

DESIGNING EFFICIENT ROUTING PROTOCOLS IN DELAY  
TOLERANT NETWORKS

---

A DISSERTATION  
SUBMITTED  
TO THE TEMPLE UNIVERSITY GRADUATE BOARD

---

IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY

---

BY  
YUNSHENG WANG  
AUGUST, 2013

*Supervisor Committee:*

Dr. Jie WU (Advisor), Department of Computer and Information Sciences

Dr. Justin Y. SHI, Department of Computer and Information Sciences

Dr. Chiu C. TAN, Department of Computer and Information Sciences

*External Readers for Dissertation Defense:*

Dr. Li BAI, Department of Electrical and Computer Engineering

## ABSTRACT

This thesis presents the design and evaluation of routing protocols for efficient content delivery and dissemination in delay tolerant networks. With the advancement in technology, the communication devices with wireless interfaces become more and more universal. Delay tolerant networks (DTNs) are characterized by intermittent connectivity and limited network capacity. There exist several different application scenarios: connectivity of developing countries, vehicular DTN road communications, and social contact networks.

In this thesis, we explore the characteristics in DTNs, such as mobility pattern, contact history information, and social feature information, to design efficient routing schemes. The research reported in this thesis investigates the technical challenges and their solutions of applying different DTN routing protocols. We design multicast schemes to forward the information to a group of destinations in DTN environment. We extend the delegation forwarding scheme in DTN multicasting. A non-replication multicast tree is also studied in this report. We also apply ticket-based and social-tie-based approaches in content distribution systems. We leverage the users' social feature information to study the hypercube-based routing schemes in social contact networks. We also study the resource management problem in DTNs. We design a joint replication-migration-based scheme to solve the storage congestion. These techniques are evaluated comprehensively in realistic simulation studies, by comparing the performance with state-of-the-art approaches in both synthetic and real traces.

**Keywords:** Delay tolerant networks, opportunistic mobile social networks, routing, system design and evaluation

# ACKNOWLEDGEMENT

This research would not have been possible without the constant support of my advisor, supervisors, and many anonymous people, to whom I'll always be indebted to in life. It has been a great privilege to spend several years in the Department of Computer and Information Sciences at Temple University, and its members will always remain dear to me.

My first debt of gratitude must go to my advisor, Dr. Jie Wu, who is the most important person guiding me and shaping the research. He patiently provided the vision, encouragement, and advice necessary for me to proceed through the doctoral program and complete my dissertation. Without his tutoring, I would have not able to achieve as much as I have. I want to thank Dr. Jie Wu for his unflagging encouragement and serving as a role model to me as a junior member of academia.

Special thanks to my committee, Dr. Justin Y. Shi and Dr. Chiu C. Tan for their support, guidance and helpful suggestions. Their guidance has served me well and I owe them my heartfelt appreciation. Their valuable comments have helped me refine my research in many ways that are impossible to achieve on my own. I am also grateful to Dr. Li Bai, who spends his valuable time on advising during my thesis defense.

My sincere thanks also goes to Dr. Wei-Shih Yang, Dr. Yuhong Guo, Dr. Mingjun Xiao, Dr. Zhen Jiang, and Dr. Feng Li, for all their help, support, interest and valuable hints in my research.

Also I thank my advisors Dr. Huchuan Lu in Dalian University of Technology, China, and Dr. Yang Yang in University College London, UK. In particular, I am grateful to Dr. Yang Yang for enlightening me the first glance of research.

I thank my fellow labmates in Network Computer Center, for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last four years.

I wish to thank my parents, Keming Wang and Yingyan Wang, and my grandparents Zhenshi Wang and Wenju Zhen. Their love provided my inspiration and was my driving force. I owe them everything and wish I could show them just how much I love and appreciate them.

To my parents

# TABLE OF CONTENTS

<b>ABSTRACT</b>	<b>ii</b>
<b>ACKNOWLEDGEMENT</b>	<b>iii</b>
<b>LIST OF FIGURES</b>	<b>viii</b>
<b>LIST OF TABLES</b>	<b>xiii</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Motivation and Challenge . . . . .	1
1.2 Major Contributions . . . . .	3
1.3 Dissertation Overview . . . . .	4
<b>2 LITERATURE REVIEW</b>	<b>5</b>
2.1 DTN Architecture . . . . .	5
2.2 DTN Multicasting . . . . .	6
2.2.1 Single Node Multicasting . . . . .	6
2.2.2 Multiple Copies Multicasting . . . . .	7
2.2.3 Single Copy Model . . . . .	8
2.3 DTN Broadcast . . . . .	9
2.4 Social-aware Routing . . . . .	9
2.5 Resource Management . . . . .	10
<b>3 MULTICAST IN DTNS</b>	<b>11</b>
3.1 Delegation Forwarding Multicast . . . . .	11
3.1.1 Preliminary work . . . . .	12
3.1.2 Single copy multicast . . . . .	13
3.1.3 Multiple copy multicast . . . . .	13
3.1.4 Delegation forwarding multicast . . . . .	14
3.1.5 Analysis . . . . .	15
3.1.6 Simulation . . . . .	20
3.1.7 Conclusion . . . . .	22
3.2 Non-Replication Multicast . . . . .	22
3.2.1 System Model . . . . .	23
3.2.2 Challenges and Main Ideas . . . . .	24
3.2.3 Compare-split . . . . .	25
3.2.4 Implementation & Extensions . . . . .	28
3.2.5 Wait and Focus . . . . .	29
3.2.6 Analysis . . . . .	30
3.2.7 Simulation . . . . .	32
3.2.8 Conclusion . . . . .	41
3.3 Cloud-based Multicasting with Feedback . . . . .	42
3.3.1 Introduction . . . . .	42
3.3.2 Cloud-based Multicast . . . . .	44
3.3.3 Analysis . . . . .	49

3.3.4	Simulation . . . . .	55
3.3.5	Conclusion . . . . .	61
3.4	Conclusion of the Chapter . . . . .	61
<b>4</b>	<b>BROADCAST IN DTNS</b>	<b>62</b>
4.1	Making Many People Happy: Greedy Solutions for Content Distribution . . . . .	62
4.1.1	Introduction . . . . .	62
4.1.2	Optimization problem . . . . .	64
4.1.3	Greedy heuristic algorithm . . . . .	69
4.1.4	Simulation . . . . .	76
4.1.5	Conclusion . . . . .	81
4.2	Ticket-based Multiple Packet Broadcasting . . . . .	81
4.2.1	Introduction . . . . .	81
4.2.2	Multiple Packet Broadcasting . . . . .	83
4.2.3	Simulation . . . . .	86
4.2.4	Conclusion . . . . .	93
4.3	Social-Tie-Based Information Dissemination . . . . .	93
4.3.1	Introduction . . . . .	94
4.3.2	Social-Tie-Based Information Dissemination . . . . .	96
4.3.3	Simulation and Evaluation . . . . .	101
4.3.4	Conclusion . . . . .	104
4.4	Conclusion of the Chapter . . . . .	105
<b>5</b>	<b>SOCIAL-AWARE ROUTING IN DTNS</b>	<b>106</b>
5.1	Hypercube-based Social Feature Routing . . . . .	106
5.1.1	Introduction . . . . .	107
5.1.2	Preliminaries . . . . .	108
5.1.3	Feature Extraction . . . . .	110
5.1.4	Multi-Path Routing . . . . .	110
5.1.5	Node-Disjointness . . . . .	114
5.1.6	Contact Frequency . . . . .	114
5.1.7	Extensions . . . . .	117
5.1.8	Simulation . . . . .	119
5.1.9	Conclusion . . . . .	124
5.2	Analysis of a Hypercube-based Social Feature Multi-Path Routing . . . . .	125
5.2.1	Introduction . . . . .	125
5.2.2	Analysis . . . . .	125
5.2.3	Discussion . . . . .	133
5.2.4	Simulation . . . . .	133
5.2.5	Conclusion . . . . .	138
5.3	Conclusion of the Chapter . . . . .	138
<b>6</b>	<b>RESOURCE MANAGEMENT IN DTNS</b>	<b>139</b>
6.1	A Joint Replication-Migration-based Routing . . . . .	139
6.1.1	Introduction . . . . .	140
6.1.2	Replication-Migration-based Routing . . . . .	141
6.1.3	Implementation and Simulation . . . . .	147
6.1.4	Conclusion . . . . .	153
6.2	Conclusion of the Chapter . . . . .	154
<b>7</b>	<b>CONCLUSION</b>	<b>155</b>
7.1	Summary of Contribution . . . . .	155
7.2	Future Research . . . . .	156
7.2.1	Routing Protocols . . . . .	156
7.2.2	Social Network Analysis . . . . .	157
7.2.3	Mobile Cloud Networks . . . . .	157

<b>BIBLIOGRAPHY</b>	<b>158</b>
<b>PUBLICATION LIST</b>	<b>164</b>

# LIST OF FIGURES

1.1	Geographical Distribution of Delay Tolerant Networks (DTNs). . . . .	2
1.2	A DTN System, its <i>Service Targets</i> and its <i>System Constraints</i> . . . . .	3
1.3	Overview of the Dissertation. . . . .	4
2.1	An illustration of single node model in DTN multicasting. . . . .	7
3.1	Multicast tree: (a) single copy; (b) multiple copy and delegation forwarding. . . . .	12
3.2	Single copy multicast in DTNs. . . . .	13
3.3	Multiple copy multicast in DTNs. . . . .	14
3.4	Delegation forwarding multicast in DTNs. . . . .	15
3.5	Comparison of the number of forwardings in synthetic traces. . . . .	20
3.6	Comparison of latency in synthetic traces. . . . .	20
3.7	Comparison of the number of forwardings in real traces. . . . .	21
3.8	Comparison of latency in real traces. . . . .	21
3.9	An illustration of the system model. . . . .	24
3.10	An illustration of ratio-based-split. . . . .	25
3.11	An illustration of <i>compare-split</i> . . . . .	26
3.12	An example of <i>ratio-based-split</i> . . . . .	27
3.13	A sample of binary-split. . . . .	29
3.14	A sample of priority-based-split. . . . .	30
3.15	Levy walks model. . . . .	34
3.16	Comparison of two <i>compare</i> methods. . . . .	35
3.17	Comparison in Levy walks model: compare-split-wait in N-RI. . . . .	36
3.18	Comparison in Levy walks model: compare-split-focus in N-RI. . . . .	36
3.19	Comparison in Gaussian distribution model: compare-split-wait in N-RI. . . . .	37
3.20	Comparison in Gaussian distribution model: compare-split-focus in N-RI. . . . .	38
3.21	Comparison in Intel trace: compare-split-wait in N-RI. . . . .	38

3.22 Comparison in Intel trace: compare-split-focus in N-RI. . . . .	38
3.23 Comparison in Cambridge trace: compare-split-wait in N-RI. . . . .	38
3.24 Comparison in Cambridge trace: compare-split-focus in N-RI. . . . .	39
3.25 Comparison in Levy walks model: compare-split-wait in RI. . . . .	39
3.26 Comparison in Levy walks model: compare-split-focus in RI. . . . .	39
3.27 Comparison in Gaussian distribution model: compare-split-wait in RI. . . . .	39
3.28 Comparison in Gaussian distribution model: compare-split-focus in RI. . . . .	40
3.29 Comparison in Intel trace: compare-split-wait in RI. . . . .	40
3.30 Comparison in Intel trace: compare-split-focus in RI. . . . .	40
3.31 Comparison in Cambridge trace: compare-split-wait in RI. . . . .	40
3.32 Comparison in Cambridge trace: compare-split-focus in RI. . . . .	41
3.33 Cloud-based Multicast in MSNs. . . . .	44
3.34 Motivation of the feedback mechanism: the values between the mobile nodes are the inter contact time. . . . .	47
3.35 Destination feedback process. . . . .	48
3.36 Metadata update process. . . . .	49
3.37 Comparing the performance in the 4 destinations situation: ( $L$ ): latency and ( $R$ ): number of forwardings. . . . .	55
3.38 Comparing the performance in the 32 destinations situation: ( $L$ ): latency and ( $R$ ): number of forwardings. . . . .	55
3.39 Comparing the performance in the Intel trace in a variety of cloud sizes: ( $L$ ): latency and ( $R$ ): number of forwardings. . . . .	57
3.40 Comparing the performance in the Infocom 2006 trace in a variety of cloud sizes: ( $L$ ): latency and ( $R$ ): number of forwardings. . . . .	57
3.41 Comparing the performance in the Intel trace in different numbers of destinations in each round: ( $L$ ): latency and ( $R$ ): number of forwardings. . . . .	58
3.42 Comparing the performance in the Infocom 2006 trace in different numbers of desti- nations in each round: ( $L$ ): latency and ( $R$ ): number of forwardings. . . . .	58
3.43 Comparing the performance in the Intel trace in different number of destinations: ( $L$ ): latency and ( $R$ ): number of forwardings. . . . .	59
3.44 Comparing the performance in the Infocom 2006 trace in different number of destina- tions: ( $L$ ): latency and ( $R$ ): number of forwardings. . . . .	59
3.45 Percentage of the nodes that received the feedback information: ( $L$ ): Intel and ( $R$ ): Infocom. . . . .	60

