolerant networks. In *ACM Computer Communication Review (CCR).*, 2005.

[KJRLK09] Weifeng Su K. J. Ray Liu, Ahmed Sadek and Andres Kwasinski. *Cooperative Communications and Networking*. Cambridge Univ Press, Hardbound, 2009.

[KL86] Brian Kantor and Phil Lapsley. Network news transfer protocol. Technical Report 977, February 1986.

[KMK04] Anurag Kumar, D. Manjunath, and Joy Kuri. *Communication Networking: An Analytical Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[KMM03] Ralf Koetter, Muriel MÃľdard, and Senior Member. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11:782–795, 2003.

[KO07] A. Keranen and J. Ott. Increasing reality for dtn protocol simulations. *Helsinki University of Technology, Tech. Rep., July*, 2007.

[KOK09] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The one simulator for dtn protocol evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ICST.

[KSB09] Jonghyun Kim, Vinay Sridhara, and Stephan Bohacek. Realistic mobility simulation of urban mesh networks. *Ad Hoc Netw.*, 7(2):411–430, 2009.

[LDLE08] Anders Lindgren, Avri Doria, Jan Lindblom, and Mattias Ek. Networking in the land of northern lights: two years of experiences from dtn system deployments. In *WiNS-DR '08: Proceedings of the 2008 ACM workshop on Wireless networks and systems for developing regions*, pages 1–8, New York, NY, USA, 2008. ACM.

[LDS03] Anders Lindgren, Avri Doria, and Olov Schelen. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, July 2003.

[Lee02] *Split multipath routing with maximally disjoint paths in ad hoc networks*, volume 10, August 2002.

[LFC05] Jérémie Leguay, Timur Friedman, and Vania Conan. Dtn routing in a mobility pattern space. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 276–283, New York, NY, USA, 2005. ACM.

[LH08] Long Le and Ekram Hossain. Tandem queue models with applications to qos routing in multihop wireless networks. *IEEE Transactions on Mobile Computing*, 7:1025–1040, August 2008.

[LKG06] Leszek Lilien, Z. Huma Kamal, and Ajay Gupta. Opportunistic networks: Challenges in specializing the p2p paradigm. *Database and Expert Systems Applications, International Workshop on*, 0:722–726, 2006.

[LTZG06] Yong Liao, Kun Tan, Zhensheng Zhang, and Lixin Gao. Estimation based erasure-coding routing in delay tolerant networks. In *Proceedings of IWCMC*, June 2006.

[MD01] M. K. Marina and S. R. Das. On-demand multipath distance vector routing in ad hoc networks. In *in Proceedings of IEEE International Conference on Network Protocols (ICNP)*, pages 14–23, 2001.

[MGC+07] Andrew Miklas, Kiran Gollu, Kelvin Chan, Stefan Saroiu, Krishna Gummadi, and Eyal de Lara. Exploiting social interactions in mobile systems. pages 409–428. 2007.

[MHM05] Mirco Musolesi, Stephen Hailes, and Cecilia Mascolo. Adaptive routing for intermittently connected mobile ad hoc networks. In *in Proc. WOWMOM*, pages 183–189. IEEE press, 2005.

[Mil67] Stanley Milgram. The small world problem. *Psychology Today*, 1:60 – 67, May 1967.

[Mit01] M. Mitzenmacher. Compressed bloom filters. In *Proc. of the 20th Annual ACM Symposium on Principles of Distributed Computing*, IEEE/ACM Trans. on Networking, pages 144–150, 2001.

[MM06] Mirco Musolesi and Cecilia Mascolo. A community based mobility model for ad hoc network research. In *REALMAN '06: Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, pages 31–38, New York, NY, USA, 2006. ACM.

[MM07] Mirco Musolesi and Cecilia Mascolo. Designing mobility models based on social network theory. *ACM SIGMOBILE Mobile Computing and Communication Review*, 11:59–70, 2007.

[MS05] Ghosal D. Mueller S. Analysis of a distributed algorithm to determine multiple routes with path diversity in ad hoc networks. In *Proceedings of the third international symposium on modeling and optimization in mobile, ad hoc, and wireless networks (WIOPT)*, page 277âĂŞ85, 2005.

[MWYL04] Xiaoqiao (George) Meng, Starsky H. Y. Wong, Yuan Yuan, and Songwu Lu. Characterizing flows in large wireless data networks. pages 174–186, 2004.

[PB94] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, 1994.

[PFH04] Alex (Sandy) Pentland, Richard Fletcher, and Amir Hasson. Daknet: Rethinking connectivity in developing nations. *IEEE Computer*, 37(1):78–83, January 2004.

[PKC96] Kihong Park, Gitae Kim, and Mark Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. *Network Protocols, IEEE International Conference on*, 0:171, 1996.

[PP03] Perrau S. Pham PP. Performance analysis of reactive shortest path and multipath routing mechanism with load balance. In *Proceedings of the IEEE INFOCOM*, pages 251–9, San Francisco, California, USA,, April 2003.

[PR97] Charles Perkins and Elizabeth Royer. Ad-hoc on-demand distance vector routing. In *In Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1997.