4.1	Content distribution in a sequence of $k$ broadcast in the $m$ -D space. . . . .	63
4.2	Comparison of approximation ratio in 10-node and 40-node environments. . . . .	72
4.3	Greedy algorithms: greedy 2: (a) - (d); greedy 3: (e) - (h); greedy 4: (i) - (l) (different symbols of the points mean different weights: 5: *; 4: $\square$ ; 3: $\diamond$ ; 2: +; 1: $\circ$ . * is the centers). . . . .	75
4.4	Comparison in 2-norm in a 2-D space using different weights. . . . .	77
4.5	Comparison in 2-norm in a 2-D space using the same weights. . . . .	77
4.6	Comparison in 1-norm in a 2-D space using different weights. . . . .	78
4.7	Comparison in 1-norm in a 2-D space using the same weights. . . . .	79
4.8	Comparison in 1-norm in a 3-D space using different weights. . . . .	80
4.9	Comparison in 1-norm in a 2-D space using the same weights. . . . .	80
4.10	An example for multiple packet broadcasting: circle nodes with packet 1, star nodes with packet 2, square nodes with both packets, and diamond nodes with no packets. . . . .	82
4.11	Area division. . . . .	84
4.12	Ticket-based multiple packet DTN broadcasting. . . . .	85
4.13	Average latency comparisons in Levy walks model ((a) and (b)) and community-based mobility model ((c) and (d)). . . . .	87
4.14	Useless contacts comparison in synthetic traces: Levy walks model ((a) and (b)) and community-based mobility model ((c) and (d)). . . . .	88
4.15	Comparison of ticket-based scheme and combined scheme in synthetic traces: Levy walks model ((a) and (b)) and community-based mobility model ((c) and (d)). . . . .	89
4.16	Average latency comparisons in the NCSU trace ((a) and (b)) and the North Carolina state fair trace ((c) and (d)). . . . .	90
4.17	Useless contacts comparison in real traces: the NCSU trace ((a) and (b)) and the North Carolina state fair trace ((c) and (d)). . . . .	91
4.18	Comparison of the ticket-based scheme and the combined schemes in real traces: the NCSU trace ((a) and (b)) and the North Carolina state fair trace ((c) and (d)). . . . .	92
4.19	An illustration of the Facebook friends network: the nodes are the Facebook users. The size of the nodes show the degree of the nodes. Larger nodes have larger degrees. The thickness of the links show the social tie strength between pairwise nodes. Thicker links have larger tie strength. . . . .	94
4.20	An illustration of the local bridge linking two communities: <i>solid</i> lines are the <i>strong</i> ties and dashed lines are the weak ties. . . . .	95
4.21	The percentage of involved weak ties in MIT reality mining data. . . . .	96

4.22	An illustration of a five-nodes social network. The value $(x/y)$ on the links represent the number of contacts/the social feature distance of linked vertices. . . . .	99
4.23	Comparing the performance of different schemes in different initial numbers of tokens in the MIT reality mining trace: ( $L$ ): delivery ratio; ( $R$ ): latency. . . . .	103
4.24	Comparing the performance of different schemes in different initial numbers of tokens in the Infocom2006 trace: ( $L$ ): delivery ratio; ( $R$ ): latency. . . . .	103
4.25	Comparing the performance of different schemes in different delivery ratios: ( $L$ ): MIT; ( $R$ ): Infocom. . . . .	104
5.1	Comparison of the contacts in the Infocom 2006 trace. . . . .	107
5.2	A 3-dimensional hypercube. . . . .	109
5.3	An example of node-disjoint-based routing with $S = G_0$ and $D = G_6$ . Solid directed paths are the shortest paths, and dashed directed paths represent the non-shortest paths. . . . .	113
5.4	An example of the composite path from 0000 to 1111, where dashed directed lines are shortcuts. . . . .	116
5.5	An illustration of contact frequency. . . . .	117
5.6	Comparing the delivery rate in the real trace: (left: $L$ ): 20 packets and (right: $R$ ): 100 packets. . . . .	119
5.7	Comparing the latency in the real trace: ( $L$ ): 20 packets and ( $R$ ): 100 packets. . . . .	119
5.8	Comparing the number of forwardings in the real trace: ( $L$ ): 20 packets; ( $R$ ): 100 packets. . . . .	120
5.9	Comparing <i>shortcut</i> and <i>non-shortcut</i> with 100 packets in the real trace: ( $L$ ): delivery rate, ( $Center$ ): latency, and ( $R$ ): number of forwardings. . . . .	121
5.10	Comparing with 100 packets in the synthetic trace: ( $L$ ): delivery rate, ( $Center$ ): latency, and ( $R$ ): number of forwardings. . . . .	121
5.11	Comparing in the percentage of the successful non-shortest path: ( $L$ ): real trace and ( $R$ ): synthetic trace. . . . .	122
5.12	Comparing in general hypercubes in the real trace: ( $L$ ): delivery rate and ( $R$ ): latency. . . . .	122
5.13	Comparing in CCCs in the real trace: ( $L$ ): delivery rate and ( $R$ ): latency. . . . .	123
5.14	An illustration of the shortcut fast feature matching process. . . . .	126
5.15	Comparison between non-shortcut and 2-hop shortcuts including in the single-path case. ( $x(i)$ is $x_j^i$ for $\forall j$ .) . . . . .	128
5.16	Comparison between non-shortcut and all shortcuts including in the single-path case. . . . .	128

5.17	Comparison of the performance after including different numbers of hop shortcuts. . . . .	129
5.18	Comparison between non-shortcut and all shortcuts including in the multi-path case. . . . .	129
5.19	Comparison of the performance between <i>multi-path</i> and <i>single path</i> : ( <i>L</i> ): delivery rate and ( <i>R</i> ): latency. ( <i>M-P</i> : multi-path, <i>S-P</i> : single-path; <i>A</i> : analysis, <i>S</i> : synthetic, <i>I</i> : Infocom, and <i>M</i> : MIT.) . . . . .	135
5.20	Comparison of the performance between <i>shortcut</i> and <i>non-shortcut</i> : ( <i>L</i> ): delivery rate and ( <i>R</i> ): latency. ( <i>S</i> : shortcut, <i>NS</i> : non-shortcut.) . . . . .	135
5.21	Comparing the delivery rate: ( <i>L</i> ): synthetic, ( <i>Center</i> ): Infocom, and ( <i>R</i> ): MIT. . . . .	136
5.22	Comparing the latency: ( <i>L</i> ): synthetic, ( <i>Center</i> ): Infocom, and ( <i>R</i> ): MIT. . . . .	137
6.1	Comparison delivery rate and latency in different hop-count thresholds using message replication. . . . .	141
6.2	An illustration of message replication. . . . .	142
6.3	An illustration of message migration. . . . .	144
6.4	An example of the replication-migration-based scheme and replication scheme comparison: there are 5 nodes in the network. All nodes have the same buffer size, which can contain 2 messages. . . . .	146
6.5	Comparison between migration and dropping in different buffer space thresholds in the synthetic trace. . . . .	148
6.6	Comparison of delivery rate, latency, number of forwardings, and number of lost messages in different hop-count thresholds in the synthetic trace. . . . .	149
6.7	Comparison between 2-hop and 1-hop in the synthetic trace. . . . .	149
6.8	Comparison between the methods by using community properties, and without using community properties, in different buffer space thresholds in the synthetic trace. . . . .	150
6.9	Comparison between migration and dropping in different buffer space thresholds in the real trace. . . . .	151
6.10	Comparison of delivery rate, latency, number of forwardings, and number of lost messages in different hop-count thresholds in the real trace. . . . .	152
6.11	Comparison between 2-hop and 1-hop in the real trace. . . . .	152
6.12	Comparison between the methods by using community properties, and without using community properties, in different buffer space thresholds in the real trace. . . . .	153

# LIST OF TABLES

2.1	Multicast Routing Schemes in DTNs. . . . .	6
3.1	Different split methods. . . . .	28
3.2	Simulation parameters . . . . .	33
3.3	Compare-Split-Wait legend of simulation results . . . . .	34
3.4	Compare-Split-Focus legend of simulation results . . . . .	35
3.5	Conclusion of simulation results . . . . .	42
4.1	Comparison of three greedy heuristic algorithms. . . . .	76
4.2	Social features in datasets. . . . .	97
4.3	The performance of different tie strength calculation schemes in MIT reality mining trace . . . . .	102
4.4	The performance of different tie strength calculation schemes in Infocom2006 trace . . . . .	102
4.5	The performance of our scheme in different two phases' switch thresholds in MIT reality mining trace . . . . .	105
4.6	The performance of our scheme in different two phases' switch thresholds in Infocom2006 trace . . . . .	105
5.1	Entropy of the social features in the Infocom 2006 trace. . . . .	111
5.2	Comparison of contact frequency with different feature distance in the Infocom 2006 trace. . . . .	115
5.3	Comparison of shortcut and non-shortcut in the multi-path case. . . . .	131
5.4	The value of $c_k$ . . . . .	132
6.1	Encounter history. . . . .	145
6.2	Simulation parameters. . . . .	147

# Chapter 1

## INTRODUCTION

Networking for challenged environments, or *delay tolerant networks* (DTNs) [1] as it is now most commonly referred to, has attracted great attention in the past few years by the networking research community. Connectivity disruptions, limited network capacity, energy and storage constraints of the participating, mobile devices and the arbitrary movement of nodes are only a few of the challenges that the protocol stack has to deal with. Clearly, current Internet protocols (i.e., the TCP/IP protocol) suffer and can fail under such conditions.

### 1.1 Motivation and Challenge

Research on DTN-related issues has already expanded and covers a wide variety of environments. For example, the DTN Research Group [2] focuses mainly on Deep-Space communications and the Bundle Protocol (which is expected to be deployed in some terrestrial environments as well), while the broader DTN research community has investigated delay-tolerant communications for developing countries, vehicular networks and social networking, just to name a few. Clearly, there is huge heterogeneity between different application environments as well as between the participating DTN-nodes and their resource availability. For instance, “line-of-sight” times (and therefore connectivity) for Deep-Space communications is known well in advance, hence routing is not much of an issue and becomes more of a scheduling and resource allocation problem. The case, however, is not the same for a vehicular DTN. That said, application metrics and service targets differ accordingly. Fig. 1.1 shows the research fields of delay tolerant networks.

Clearly, there exists huge diversity among the several different application scenarios for DTNs. So, the challenging question that one may pose, is: “Can one protocol stack deal with all potential DTN application scenarios?”. For example, application diversity in the Internet is handled through

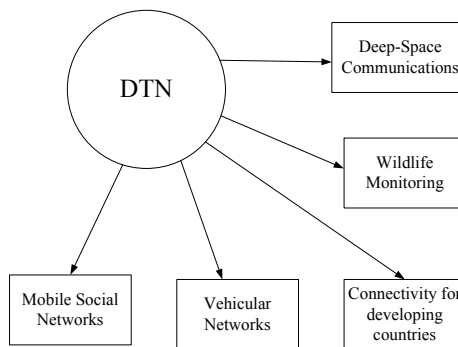


Figure 1.1: Geographical Distribution of Delay Tolerant Networks (DTNs).

the homogeneous common TCP/IP protocol stack and innovation is mainly pushed to the application layer. However, in the terrestrial Internet, continuous connectivity can be assumed in the majority of instances and delays are very small allowing for rapid responses and use of tight, closed control loops. In DTNs the situation is different: connectivity comprises the exception rather than the rule and control loops, like end-to-end connectivity and the path between source and destination, are not closed. Therefore, timeliness, in terms of data delivery deadlines, may be difficult to meet.

Moreover, it is not defined yet whether 100% reliability (in terms of delivery rate) is always required. For example, although reliable transmission is essential for applications such as telecommand, or e-mail, the case may be different for telemetry or road traffic management. If (non-critical) telemetry packets are collected on a one-second basis, for instance, then loss of one (or more) packets will probably not have major impact on the outcome of the monitoring application. In contrast, trying to provide full reliability to a telemetry application may negatively affect the system's performance as a whole. Imagine, for example, a sequence of telemetry packets being trapped in a portion of a DTN. This information will keep on consuming valuable network resources (i.e., storage space) and in turn, the node will not be able to serve new nodes/applications, due to storage space shortage.

We consider there exist three ultimate goals a DTN system should attempt to achieve:

1. (*high*) **delivery rate**: the average delivery ratio of the routing packet. The ultimate goal of the DTN routing design is to find the best possible path(s) to the destination(s).
2. (*low*) **latency**: the average delivery time for the delivered destination(s) to receive the message.
3. (*small*) **number of forwardings**: the average number of forwardings for the destination(s) to receive the message. This value represents the overhead in the network in terms of how many times a message must take in order to reach the destination(s).

We call these three goals the *service targets* of a DTN system. Given that DTNs consist mainly of mobile, battery-powered devices, they will be constrained with regard to: (1) energy consumption

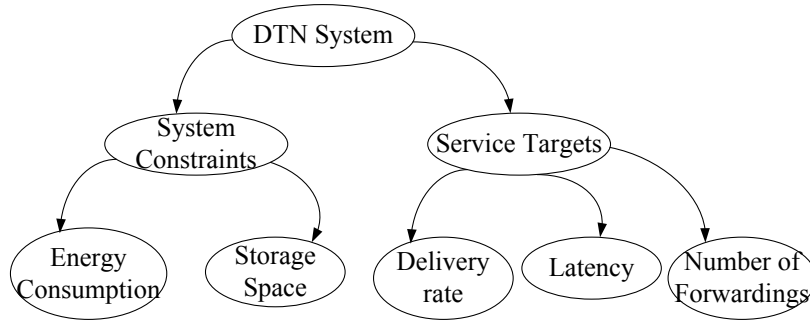


Figure 1.2: A DTN System, its *Service Targets* and its *System Constraints*.

and (2) storage space. We refer to these constraints as *system constraints* (Fig. 1.2).

## 1.2 Major Contributions

In this thesis, we carry out the task of designing and deploying different DTN routing approaches. Our study can be categorized into the following parts: *multicast*, *content dissemination*, *social-aware routing*, and *resource management*. While routing in the Internet and mobile ad hoc networks has been studied extensively, efficient routing in DTNs is a considerably different and challenging problem due to the probabilistic nature of contact among nodes. Our major contributions are the following:

- In the DTN multicast scenario, we design efficient multicasting schemes compared with existing state-of-the-art approaches. We extend the delegation forwarding into DTN multicast. A non-replication multicast tree is designed for information forwarding guidance. We also study the community structure and propose a cloud-based multicast scheme in DTNs.
- Broadcast is also an important application in DTNs. In this thesis, we propose a greed solution to satisfy as many users as possible. We also design ticket-based and social-tie-based schemes for content distribution in DTN environment. In the ticket-based multiple packet broadcasting scheme, we study the human mobility patterns for packet selection and relay node selection. In the social-tie-based content distribution approach, we utilize the weak tie and strong tie property to design different forwarding strategies.
- As a new type of DTNs, *opportunistic mobile social networks* (OMSNs) caught many attention recently. We design different social-aware routing approaches in OMSNs. In this thesis, a hypercube-based multi-path schemes is proposed, which leverage the social feature information of the users to enhance the correctness of the information forwarding decision. We also formal analyze the performance of the hypercube-based social feature routing scheme

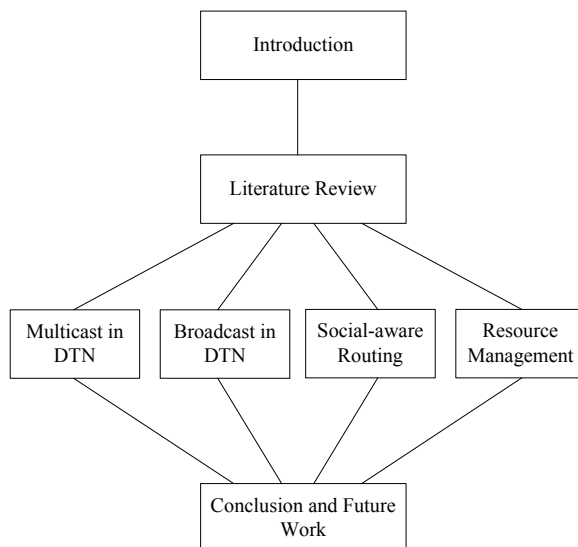


Figure 1.3: Overview of the Dissertation.

- A comprehensive study of a method for resource management is presented. Storage congestion control has been studied extensively in traditional Internet. Due to the challenge environment in DTNs, congestion control problem is different and difficult. We apply the message migration mechanism in this thesis by designing a joint replication-migration-based routing scheme to solve the problem.

### 1.3 Dissertation Overview

The rest of the thesis presents the details of the routing schemes we design for DTN. Our designs follow the basic architecture of DTN and incorporate specific techniques that suit the demands of the applications. We hope this thesis would be proved to be useful for the development of practical DTN routing schemes.

Chapter 2 reviews representative relevant research works. We survey the architecture, DTN multicasting approaches, data dissemination schemes, social-aware routing protocols, and resource management problem in DTNs. Chapter 3 presents our work on DTN multicast schemes. We propose three multicast approaches in DTN environment: delegation forwarding multicast, non-replication multicast, and cloud-based multicast with feedback. Chapter 4 presents the broadcast approaches we designed in DTNs. Chapter 5 studies the hypercube-based social feature multi-path routing scheme. Chapter 6 presents a joint-replication-migration-based mechanism to solve the resource management problem in DTNs. Chapter 7 concludes this thesis and identifies potential future research topics in DTNs.

## Chapter 2

# LITERATURE REVIEW

This chapter surveys existing research works that are closely relevant to the research of this thesis. We start with a short overview of DTN architecture. We then discuss the major topics: routing protocols in DTNs including unicast, multicast, and broadcast. We will also discuss the resource management problem in DTNs. At last, we overview the experimental datasets are used in DTN research.

### 2.1 DTN Architecture

The delay tolerant networking architecture was initially proposed as an approach to make the InterPlanetary Internet (IPN) [3] a viable networking environment. In the ZebraNet project [4], researchers “equipped” Zebras in Kenya with sensor-enhanced collars in order to gather information regarding the animals’ moving and social behaviors. Recently, a considerable amount of research has been carried out with the goal of providing connectivity in developing countries. In the DakNet project [5], it provides connectivity to remote villages in India and Cambodia. Computer-equipped kiosks within remote villages serve villagers with digital applications, requests for national documents, bank services, e-mail etc. Once the villagers’ applications are gathered, they are transferred to the closest city, where Internet connection exists, with public transport vehicles (i.e., buses) equipped with Wi-Fi devices. Vehicular communications have attracted a lot of attention in recent years, based on early studies such as [6]. The ultimate goal of that field of research is to provide (some sort of) connectivity to commuters. In the Huggle project [7], researchers attempt to establish a people-centric approach to DTNs. They gather potential Huggle-user preferences to investigate tradeoffs of reliability and data delivery delay in Pocket-Switched Networks [8] and Huggle applications. we consider the Huggle’s approach to be one of the most interesting approaches to DTN

Table 2.1: Multicast Routing Schemes in DTNs.

Single Node Model	Multiple Copies Model	Single Copy Model
[11], [17], [14]	[18], [24], [19], [25], [9], [21], [15], [22]	[23], [12], [20], [16]

research, since it takes into consideration the application’s and the user’s preferences.

## 2.2 DTN Multicasting

Multicast is an important routing function that supports the distribution of data to a group of users, a service needed for many potential DTNs applications. While multicasting in the Internet and mobile ad hoc networks has been studied extensively, efficient multicasting in DTNs is a considerably different and challenging problem due to the probabilistic nature of contact among nodes.

There also has been some work on multicast routing protocols in DTNs [9–25]. In this chapter, we survey the most recent contributions on DTN multicast routings.

The existing research focuses on three models:

- **Single node** (also called *ferry*) **model**, in which one single node holds all destinations, and delivers them to each destination, as they contact one another, through movement.
- **Multiple copies model**, in which the destination set is replicated at a contact once a certain condition related to the quality of the encountered node is satisfied. Most of them are extensions of unicast routing protocols.
- **Single copy model**, where a single copy of each destination is maintained where destinations can be scattered at different nodes. Each destination is forwarded to an encountered node if it has a higher probability of reaching the corresponding destination.

### 2.2.1 Single Node Multicasting

In the *single node* (also called *ferry*) *multicasting model*, one single node holds all destinations, and delivers them to each destination as they contact one another, through movement. As shown In Fig. 2.1, a ferry moves around, picks up the message from the source node, and drops it off to the destination nodes.

In [11], Zhao, Ammar, and Zegura proposed the basic single node model, together with new semantics for DTN multicasting, which explicitly specify temporal constraints on group membership and message delivery. [11] was the first study of multicasting in DTNs. The authors concluded that

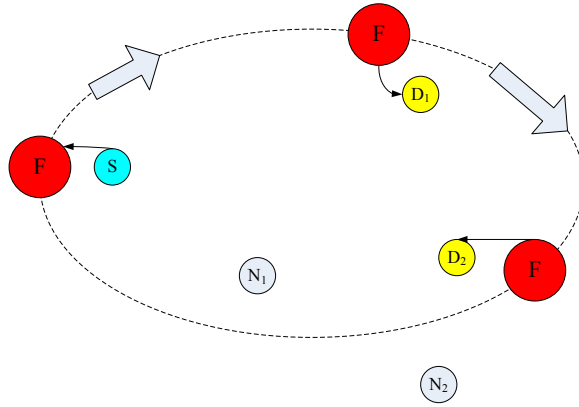


Figure 2.1: An illustration of single node model in DTN multicasting.

even with partial knowledge, multicast routing algorithms can perform efficiently in DTN environments. They also summarized that topology information is more important than group membership information in performing efficient multicasting in DTNs. Yang and Chuah presented a two-stage single node model in [17], where routes to destinations are first identified through a ferry, followed by the message delivery along the discovered routes. In [14], the authors extended the two-hop relay strategy in DTN multicasting. The relay node is responsible for delivering messages directly to each multicast receiver. In this situation, the relay node is the ferry. The authors reported that their proposed multicast routing protocol can achieve the throughput upper bound  $\Theta(n\lambda)$  for the case  $n_s n_d = O(n)$ , and  $\Theta(\frac{n^2\lambda}{n_s n_d})$  for the case  $n_s n_d = \omega(n)$ , where  $n$  is the number of nodes, and the inter-contact time of an arbitrary pair of nodes follows an exponential distribution with rate  $\lambda$ . There are  $n_s$  sources, each of which is associated with  $n_d$  random destinations.

### 2.2.2 Multiple Copies Multicasting

In the multiple copies multicasting model, the destination set is replicated at a contact once a certain condition related to the quality of the encountered node is satisfied. The number of replications, or copies, is hard to control in the DTN environment in the multiple copies model, in which *ticket-based* or *token-based* schemes were proposed. There are different ways to measure the multicast ability of each node. The researchers used the inter-contact time, contact frequency, and contact duration of the node, with the destinations or all of the nodes as the evaluation metric. Recently, social network concepts have been introduced to measure the multicast ability of each node, such as centrality, similarity, and so on.

Epidemic routing [26], using a flooding-based scheme, is also suitable for DTN multicast. However, using epidemic routing will cause high overhead. In [18], Abdulla and Simon studied the effects of different controlled epidemic routing schemes using time-to-live (TTL), message expiration times,

forwarding probabilities, and the number of copies to spread.

The approach discussed above relies on the opportunistic connectivities among nodes for delivery, without making use of additional information such as node location and node velocities. In 2009, Chuah and Yang introduced a context aware multicast routing in [19], which is a node-density based adaptive multicast scheme. Their scheme estimated the local node density and 2-hop neighbor contact probability to enhance the multicast routing scheme. This node-density based adaptive multicast routing scheme can address the challenges of opportunistic link connectivity in DTNs, by using information like node locations and velocities. Hence, this scheme is flexible, and can adapt to different network environments.

The multicast routing schemes discussed above assumed a route discovery process that is similar to the mobile ad hoc network routing approaches. In [25], the authors introduced an encounter-based multicast scheme in DTNs, which was built on top of the PROPHET routing scheme [27]. The authors built a dynamic tree by referring to the contact probability between two nodes, rather than the shortest path route between two nodes.

Network coding is a mechanism in which nodes encode two or more incoming messages, and forward encoded messages instead of forwarding them as they are. In [15], Narmawala and Srivastava designed a network coding based multi-copy routing scheme for DTN multicast. The authors proposed different packet purging schemes to drain messages out of the network, which takes advantage of features of network coding, as to increase buffer efficiency.

### 2.2.3 Single Copy Model

In order to reduce the overhead, the single copy model is introduced in DTN multicast. There is only one copy for each destination, and destinations can be scattered at different nodes. Each destination is forwarded to an encountered node if it has a higher probability of reaching the corresponding destination.

In [12], Gao et al. developed a single copy model where the forwarding metric is based on the social network perspective, which is a social-based approach, based on a notion of “ego-centric betweenness,” in order to optimize multicast performance in DTNs. The authors considered the node selfishness on DTN multicasting in [20]. They investigated how the selfish behaviors of nodes affect the performance of DTN multicast, in two-hop relaying and epidemic relaying schemes. In [16], the authors studied the DTN multicast problem from the graph indexing point of view. They solved the problem of minimizing the remote communication cost for multicast in DTNs.

## 2.3 DTN Broadcast

In DTNs, broadcasting (or data dissemination) is an important routing service that supports the distribution of data to all users in the network. Efficient broadcasting in DTNs is a challenging problem due to the lack of continuous network connectivity.

Goundan, Coe, and Raghavendra discussed a mechanism for energy efficient broadcasting in [28]. This is a  $k$ -neighbor broadcast scheme, where nodes do not broadcast all of the time, but wait for an opportunity to reach multiple nodes with one transmission, thereby reducing the number of transmissions overall. In [29], Karlsson, Lenders, and May proposed a design for an open, receiver-driven broadcasting system that relies on delay tolerant forwarding of data chunks through the mobility of wireless nodes. The system provides public broadcast channels, which can be openly used for both transmission and reception. In [30], the authors study broadcasting of information in a system of moving agents equipped with omni-directional as well as directional antenna. They derive an expression for the mean broadcasting time and study the information dissemination robustness of the system using elements of classical epidemiology and physics.

In [31], Ning et al. proposed a credit-based incentive-aware data dissemination scheme in DTN, which effectively tracks the value of a message, which highly depends on its probability to deliver by an intermediate node. Gao and Cao proposed a user-centric data dissemination in [32]. Their approach was based on a social centrality metric, which considers the social contact patterns and interests of mobile users simultaneously, and thus ensures effective relay selection.

## 2.4 Social-aware Routing

There exists a number of studies that investigate the impact of human mobility and their social relations on the design and performance of the appropriate routing algorithms [8, 33–41]. It can be divided into two categories: one is analysis the people mobility model to enhance the DTN routing performance [35, 36, 40, 41]; another is using the social network concept to design efficient DTN routing [8, 33, 34, 37–39].

Authors in [35], analyze a large number of traces from diverse human-mobility environments and find that the inter-contact time distribution is heavy-tailed. This result implies that routing algorithms for DTN environments have to be tested under different mobility models than the Random Way-Point. In [8], the authors investigate human mobility models in conference environments and find that some nodes are more “popular” than others. That said, contact history-based forwarding algorithms may be benefited from such information. Apparently, popularity will impact energy and storage resources. In a similar study [39], the authors investigate the path explosion phenomenon

and explore its impact on the performance of the forwarding algorithm. Path explosion is the phenomenon according to which, once a path to the destination is formed, the number of subsequent paths grows rapidly. This has a direct impact on the performance of several routing proposals, such as epidemic [26], which appear to perform similarly. Although storage and energy issues are not investigated, results in [39] call for further investigation as for the availability of forwarding paths within a DTN setup.