[PRSR06] Lucian Popa, Costin Raiciu, Ion Stoica, and David Rosenblum. Reducing congestion effects in wireless networks by multipath routing. In *ICNP '06: Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 96–105, Washington, DC, USA, 2006. IEEE Computer Society.

[Ree99] David P. Reed. That sneaky exponential—Beyond Metcalfe's law to the power of community building. http://www.reed.com/Papers/GFN/reedslaw.html, 1999.

[RK02] S.C. Rhea and J. Kubiatowicz. Probabilistic location and routing. In *Proceedings. IEEE INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1248 – 1257, November 2002.

[SBKH03] Narayanan Sadagopan, Fan Bai, Bhaskar Krishnamachari, and Ahmed Helmy. Paths: analysis of path duration statistics and their impact on reactive manet routing protocols. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 245–256, New York, NY, USA, 2003. ACM.

[SCE+05] Lakshminarayanan Subramanian, Matthew Caesar, Cheng Tien Ee, Mark Handley, Morley Mao, Scott Shenker, and Ion Stoica. Hlp: a next generation inter-domain routing protocol. In *SIGCOMM '05:*

Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, pages 13–24, New York, NY, USA, 2005. ACM.

[Sch05] Jan Aart Scholte. Globalization: A critical introduction,. *Basingstoke: Palgrave,*, I:p. 68., 2005,.

[SPR05] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *ACM workshop on Delay Tolerant Networking*, 2005.

[SPR06] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Performance analysis of mobility-assisted routing. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '06, pages 49–60, New York, NY, USA, 2006. ACM.

[Spy07] *Utility-based Message Replication for Intermittently Connected Heterogeneous Networks*, 2007.

[SRJB03] Rahul C. Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. Data MULEs: Modeling a three-tier architecture for sparse sensor networks. In *Intel Research Technical Report*, 2003.

[SSL+10] Sushant Sharma, Yi Shi, Jia Liu, Y. Thomas Hou, and Sastry Kompella. Is network coding always good for cooperative communications? In *Proceedings of IEEE INFOCOM 2010*, pages 1–9. IEEE, March 2010.

[TG05] C. Tuduce and T. Gross. A mobility model based on wlan traces and its validation. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 1, pages 664–674 vol. 1, 2005.

[TH01] A. Tsirigos and Z. J. Haas. Multipath routing in the presence of frequent topological changes. *Communications Magazine, IEEE*, 39(11):132–138, 2001.

[TSR08a] Konstantinos Psounis Thrasyvoulos Spyropoulos and Cauligi Raghavendra. Efficient routing in intermittently connected mobile networks: The multiple-copy case. In *ACM/IEEE journal of Transactions on Networking,*, 2008.

[TSR08b] Konstantinos Psounis Thrasyvoulos Spyropoulos and Cauligi Raghavendra. Efficient routing in intermittently connected mobile networks: The single-copy case. In *ACM/IEEE journal of Transactions on Networking,*, 2008.

[TTAE09] Mohammed Tarique, Kemal E. Tepe, Sasan Adibi, and Shervin Erfani. Survey of multipath routing protocols for mobile ad hoc networks. *Journal of Network and Computer Applications*, 32(6):1125 – 1143, 2009.

[UNC06] UNCTAD. The information economy report the development perspective, new york. Technical report, United Nations Conference on Trade and Development, 2006.

[VB00] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, April 2000.

[WH91] Ross Wilkinson and Philip Hingston. Using the cosine measure in a neural network for document retrieval. In *SIGIR '91: Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 202–210, New York, NY, USA, 1991. ACM.

[WH01] Kui Wu and Janelle Harms. On-demand multipath routing for mobile ad hoc networks. In *Networks EPMCC 2001, Vienna, 20th âĂŞ 22nd February 2001*, pages 1–7, 2001.

[WJMF05] Y. Wang, S. Jain, M. Martonosi, and K. Fall. Erasure-coding based routing for opportunistic networks. *ACM workshop on Delay Tolerant Networking*, 2005.

[WL05] Lee T-Y. Wang L, Jang S. Redundant source routing for real-time services in ad hoc networks. In *Proceedings of IEEE international conference on mobile ad hoc and sensor systems conference,*, Washington, DC,, November 2005.

[WL09] Chia-Hung Wang and Hsing Luh. *Blocking Probabilities of Multiple Classes in IP Networks with QoS Routing*, chapter 16, pages 281–290. Springer, 2009.

[YZ03] Tripathi SK. Ye Z, Krishnamurthy SV. A framework for reliable routing in mobile ad hoc networks. In *Proceedings of the 22nd annual joint*

conference of the IEEE computer and communications societies (INFOCOM),, volume I, page 270âĂŞ80., 2003.

[ZA03] Wenrui Zhao and Mostafa H. Ammar. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. pages 308–314, 2003.

[ZAZ05] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura. Multicasting in delay tolerant networks: semantic models and routing algorithms. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 268–275, New York, NY, USA, 2005. ACM.

[Zin10] Michael Zinsmaier. Performance analysis of routing algorithms in interest-based mobile social networks. Technical report, University of Konstanz, 2010.