In [34], authors propose SimBet routing, which exploits the exchange of preestimated “betweenness” centrality metrics and locally determined social “similarity” to the destination node. Betweenness and similarity concepts are from social network. BUBBLE Rap: social-based forwarding algorithm is introduced in [33]. The authors discover that human interaction is heterogeneous both in terms of hubs (popular individuals) and groups or communities. BUBBLE combines the knowledge of community structure with the knowledge of node centrality to make forwarding decisions, which achieves significant improvement of forwarding efficiency.

There is a considerable amount of research going on presently with regard to mobile social networks. The main issue there is that messages should be forwarded to nodes that are interested in the specific content of the message (e.g., latest update on football news, [42–44]). That said, there is no ultimate destination for the message - the message keeps on spreading once it finds nodes interested in its content.

## 2.5 Resource Management

Several solutions have been proposed to handle storage congestion control problems in DTNs. Most of them are based on message dropping policies [45–47]. In [45], Zhang et al. discussed buffer-constrained epidemic routing. They compared some simple buffer management policies, and concluded that *Drop Head* (the message that was first entered into the queue is the first message to be dropped when the buffer is full) outperforms *Drop Tail* (removing the newly received message) in terms of both delivery rate and delay. Lindgren et al. evaluated a more extensive set of combinations of existing buffer management policies for DTNs in [46]. In [47], Krifa et al. provided optimal buffer management policies, based on global knowledge about the network, for DTNs. In [48], Seligman et al. investigated *storage routing* to avoid storage congestion in DTNs. The basic idea is that once a node becomes congested, it forwards some fraction of its stored messages to alternative custodians in order to decongest itself; therefore, the node will be able to serve incoming traffic.

## Chapter 3

# MULTICAST IN DTNS

DTN-based store-carry-forward communication is well-suited to multicast and data sharing services, since bundle storage and forwarding to multiple contacts is an essential feature of DTN bundle routers. The goals for DTN multicast are: achieving a high total delivery rate for messages to interested receivers, achieving a fairly distributed delivery rate for messages from different sources, achieving minimum delivery delays, and optimizing resource utilization.

This chapter first presents the delegation forwarding multicast, in Section 3.1, then a non-replicate multicast tree approach is proposed in Section 3.2. Finally, we apply the community concept in DTN multicasting and present a cloud-based multicast scheme in Section 3.3.

### 3.1 Delegation Forwarding Multicast

In this section, we present a delegation forwarding multicast scheme in DTNs, in which the message holder for each destination will replicate the copy (for that destination) and forward it to an encountered node that has a higher quality than all previous nodes seen so far, with respect to that particular destination. We also present other two schemes: *single copy multicast* and *multiple copy multicast*. In single copy multicast scheme, there is only one copy for all destinations. The message holder will only forward the copy to a node whose quality is higher considering all destinations. In multiple copy multicast scheme, there is one copy for each destination. The message holder for each destination can be different. The message holder for a particular destination will forward the copy to an encountered node which has a higher quality, with respect to the destination. These three multicast schemes are all based on the dynamic multicast trees, as shown in Fig. 3.1. In Fig. 3.1(a), we can see that in the single copy model, there is only one tree branch. In multiple copy and delegation forwarding models, there are multiple copies of the message in the network, hence, there are

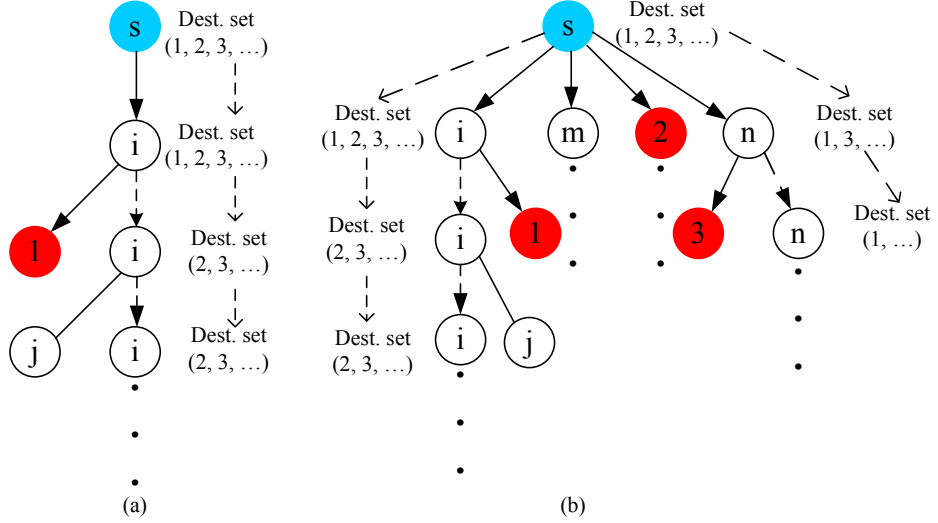


Figure 3.1: Multicast tree: (a) single copy; (b) multiple copy and delegation forwarding.

multiple tree branches to seek the destination nodes in Fig. 3.1(b).

### 3.1.1 Preliminary work

Recently, an approach called *delegation forwarding* (DF) [49] caught significant attention in the research community because of its simplicity and impressive performance. Its main idea is to assign a quality and a level value to each node. We will use the frequency of a node meeting the destinations as the quality value of a node in this report. Initially, the level value of each node is equal to its quality value. During the routing process, a message holder only forwards the message to a node with a higher quality than its own level. In addition, the message holder also raises its own level to the quality of the higher quality node. This means a node will forward a message only if it encounters another node whose quality value is greater than any node met by the message so far.

In DF, with the increase of its level, a message holder’s forwarding chance is expected to be decreased, which means the number of copies duplicated for a message and its total number of forwardings are expected to be decreased. Thus, using DF can reduce the network cost. From analysis in [49], we see that in an  $N$ -node network, delegation forwarding has an expected cost of  $O(\sqrt{N})$  when compared with a naive scheme of forwarding to any higher quality node having an expected cost of  $O(N)$ .

Because DF’s performance is capable of reducing the cost in DTNs, in this report we will extend it into DTNs multicast research to analyze two metrics: (1) *the number of forwardings*: the number of forwardings for a whole multicast process. This can be considered as the cost for the multicast process; (2) *latency*: the average duration between a message’s generation and the arrival time at the last destination. “High performance” means fewer number of forwardings and smaller latency.

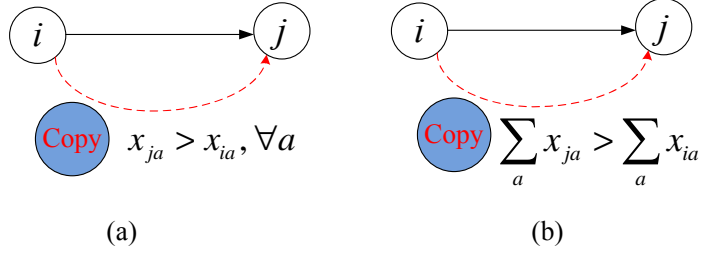


Figure 3.2: Single copy multicast in DTNs.

### 3.1.2 Single copy multicast

The main idea of the single copy multicast model is that the source node will multicast a single copy to  $D$  destinations. Quality value  $x_{ia}$  denotes the frequency node  $i$ , which meets with destination  $a$ , ( $a \in \{1, 2, \dots, D\}$ ). When node  $i$  meets with node  $j$ , if for all destinations  $x_{ja} > x_{ia}$ , then the copy will be forwarded from node  $i$  to node  $j$ . Otherwise, unless node  $j$  is a destination, node  $i$  will not forward the message to node  $j$ . This means the message holder will just forward the copy to a node which has a higher quality for all destinations. Fig. 3.2(a) shows the forwarding decision rule for this algorithm.

We also apply a weak strategy in our simulation. We call it *single copy (sum)*. When node  $i$  meets node  $j$ , they compare the sum of the quality value for all destinations. If  $\sum_{a=1}^D x_{ja} > \sum_{a=1}^D x_{ia}$ , node  $i$  will forward the copy to node  $j$ . When the copy is forwarded to one of the destinations, this destination will be deleted from the destination set. Fig. 3.2(b) gives the simple forwarding algorithm, as we mentioned above.

### 3.1.3 Multiple copy multicast

Although single copy multicast has a smaller number of forwardings, it has a much longer delay. We believe that another algorithm based on the multiple copy multicast will reduce the latency. Compared with the single copy model, there are  $D$  copies (same as the destinations number) in the source node in the multiple copy multicast model. The main idea is, after meeting with node  $j$ , which has higher quality  $x_{ja}$  for destination  $a$ , node  $i$  will forward a copy to node  $j$  and ‘ask’ node  $j$  to forward this copy to destination  $a$ . If node  $j$  is a destination, node  $i$  will forward a copy to this destination node without hesitation. The destination node can also be a relay for other destinations. This forwarding algorithm is shown in Fig. 3.3. As shown in Fig. 3.3, copy  $a$  is forwarded from node  $i$  to node  $j$ , because node  $j$  has a higher probability to meet with destination  $a$  ( $x_{ja} > x_{ia}$ ).

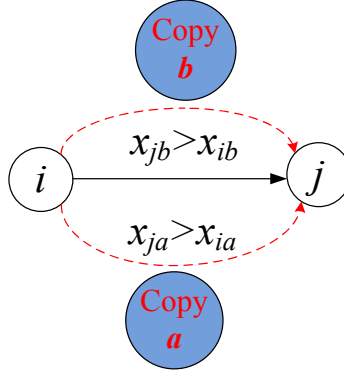


Figure 3.3: Multiple copy multicast in DTNs.

### 3.1.4 Delegation forwarding multicast

For any node  $i$ , the forwarding problem is a simple question: “upon contact with node  $j$ , should node  $i$  forward the message to node  $j$ ?” For many algorithms, the answer to the forwarding question is “forward the message if node  $j$ ’s quality is higher than node  $i$ .” However, the cost of this approach can still be quite high. To reduce the cost, we seek to forward the message only to the highest quality nodes that have previously met. Conceptually, we would like to forward less, and give the message to the nodes which are the best candidates for eventual delivery to the destinations. Thus, the forwarding question becomes “is node  $j$  among the very highest quality nodes?”

The delegation forwarding multicast algorithm’s main idea is to assign a quality value (which is static) and a level value (which is dynamic) for each node to each destination. Initially, the level value  $\tau_{ia}$  for destination  $a$  of each node is equal to its quality value  $x_{ia}$  for destination  $a$ . During the routing process, a message holder  $i$  compares the quality  $x_{ja}$  of the node  $j$  it meets with its level value  $\tau_{ia}$ . It only forwards the message to a node with a higher quality value than its own level value and ‘asks’ this node to help forward the message to destination  $a$ . This approach does not need global knowledge. Each node decides whether it should or should not forward the message by itself. This is suitable for a distributed environment, such as DTNs. In addition, the message holder also raises its own level to the higher quality. If node  $j$  is one of the destinations, node  $i$  will forward a message to it and also use the strategy to determine whether node  $j$  is a good relay to forward the message.

The DF algorithm is shown in Fig. 3.4 and Algorithm 1. The main difference from the previous two models is, in DF, the message will be replicated and after the forwarding process, initial message holder  $i$  and its relay node  $j$  will both have the copy of the message, therefore, there will be multiple nodes to seek the destinations. This means DF can reduce the cost and delay dramatically. The analysis and simulation results support our expectations.

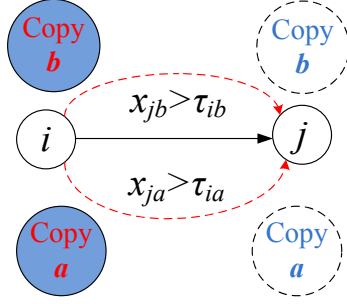


Figure 3.4: Delegation forwarding multicast in DTNs.

### 3.1.5 Analysis

In this section, we offer mathematical models according to our algorithms proposed in previous sections. We compare the number of forwardings and latency of these three multicast models.

#### Number of forwardings

In the following, we will consider a single message and calculate how many times it is forwarded before reaching all destinations. We will first offer the results of the delegation forwarding multicast model. The results of the other two models can be derived from this model.

#### Number of forwardings with the delegation forwarding multicast model

The cost of DF in DTN unicast is given in [49]. To make the report self-contained, we include some of the ideas and proof methods in [49]. For any node  $i$  maintaining a quality metric for destination  $a$ :  $x_{ia}$  (which lies between  $(0, 1]$ ) and a level value  $\tau_{ia}$ , we focus on the gap  $g_{ia} = 1 - \tau_{ia}$  between the current level and 1. The node that generates the message has a level value initially equal to its quality value, *i.e.*,  $\tau_{ia} = x_{ia}$ . We denote the initial gap  $g_a = 1 - x_a$ .

**Theorem 1** *Given the level value  $\tau_{ia}$ , the expected number of forwardings in the delegation forwarding multicast model is*

$$E[F_{delegation}] \lesssim \frac{1}{2}\sqrt{N} + \frac{1}{3}D \cdot \sqrt{N},$$

where  $N$  is the number of nodes, and  $D$  is the number of destinations.

**Proof.** Suppose a node updates its gap value  $n$  times. The node's current gap is denoted as the random variable  $G_n$ . Since nodes meet according to rates that are independent of node quality, the node is equally likely to meet a node with any particular quality value. The next update of the gap of the nodes then occurs as soon as it meets a node with a higher quality value than  $G_n$ , and all values above this level are equally likely.

Hence, we can write

$$G_{n+1} = G_n \cdot U, \tag{3.1}$$

---

**Algorithm 1** Delegation Forwarding

---

- 1: There are  $N$  nodes in the network.
  - 2: There are  $D$  destinations need to be multicast.
  - 3: Node  $n$  has quality  $x_{nd}$  and level  $\tau_{nd}$  for destination  $d$ .
  - 4: INITIALIZE  $\forall n, d : \tau_{nd} \leftarrow x_{nd}$ .
  - 5: On contact between node  $i$ , which is the message holder for destination  $a$  and node  $j$ :
  - 6: **if**  $x_{ja} > \tau_{ia}$  **then**
  - 7:      $\tau_{ia} \leftarrow x_{ja}$
  - 8:     **if** node  $j$  does not have the message for destination  $a$  **then**
  - 9:         replicate a message to node  $j$ .
  - 10:     **end if**
  - 11: **else**
  - 12:     **if** node  $j$  is the destination  $a$  **then**
  - 13:         replicate a message to node  $j$ .
  - 14:     **end if**
  - 15: **end if**
- 

where  $U$  is independent of  $G_n$  and follows a uniform distribution on  $(0, 1]$ . According to [49], in our multicast scheme, we can find:

$$E[G_{n+1}|G_n] = \frac{G_n}{2}, \text{ hence, } E[G_n] = \frac{\sum_{a=1}^D g_a}{2^n}.$$

Moreover, from Eq. (3.1), we see that  $G_n$  approximately follows a lognormal distribution, with median  $(\sum_{a=1}^D g_a)/e^n$ . Hence, the distribution is highly skewed with most of the probability mass below the mean. So with a large probability, we have  $G_n \leq (\sum_{a=1}^D g_a)/2^n$ .

As in [49], the replication process can be described by a dynamic binary tree  $T$ , which contains all the nodes that have a copy of the message. We define the set  $B_a = \{i | x_{ia} \geq 1 - \frac{g_a}{\sqrt{N}}\}$ ,  $a \in \{1, 2, \dots, D\}$ , which we call the *target set*. We will also identify a subtree of the tree  $T$  in which children are excluded for nodes having a threshold above  $1 - \frac{g_a}{\sqrt{N}}$ . All nodes in the subtree have a gap less than  $\frac{g_a}{\sqrt{N}}$ . This subtree is called the *target-stopped tree*.

According to [49], the essential observation is the following: if  $n$  is close to  $\log_2(\sqrt{N})$ , then except for a small probability, a node at generation  $n$  in the tree has a gap of at most  $g_a/2^n \leq g_a/\sqrt{N}$ . Hence, we can safely assume that the target-stopped tree has a depth of at most  $n$ . Note that the total number of nodes appearing at generations  $0, 1, \dots, n-1$  is at most  $2^n = \sqrt{N}$ .

In [49], Erramilli et al. offer the number of forwardings in the delegation forwarding unicast model. Hence, in delegation forwarding multicast model of  $D$  destinations, the total size of this tree is at most:

$$C_{delegation} \lesssim \sqrt{N} + |\sum_{a=1}^D B_a| = (1 + \sqrt{\sum_{a=1}^D g_a}) \cdot \sqrt{N}.$$

Then, we obtain the total number of forwardings:

$$F_{delegation} \lesssim \frac{1}{2} \left(1 + \sqrt{\sum_{a=1}^D g_a}\right) \cdot \sqrt{N},$$

Since, we know  $\sqrt{\sum_{a=1}^D g_a} \leq \sum_{a=1}^D \sqrt{g_a}$ .

Hence,

$$\int_0^1 \sqrt{\sum_{a=1}^D g_a} dg_a \leq \sum_{a=1}^D \int_0^1 \sqrt{g_a} dg_a = \frac{2}{3} D.$$

Therefore,

$$\begin{aligned} E[F_{delegation}] &= \int_0^1 \frac{1}{2} \left(1 + \sqrt{\sum_{a=1}^D g_a}\right) \cdot \sqrt{N} dg_a \\ &\lesssim \frac{1}{2} \sqrt{N} + \frac{1}{3} D \cdot \sqrt{N}. \end{aligned} \tag{3.2}$$

■

### Number of forwardings with the single copy multicast model

In this part, we analyze the two single copy multicast models: *single copy* and *single copy (sum)*.

**Theorem 2** *In the single copy model, the expected number of forwardings is:*

$$E[F_{single}] \lesssim D \cdot \log_D N,$$

**Proof.** In the single copy (all) model, we need to compare all of the destinations. When all of the quality values are larger than the current one, we will forward the copy. Thus, the probability of forwarding is equal to  $\frac{1}{D}$ , where  $D$  is the number of destinations.

Thus, in the single copy (all) model, the expectation of  $G_n$  becomes:

$$E[G_n] = \frac{\sum_{a=1}^D g_a}{D^n},$$

where  $D$  is the number of destinations.

Using the same methods, we obtain the number of forwardings:

$$F_{single} \lesssim \log_D \left( N \cdot \sum_{a=1}^D g_a \right),$$

hence,

$$E[F_{single}] \lesssim D \cdot \log_D N. \tag{3.3}$$

■

**Theorem 3** *In the single copy (sum) model, the expected number of forwarding times is the same as the delegation forwarding multicast model:*

$$E[F_{single(sum)}]dg \lesssim \frac{1}{2}\sqrt{N} + \frac{1}{3}D \cdot \sqrt{N}.$$

**Proof.** In the single copy (sum) model, when node  $i$  meets node  $j$ , the probability of the *sum* value on node  $i$ , which is larger than node  $j$ , is equal to 0.5. Thus, it is the same situation of delegation forwarding. Hence, the probability of the forwarding decisions is also 0.5.

$$E[F_{single(sum)}] = \int_0^1 F_{delegation}dg \lesssim \frac{1}{2}\sqrt{N} + \frac{1}{3}D \cdot \sqrt{N}. \quad (3.4)$$

■

### Number of forwardings with the multiple copy multicast model

Here, we will discuss the multiple copy multicast model.

**Theorem 4** *In the multiple copy model, the expected number of forwardings is:*

$$E[F_{multiple}] \lesssim D \cdot \log_2 N.$$

**Proof.** Since the probability for the node to forward the copy is  $\frac{1}{2}$ , according to Eq. 3.3, we have:

$$F_{multiple} \lesssim \log_2(N \cdot \sum_{a=1}^D g_a),$$

hence,

$$E[F_{multiple}] \lesssim D \cdot \log_2 N. \quad (3.5)$$

■

### Number of forwardings using flooding

In this part, we will discuss the number of forwardings using flooding.

**Theorem 5** *When using flooding routing protocol, the expected number of forwardings is:*

$$E[F_{flooding}] \approx \frac{D \cdot N}{2}.$$

**Proof.** From [49], we know that the number of forwardings of non-delegation forwarding in the unicast method is:

$$C_{no-delegation}(g) = gN,$$

where  $g$  is the initial gap value and  $N$  is the number of nodes. Thus, in the multicast forwarding, we have:

$$F_{flooding} \approx N \cdot \sum_{a=1}^D g_a,$$

hence,

$$E[F_{flooding}] \approx \frac{D \cdot N}{2}. \quad (3.6)$$

■

We find that our methods all have a smaller number of forwardings compared with flooding.

### Latency

We assume the contact time between one node to its next relay is  $t$ . The total time for multicast in these three models is:

$$T = \sum_{i=1}^D C_i \cdot t, \quad (3.7)$$

where  $C_i$  denotes the contact times between two destinations  $d_{i-1}$  and  $d_i$ .

1) In the single copy model, the probability of meeting with a higher quality node is  $\frac{1}{D^n}$

$$T_{single} = t \cdot \sum_{i=1}^D D^i = t \cdot N \cdot \frac{D \cdot (D^D - 1)}{D - 1}. \quad (3.8)$$

2) In the multiple copy model, the probability of meeting with a higher quality node is  $\frac{1}{2^n}$

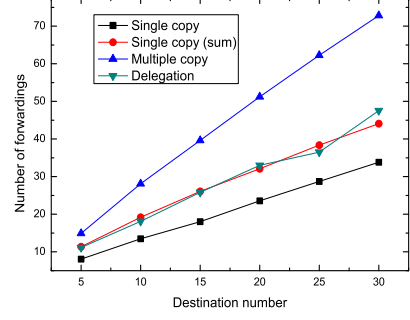
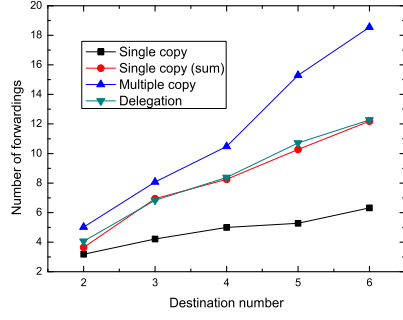
$$T_{multiple} = t \cdot \sum_{i=1}^D 2^i = 2t \cdot N \cdot (2^D - 1). \quad (3.9)$$

3) In the delegation forwarding model, we need to calculate the maximum height of the target-stopped tree, as mentioned in the above. There are  $n = \log_2(N \cdot \sum_{a=1}^D g_a)$  generations to finish multicasting the copies to all destinations.

In the worst case,  $g_a = 1$ , ( $a \in \{1, 2, \dots, D\}$ ), then

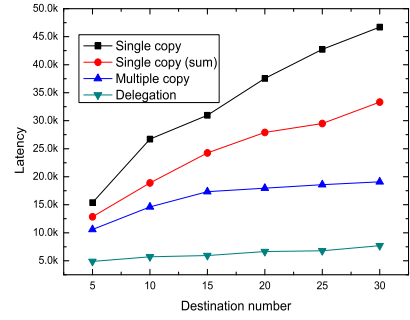
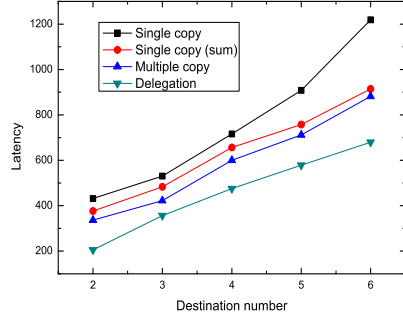
$$T_{delegation} = t \cdot \sum_{i=1}^{\log_2 DN} 2^i = 2t \cdot (DN - 1). \quad (3.10)$$

We can clearly see that  $T_{delegation} < T_{multiple} < T_{single}$ . Delegation forwarding has the best



(a) Number of forwardings in 20-node trace (b) Number of forwardings in 100-node trace

Figure 3.5: Comparison of the number of forwardings in synthetic traces.



(a) Latency in 20-node trace

(b) Latency in 100-node trace

Figure 3.6: Comparison of latency in synthetic traces.

performance in DTNs multicast latency.

### 3.1.6 Simulation

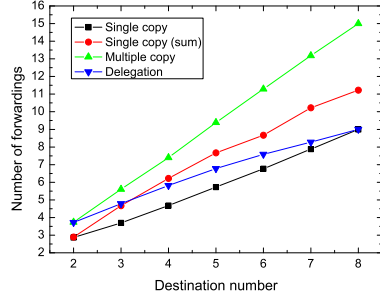
In the previous parts, we analyzed the single copy, multiple copy, and delegation forwarding multicast algorithms in DTNs multicast, and have shown that they can dramatically reduce the number of forwardings. Here, we evaluate the performance of the multicast routing algorithms presented in this thesis. We use the Intel and Cambridge traces [50] in our simulation. These data sets consist of contact traces between short-range Bluetooth enabled devices carried by individuals.

The following metrics are calculated in our simulation. Each simulation is repeated 1,000 times.

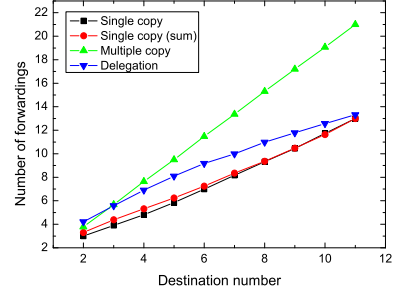
1. *Average cost*: the average number of forwardings for all destinations to receive the message.
2. *Actual delay*: the average latency for all the delivered destinations to receive the message.

### Results

First, we compared the performance of these forwarding algorithms in the synthetic traces, as shown in Figs. 3.5 and 3.6. In both the 20-node trace and the 100-node trace, we can see that the single copy model with strong strategy has the fewest number of forwardings. The delegation forwarding

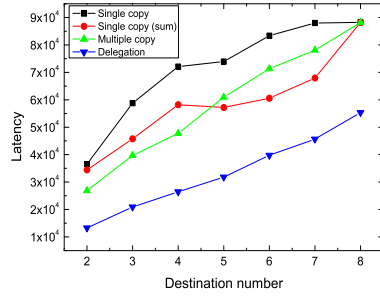


(a) Number of forwardings in Intel

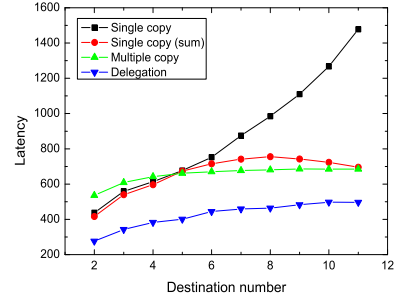


(b) Number of forwardings in Cambridge

Figure 3.7: Comparison of the number of forwardings in real traces.



(a) Latency in Intel



(b) Latency in Cambridge

Figure 3.8: Comparison of latency in real traces.

has a similar number of forwardings as the single copy (sum) model, and both better than the multiple copy model. From Fig. 3.5(a), we can see that the single copy model has 50% fewer number of forwardings than the delegation forwarding model. Delegation forwarding has about 30% fewer number of forwardings compared with the multiple copy model. In the 100-node trace, the results are similar. At the same time, delegation forwarding has much shorter latency than other models, while the single copy model has the longest latency among these protocols, in Fig. 3.6.

Then, we compared the number of forwardings among these three forwarding algorithms in real traces, as shown in Fig. 3.7. We can see that the single copy model using the strong strategy has the fewest number of forwardings. The delegation forwarding has a smaller number of forwardings than the multiple copy model in both Intel and Cambridge traces. In the Intel trace in Fig. 3.7(a), it needs about 1.2 times the number of forwardings to arrive at a destination using the strong strategy single copy model while the weak strategy needs 1.48 times. The multiple copy model and delegation forwarding model need 1.9 and 1.4 times, respectively. In the Cambridge trace, the number of forwardings per destination in the strong strategy and weak strategy single copy model is 1.2 and 1.3, respectively. Also, they are 1.9 and 1.5 times for the multiple copy and delegation forwarding models, respectively, as shown in Fig. 3.7(b). These results are the same as what we analyzed in Section 3.1.5.

The results of the latency comparison are shown in Fig. 3.8. Delegation forwarding has the least amount of latency, which has a 48% time reduction over the single copy model with the strong strategy. The single copy model has the longest latency among these algorithms. The delegation forwarding model has the least amount of latency, both in the Intel and Cambridge traces.

### Summary of simulation

We first use these three forwarding methods in DTNs multicast. Simulation results confirmed that they have their own benefit used as the forwarding algorithm in DTNs multicast. We know that the single copy model has the longest latency and fewest number of forwardings, both in the simulation and analytical results. The multiple copy model reduces the latency from the single copy model, because it has more of a chance to meet with other higher priority nodes. Delegation forwarding uses many branches to forward the copies, so it has the shortest latency among these models, which has been proven by analytical results and simulation results. Although the delegation forwarding model has a slightly increased number of forwardings than the single copy model, it reduces the cost from the multiple copy model significantly. These forwarding algorithms are all better than flooding when comparing the number of forwardings.

### 3.1.7 Conclusion

In this section, we studied the problem of multicasting in DTNs. We focused on the multicast forwarding algorithms. We discussed the single copy, multiple copy, and delegation forwarding algorithms in DTNs multicast. Then, we analyzed these three models mathematically. We finally turn to studying the performance of these three forwarding algorithms not only in synthetic traces, but also in real mobility traces. Trace driven simulation results have shown that using delegation forwarding has the smallest latency, while the single copy model has the fewest number of forwardings. In the future, we go forward with the probability delegation forwarding (PDF) [51] and threshold-based probability delegation forwarding (TPDF) schemes to further reduce the number of forwardings. We believe that this report presents the first step in exploiting forwarding decision rules in DTNs multicast. Future research can benefit from our results by developing specific applications based on the provided multicast forwarding architecture in DTNs.

## 3.2 Non-Replication Multicast

Non-replication multicast scheme is based on the single copy model with the objective to reach destinations quickly while minimizing the number of forwardings. We observe that pure priority-based-

split may produce an excessive number of forwardings (e.g., for a succession of small improvements). We propose to use the node's *active level* together with the *contact rate level* to determine when and how to split a destination set during a contact. The notion of the active level is based on the observation that an active node has a better chance of contacting a higher priority node later to improve its delivery time. More specifically, we have the following two notions:

- *Contact rate level* with respect to a destination: a priori knowledge or estimation of the number of contacts with the destination in a given period.
- *Active level* of a node: a priori knowledge or estimation of the number of total contacts in a given period.

In this report, we propose a *compare-split* scheme at each contact during the construction of a dynamic multicast tree. The first step is the *compare* part. When node  $a$ , with a destination subset, has a contact with node  $b$  without any destination subset, we set the condition for splitting as follows: a split occurs when the sum of the contact rate levels for all destinations associated with  $b$  is higher than the one associated with  $a$ . The second step is the *split* part. We propose a *ratio-based-split (RS)*, which splits the destination subset based on the active levels of two encountered nodes. We then present an *optimal split algorithm*, which partitions the destination subset based on the calculated ratio such that the combined sum of the contact rate levels at nodes  $a$  and  $b$  are maximized.

When there is only one destination in the message holder's destination set, we use two schemes to forward the message to this destination: (1) *wait*: wait until meeting the destination; (2) *focus*: forward the message to a higher contact rate level node until arriving at the destination.

### 3.2.1 System Model

Assume that there are  $N$  nodes in the whole network. The destination set of a multicast is represented as  $D = \{1, 2, \dots, n\}$ . Each node  $a$  is associated with a contact rate vector  $(f_1^a, f_2^a, \dots, f_n^a)$ , where  $f_i^a$  indicates the frequency that node  $a$  meets destination  $i$  in a given period  $T$ .  $f_i^a$  is also called *contact rate level* for destination  $i$  in period  $T$ . We use  $(c_1^a, c_2^a, \dots, c_n^a)$  to indicate the number of times that node  $a$  meets destination  $i$  in a given period  $T$ . Hence, the contact rate level  $f_i^a$  can be presented as follows:

$$f_i^a = \frac{c_i^a}{T}. \quad (3.11)$$

In order to involve the effect from recent information, we also define  $T'$ , which is considered as

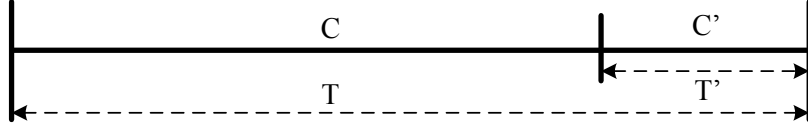


Figure 3.9: An illustration of the system model.

a recent period.  $(c_1^{a'}, c_2^{a'}, \dots, c_n^{a'})$  indicates the frequency that node  $a$  meets destination  $i$  in a given period  $T'$ . Hence, we define the contact rate level  $f_i^a$  as follows:

$$f_i^a = w \cdot \frac{c_i^{a'}}{T'} + (1 - w) \cdot \frac{c_i^a - c_i^{a'}}{T - T'} \quad (3.12)$$

where  $w$  is the weight of the recent information for the contact rate level.

The *active level* of node  $a$ :  $A_a$  is the number of total contacts per time unit  $T$  that node  $a$  meets with all other nodes in the network.

$$A_a = \sum_{i=1}^N f_i^a. \quad (3.13)$$

Fig. 3.9 illustrates the system models considering the recent information.

### 3.2.2 Challenges and Main Ideas

Contact rate level indicates the contact frequency of reaching a particular destination without further forwarding, while active level indicates the likelihood of contacting other nodes to enhance the contact rate level through forwarding. The challenges lie in the balancing of these two factors when two nodes meet. In our single-copy multicast, the key is to decide when and how a split should occur in constructing a multicast tree.

In this thesis, we propose a *compare-split* scheme at each contact during the construction of a dynamic multicast tree. The first step is the *compare* part, which determines when a split should occur. When node  $a$ , with a destination subset, has a contact with node  $b$ , without any destination subset, we set the condition for splitting as follows: a split occurs when the sum of the contact rate levels for all destinations associated with  $b$  is higher than the one associated with  $a$ . The second step is the *split* part, which decides how a split should be done. We propose a *ratio-based-split (RS)*, which splits the destination subset based on active levels of two encountered nodes. We then present an *optimal split algorithm*, which splits the destination subset based on the calculated ratio such that the combined sum of contact rate levels at nodes  $a$  and  $b$  are maximized.

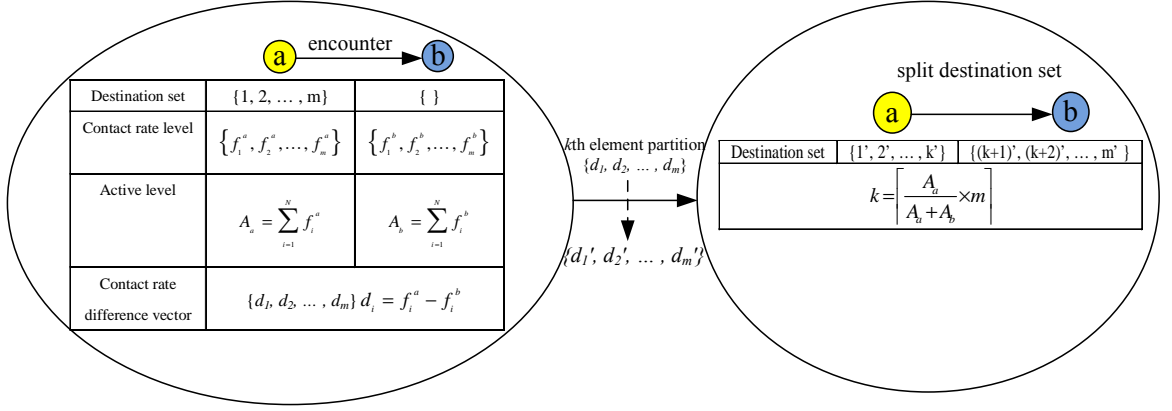


Figure 3.10: An illustration of ratio-based-split.

### 3.2.3 Compare-split

In this part, we propose a *compare-split* scheme at each contact during the construction of a dynamic multicast tree. Here, we will present the two steps of this method and give an example to explain the whole process. The first step is “compare”, which determines whether a split should occur. The second step is “split”, which decides how a split should be done.

#### Compare

The first step for our non-replication multicasting scheme is *compare*. When node  $a$ , with a subset of destinations  $D' \subseteq D$  (as shown in Fig. 3.10,  $m$  is the size of a subset  $D'$  of destination set  $D$ ,  $n$  is the size of destination set  $D$ ), has a contact with a new node  $b$ , without any destination subset, node  $a$  will first send  $D'$  information to node  $b$  and nodes  $a$ , and  $b$  exchange their contact rate vectors,  $(f_1^a, f_2^a, \dots, f_m^a)$  and  $(f_1^b, f_2^b, \dots, f_m^b)$ , upon their contact.  $m$  is the size of the subset  $D'$  of destination set  $D$ . After comparing these two nodes' sum of the contact rate levels for all destinations, if  $\sum_{i=1}^m f_i^b > \sum_{i=1}^m f_i^a$ , then go to the next *split* step.

Note that two rounds of exchanges are used. One round can be saved by exchanging  $(f_1^a, f_2^a, \dots, f_m^a)$  and  $(f_1^b, f_2^b, \dots, f_m^b)$ .  $(f_1^a, f_2^a, \dots, f_m^a)$  and  $(f_1^b, f_2^b, \dots, f_m^b)$  can then be extracted locally.

#### Split

The second step is to *split* the destination set. Suppose that  $d_i = f_i^a - f_i^b$  is called the *contact rate difference* between nodes  $a$  and  $b$  for destination  $i$ . The active levels  $A_a$  can be denoted by the number of total contacts that node  $a$  meets with all other nodes.

$$A_a = \sum_{i=1}^N f_i^a \quad (3.14)$$

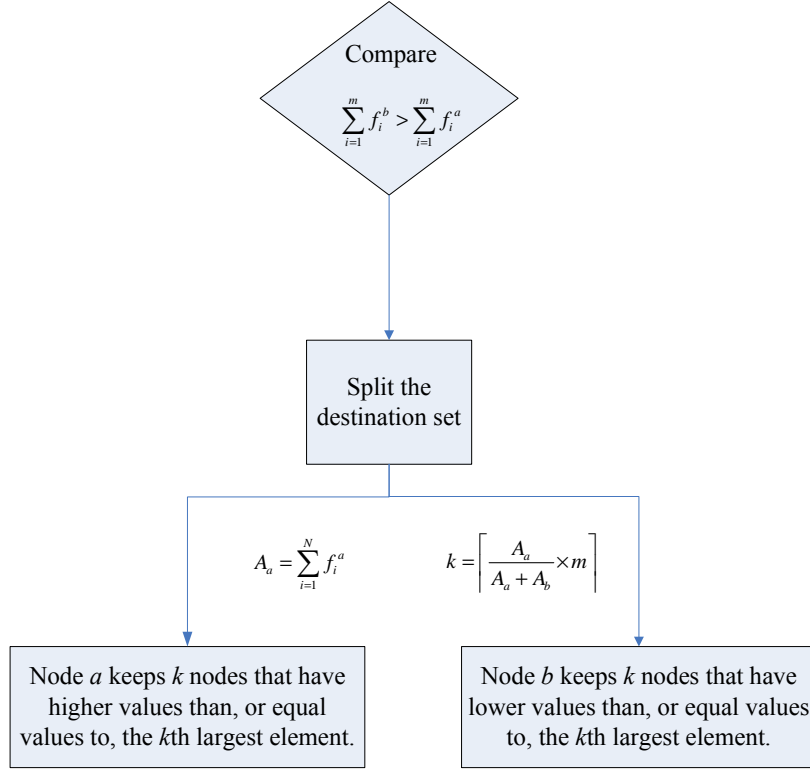


Figure 3.11: An illustration of *compare-split*.

The destination set splitting is based on the ratio of two encountering nodes' active levels. The ratio  $k$  can be denoted as:

$$k = \left\lfloor \frac{A_a}{A_a + A_b} \times m \right\rfloor \quad (3.15)$$

1. Both  $a$  and  $b$  generate the contact rate difference vector  $(d_1, d_2, \dots, d_m)$ . Find the  $k$ th largest element in  $O(m)$  operations using a general *selection algorithm* [52].
2. Node  $a$  keeps  $k$  nodes that have higher values than, or equal values to, the  $k$ th largest element. In the case of a tie, when two contact rate differences are equal, the node ID is used to break the tie.
3. Node  $b$  keeps  $m - k$  nodes that have lower values than, or equal values to, the  $k$ th largest element.

In step (1), the optimal linear solution is used to find the  $k$ th largest element. The whole split process is shown in Fig. 3.10.

destination	1	2	3	4	5	Active level
a	5	2	13	8	15	100
b	3	6	10	11	14	90
d	2	-4	3	-3	1	

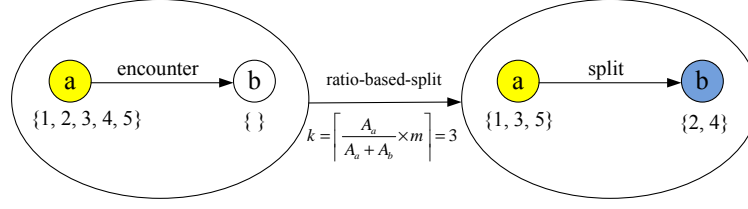


Figure 3.12: An example of *ratio-based-split*.

### Example

Fig. 3.11 illustrates the whole process of our proposed compare-split method. Next we can use Fig. 3.12 as an example. Node  $a$ , with a subset of destinations  $D' = \{1, 2, 3, 4, 5\}$ , makes contact with node  $b$ , without any destination subset. First, node  $a$  sends  $D'$  to node  $b$ , and they exchange their contact rate vectors:  $(f_1^a, f_2^a, \dots, f_5^a) = (5, 2, 13, 8, 15)$  and  $(f_1^b, f_2^b, \dots, f_5^b) = (3, 6, 10, 11, 14)$ . After the calculations, we have  $\sum_{i=1}^5 f_i^b = 44$  and  $\sum_{i=1}^5 f_i^a = 43$ . Hence, the sum of the contact rate levels for all destinations associated with  $b$  is higher than the one associated with  $a$ . Then, we go to the second step. The active levels of node  $a$  and  $b$  are 100 and 90, respectively.

We first calculate the *contact rate difference vector*:

$$(d_1, d_2, \dots, d_5) = (2, -4, 3, -3, 1)$$

and *ratio*:  $k = \lfloor \frac{A_a}{A_a + A_b} \times m \rfloor = 3$ .

Then, we use the selection algorithm to find the *third* largest number in the contact rate difference vector, which is 1.

After splitting the destination set, node  $a$  keeps 3 destinations:  $\{1, 3, 5\}$ , and destinations 2 and 4 will be assigned to node  $b$ . The *combined contact rate* of node  $a$  and  $b$  is  $f_1^a + f_3^a + f_5^a + f_2^b + f_4^b = 50$ , which is larger than  $\sum_{i=1}^5 f_i^a = 43$ . This means that using the compare-split algorithm can increase the contact frequency of meeting with the destinations.

In contrast, the usual greedy way of the splitting process is as follows: (1) possible split 1: node  $a$  will keep the 3 largest contact rate level destinations and assign all other destinations to node  $b$ . In this example, node  $a$  will keep destinations  $\{3, 4, 5\}$  and assign destinations 1 and 2 to node  $b$ . After this process, the combined contact rate of nodes  $a$  and  $b$  is  $f_3^a + f_4^a + f_5^a + f_1^b + f_2^b = 45$ , which is smaller than using the compare-split algorithm; (2) possible split 2: node  $b$  will get the 2 largest

Table 3.1: Different split methods.

<i>Split</i>
ratio-based-split (RS)
random-binary-split (RBS)
median-binary-split (MBS)
priority-based-split (PS)

contact rate destinations, and node  $a$  keeps the rest. Hence, after splitting, node  $a$  keeps destinations  $\{1, 2, 3\}$ , and node  $b$  keeps destinations 4 and 5. After this process, the combined contact rate of nodes  $a$  and  $b$  is  $f_1^a + f_2^a + f_3^a + f_4^b + f_5^b = 45$ , which is also smaller than the result we get from the compare-split algorithm.

### 3.2.4 Implementation & Extensions

There are many other methods that can be implemented in the *compare-split* rule. First, we will explain some conditions in the *compare* phase. Then, we provide three other schemes when splitting the destination set: *random-binary-split (RBS)*, *median-binary-split (MBS)*, and *priority-based-split (PS)*. Finally, we will present two methods: *wait* and *focus* [53], [54], when there is only one destination in the destination subset.

#### Compare

In the previous section, we use the threshold-based condition (when node  $b$  has a higher sum of the contact rate levels for all destinations than node  $a$ , a split will occur) in the *compare* step. Also, we don't have to use any conditions in the first step. We will compare these two methods in our simulation.

Another method is: if node  $b$  already has a destination subset, node  $a$  and node  $b$  will combine their destination sets, then split. It will increase the number of forwardings. We will also compare this method with our scheme in our simulation.

#### Split

In the split step, we also have many other schemes: *binary-split (BS)* (*random-binary-split* and *median-binary-split*) and *priority-based-split*.

**Binary-split (BS):** In binary-split, we will not consider *active level*. The destination split will be *equal partition*. The BS process is shown in Fig. 3.13: nodes  $\{a, b, c, d, e\}$  are relay nodes, and nodes  $\{1, 2, 3\}$  are destination nodes. When one node meets a destination, it will first assign this destination to it and then use the binary-split.

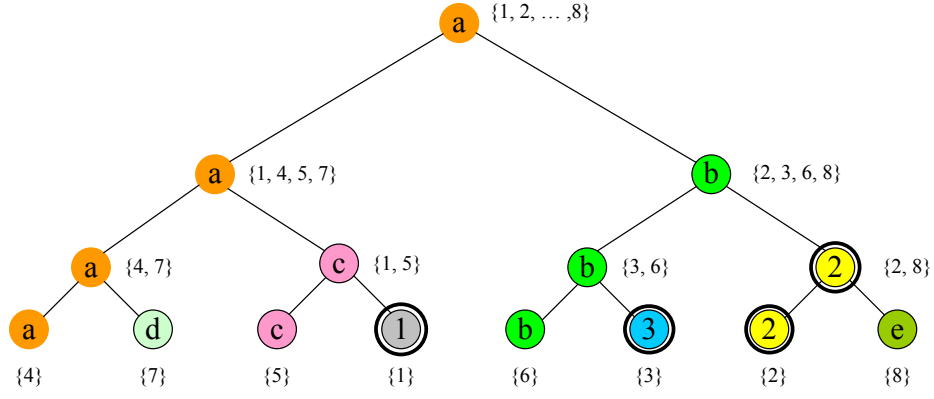


Figure 3.13: A sample of binary-split.

- *random-binary-split (RBS)*: after meeting with node  $b$ , node  $a$  will give half of the destination subset  $D'$  to  $b$  randomly. This means node  $a$  keeps  $\lfloor m/2 \rfloor$  nodes, and node  $b$  keeps  $\lfloor m/2 \rfloor$  nodes.
- *median-binary-split (MBS)*: in RBS, message holder  $a$  partitions the destinations randomly. It may assign a destination to a node with a small contact rate level to this particular destination. Hence, the multicast process will have a large latency. We use another equal partition which is based on contact rate difference. We use the *median of medians algorithm* [52]: a linear solution to find the median of the contact rate difference vector. Then, node  $a$  keeps  $\lfloor m/2 \rfloor$  nodes that have higher values than, or equal values to, the median, and node  $b$  keeps  $\lfloor m/2 \rfloor$  nodes that have lower values than, or equal values to, the median.

MBS can be viewed as a special case of RS when the active levels of two encounter nodes are approximately the same.

**Priority-based-split (PS)**: Another solution is for node  $a$  to keep the destinations with their contact rate difference values higher than 0 and to assign all other destinations to node  $b$ . This means that only the destinations with higher contact rate levels in node  $b$  than in node  $a$  will be assigned to node  $b$ .

Priority-based-split is shown in Fig. 3.14. Initially, node  $a$  takes 8 destinations  $\{1, 2, \dots, 8\}$ . In the *split* phase, the copy arrives at destinations  $\{1, 2, 3\}$  and destinations are assigned to nodes  $\{b, c, d, e, f, g\}$ .

### 3.2.5 Wait and Focus

When there is only one destination that is carried by node  $a$ , we also have two strategies for forwarding decisions, as in [53] and [54]:

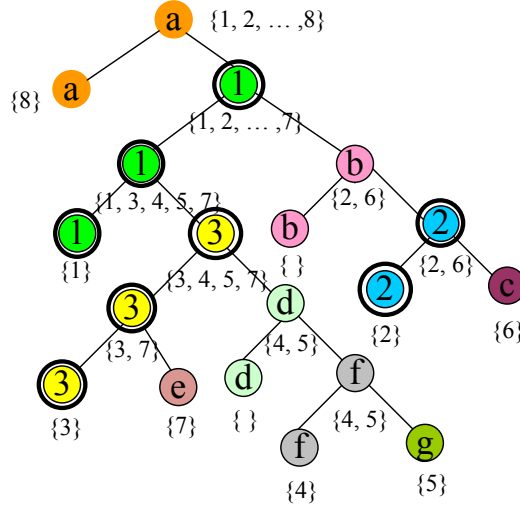


Figure 3.14: A sample of priority-based-split.

- *wait*: Node  $a$  will keep this destination until it meets the particular destination.
- *focus*: Node  $a$  will assign this destination only to a node which has a high contact rate value for this destination.

### 3.2.6 Analysis

In this section, we will explain the optimal split process at each branch of the multicast tree. Then, we analyze the benefit considering contact rate level and active level at the same time. Finally, we compare the difference among single node, single copy, and multiple copies models.

#### Optimal Split Algorithm

Our major goal in using the non-replication multicasting scheme in DTNs is to ensure that the delivery of multicast information is done over different paths. Each path has a relatively high contact frequency of reaching the corresponding destination subset quickly. Then, multiple holders for destination nodes can search for destinations in parallel. These solutions can reduce the multicast cost. The number of forwardings is a major metric to measure the cost of the multicasting process. Compare-split can also reduce the latency in DTN multicasting.

Suppose that  $D_a$  is the destination subset kept in node  $a$  and that  $D_b$  is the destination subset assigned to  $b$ , we would like to maximize the *combined contact rate* of  $a$  and  $b$  as follows:

$$\max\left\{\sum_{i \in D_a} a_i + \sum_{j \in D_b} b_j\right\}. \quad (3.16)$$

**Theorem 1.** *Suppose that  $D_a$  and  $D_b$  are two subsets as a result of  $k$ th element partition.*

$d_i = f_i^a - f_i^b$  is called contact rate difference between nodes  $a$  and  $b$  for destination  $i$ . Maximum combined contact rate of visiting any of the destinations within a time period occurs when for each  $i \in D_a$  and  $j \in D_b$ ,  $d_i \geq d_j$ .

**Proof.** It is clear that any other partition (including the optimal one) can be generated through a sequence of swaps between two elements, one each from  $D_a$  and  $D_b$ . We show that each swap will deteriorate the combined contact rate level. Suppose  $i$  in  $D_a$  and  $j$  in  $D_b$  are swapped. Based on condition  $d_i \geq d_j$ , we have  $f_i^a - f_i^b \geq f_j^a - f_j^b$ , or  $f_i^a + f_j^b \geq f_i^b + f_j^a$ .

Note that  $f_i^a + f_j^b$  is the combined contact rate involving destinations  $i$  and  $j$ , whereas  $f_j^b + f_i^a$  is the combined contact rate after the swap of  $i$  and  $j$ . The theorem follows. ■

This optimal split algorithm can partition the destinations to nodes with higher contact rate levels; hence, it can reduce the number of forwardings and latency in DTN multicasting.

### Contact Rate Level and Active Level

Both the contact rate level and active level can be estimated based on past contacts. In fact, each mobile node can start with a predefined default value for both contact rate level and active level. It then iteratively enhances its estimates based on new contacts.

In this part, we analyze the necessity using contact rate level and active level together for compare-split. We will use a multicast with two destinations, *black* and *white* nodes, as an example to illustrate. Initially, node  $a$  holds both destinations. Consider that  $a$  is associated with a tape  $T_a$  of a sequence of numbered slots that hold contacts node  $a$  has with other nodes.

1) Case 1: select  $T_a$  with four randomly selected distinct slots - two for black and two for white. The process is called node's  $T$  assignment. To see the reason of having the same condition (both for contact rate level and active level), it is still better to split both destinations between nodes  $a$  and  $b$  than to let  $a$  keep both. We compare the following two approaches. The completion time for the non-split case is the maximum slot number of the first white node and the first black node in node  $a$ 's tape ( $T_a$ ). The completion time of the split case is the maximum slot number of the first white node's slot number in  $T_a$  and the first black node's slot number in  $b$ 's tape ( $T_b$ ). The latter has a shorter expected delivery time.

2) Case 2: to view the importance of the contact rate level during a split, consider a case where  $T_a$  has three black slots and one white slot, while  $T_b$  has one black slot and three white slots. Both nodes  $a$  and  $b$  have the same activity level, and we can easily extend the argument from Case 1 to the fact that it is better to split. It is obviously better to assign the black destination to node  $a$  and the white destination to node  $b$ . Therefore, the priority-based-split algorithm is important as each node ( $a$  or  $b$ ) will increase its chance to reach the corresponding destination directly, resulting in a

smaller latency. A larger contact rate level will also reduce the number of forwardings as its contact rate level is more difficult to be surpassed.

3) Case 3: to view the importance of the active level during a split, consider  $T_a$  with two black slots, two white slots, and four red slots, and  $T_b$  with two black, two white, and no other slots. Although both nodes  $a$  and  $b$  have the same contact rate levels to both destinations, node  $a$  is twice as active as node  $b$ . In this case,  $a$  has contacts with non-destination nodes (red slots) which may have a better contact with destination  $a$  or  $b$ . In other words, destination(s) associated with  $a$  will have a chance to be forwarded to a third node with a better contact rate level to  $a$  and/or  $b$ . Therefore, it is better to assign both destinations to node  $a$ , assuming the benefit from the active status outweighs the benefit from the split (as in Case 1).

### Single Node, Single Copy, and Multiple Copies Models

The single node model uses the minimum number of forwardings (in fact, it is the same as the number of destinations). The delivery ratio can be an issue if the holder has a very low contact rate level to a particular destination. Improvement includes creating a delegation when an encountered node that has better contact rate levels to all destinations. Like the single node model, the single copy model also keeps one copy for each destination, but it allows many holders. The number of forwardings is moderate as each copy is forwarded only when there is a better condition (based on the contact rate levels). Latency is an issue; however, it can be easily traded with the delivery ratio as the destination set is quickly partitioned to subsets with only a single node. Each holder can judiciously determine whether and when to terminate a delivery process.

The multiple copies model includes flooding, which copies the destination set at each node encountered. It is the fastest approach, but it incurs a sufficient number of copies per destination. The number of copies can be controlled through delegation (i.e., copy destination set only to ones with a better condition). It still has  $\frac{5}{3}\sqrt{N}$  ( $N$  is the total number of nodes in the network) [49] number of forwardings, even for a destination set with one destination. TTL-based or ticket-based approaches can control the number of copies, but it is still a challenge to have a good estimate for TTL and ticket numbers to assure delivery while controlling the number of copies. Excessive copies also consume limited memory space at each node, which can prevent and limit the support of multiple flows.

### 3.2.7 Simulation

In this report, we compare the performances of the schemes we mentioned in the previous sections. Each simulation is repeated 1,000 times in MATLAB. In our simulation, the 90 percent confidence interval of each result is within  $\pm 1$  percent. The following metrics are calculated in our simulation.

Table 3.2: Simulation parameters

Trace	Number of nodes	Number of destinations
Levy walks model	100	2, 4, 8, 16, 32
Gaussian distribution model	100	2, 4, 8, 16, 32
Intel trace	118	2-8
Cambridge trace	211	2-11

1. *Average cost*: the average number of forwardings for all destinations to receive the multicast message.

2. *Average latency*: the average latency for all of the delivered destinations to receive the multicast message.

3. *Average latency  $\times$  average cost*: the average latency  $\times$  average cost for all of the delivered destinations to receive the multicast message.

We will compare the multicasting schemes both in synthetic and real traces.

### Simulation Methods and Setting

We have used the traces, not only in synthetic mobility models, but also from real traces. We will compare the number of forwardings, latency, and their product in each trace.

In this thesis, we consider that the given period  $T$  is the whole period, and  $T'$  is 10% of  $T$ .

Our simulation is based on two situations:

- **Without considering recent information (N-RI)**: using equations (3.11) and (3.13) to calculate the contact rate level and active level, which does not consider the recent period information;
- **Considering recent information (RI)**:  $w = 0.5$  in equations (3.12) and (3.13), which gives more weight to the recent information.

In the synthetic mobility models, we set up a 100-node environment. We set up two synthetic traces: Levy walks and Gaussian distribution models.

(a) **Levy walks model**: from the simulation results in [55], a Levy distribution with a scale factor  $c$  and exponent  $\alpha$  in terms of a Fourier transformation can be defined as the following:

$$f_X(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{-itx - |ct|^\alpha} dt, \quad (3.17)$$

where  $\alpha$  is the Levy exponent for flight length distribution, which follows power-law distribution:  $p(l) \sim \frac{1}{l^{1+\alpha}}$ ,  $0 < \alpha \leq 2$ . A power law distribution of pause times is denoted by  $\psi(\Delta t_p) \sim 1/\Delta t_p^{1+\beta}$ , where  $\beta$  is the Levy exponent for pause time distribution,  $0 < \beta \leq 2$ .

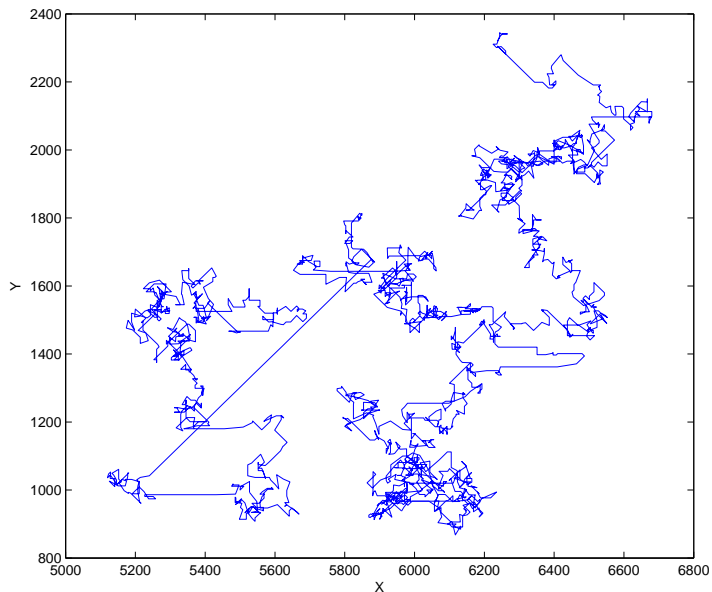


Figure 3.15: Levy walks model.

Table 3.3: Compare-Split-Wait legend of simulation results

split	No condition	Threshold-based
median-binary-split (MBS-W)	1	5
random-binary-split (RBS-W)	2	6
ratio-based-split (RS-W)	3	7
priority-based-split (PS-W)	4	8

Each time two nodes make a contact with each other, we give a contact time and a GPS address. In the Levy walks mobility pattern, we set up the Levy exponent for flight length distribution  $\alpha$ , which is 1, and the Levy exponent for pause time distribution  $\beta$ , which is also 1 in our simulation [56]. Fig. 3.15 shows the mobility pattern of Levy walks. The active level and contact rate levels can be calculated from the generated trace. Because we plan to examine the performance of equal partitioning, we set the destination numbers as  $2^i, i \in \{1, 2, 3, \dots\}$ .

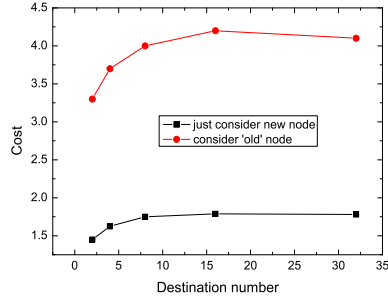
#### (b) Gaussian distribution model

In this model, we first randomly select a node's active level based on a Gaussian distribution model with  $\mu = 5,000$  and  $\sigma = 3,000$ . Once the active level of a node is selected, the active level is partitioned into contact rate levels to all nodes. Suppose node  $a$ 's contact rate level to node  $b$  is  $k$ , then in  $a$ 's  $T$ ,  $k$  slots are randomly selected. The destination number setting and measuring parameters are the same as the Levy walks model.

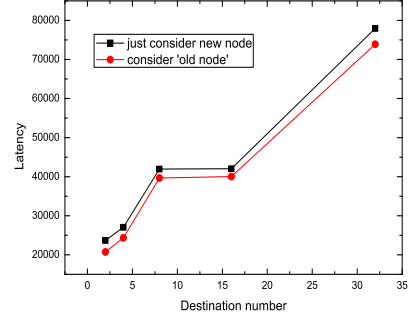
In the real traces, we use Intel and Cambridge traces [50] in our simulation. These data sets consist of contact traces between short-range Bluetooth enabled devices carried by individuals.

Table 3.4: Compare-Split-Focus legend of simulation results

split	No condition	Threshold-based
median-binary-split (MBS-F)	9	13
random-binary-split (RBS-F)	10	14
ratio-based-split (RS-F)	11	15
priority-based-split (PS-F)	12	16



(a) Number of forwardings



(b) Latency

Figure 3.16: Comparison of two *compare* methods.

## Simulation Results

### Compare

As we mentioned in Section 5, if one node has a contact with a node which already has a destination subset, we propose another method: that these two nodes' destination subsets are combined together and then split. From Fig. 3.16, we can see that this method increases the number of forwardings compared to our method that just splits the destination subset to a new node; at the same time, it cannot reduce the latency much. Hence, in the rest of this section, we will not use this method.

We compare the number of forwardings, latency, and their product in 16 multicasting schemes, as shown in Tables 3.3 and 3.4.

#### Without considering recent information (N-RI)

In this part, we will compare all schemes without considering recent information based on equations (3.11) and (3.13).

##### (a) Results in synthetic mobility models

In the Levy walks model, we compared the number of forwardings, latency, and their product among these 16 solutions, as shown in Figs. 3.17 and 3.18. It shows that RS has the fewest number of forwardings and the shortest latency among these four schemes in all conditions (using threshold or not, wait or focus). PS performs better than the other two binary-split schemes, while MBS is better than RBS. We use compare-split-focus with a threshold-based condition in Figs. 3.18(a), 3.18(b), and 3.18(c) to explain. RS-F (Line 15) has about 18% less forwardings than PS-F (16) and 21% less

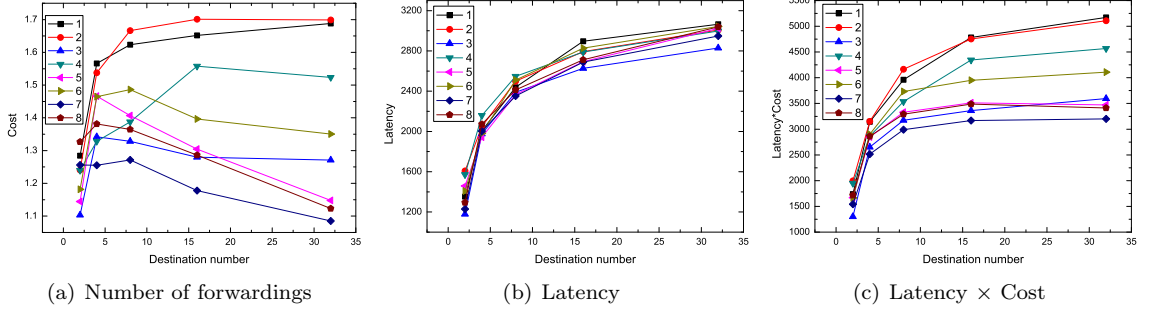


Figure 3.17: Comparison in Levy walks model: compare-split-wait in N-RI.

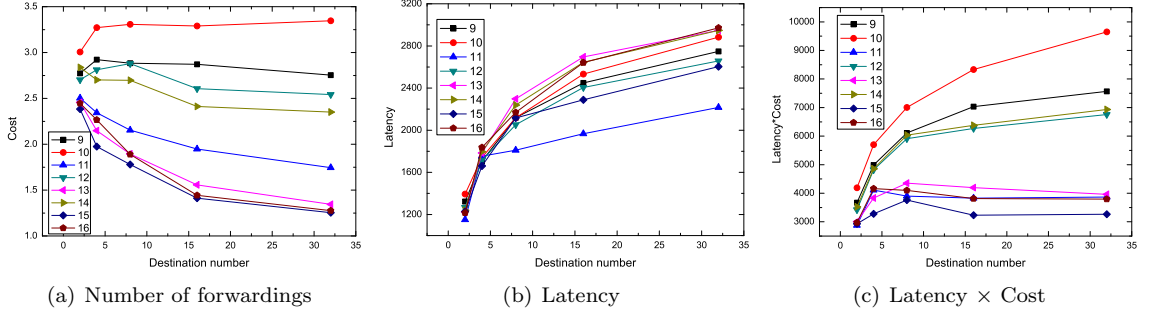


Figure 3.18: Comparison in Levy walks model: compare-split-focus in N-RI.

than MBS-F (13) from Fig. 3.18(a). RS-F reduces the latency by 14% from PS-F and 16% from binary-split in Fig. 3.18(b). By comparing the product of number of forwardings and latency, we can see from Fig. 3.18(c) that RS-F performs better than other schemes. Using the threshold-based condition to decide whether to split the destination set can reduce the number of forwardings by about 9.4%. This means using the threshold-based condition can help the message holder to meet higher contact rate nodes. Using the wait scheme can reduce the number of forwardings, while using the focus scheme can reduce the latency.

In the Gaussian distribution model, RS and BS perform better than PS, as shown in Figs. 3.19 and 3.20. For example, when using compare-split-focus with the threshold-based condition, RS-F(15) has the best performance among these four solutions. Compared with the number of forwardings, it is 2% fewer than MBS-F (13), 16.4% fewer than RBS-F (14), and 33.2% fewer than PS-F (16) from Fig. 3.20(a). In Fig. 3.20(b), we know that RS-F has 8% shorter latency than MBS-F, 10% shorter latency than RBS-F, and 12% shorter latency than PS-F in this case. RS and MBS perform better, when comparing the product of the cost and latency, than the other two schemes both in compare-split-focus and compare-split-wait in Figs. 3.19 and 3.20. Using the threshold-based condition can reduce the latency by about 2.8% and reduce the number of forwardings by about 6.2% from the no condition in the compare step. Using wait can reduce the number of forwardings by about 60%, while using focus can reduce the latency by about 70% when there is only one destination in the

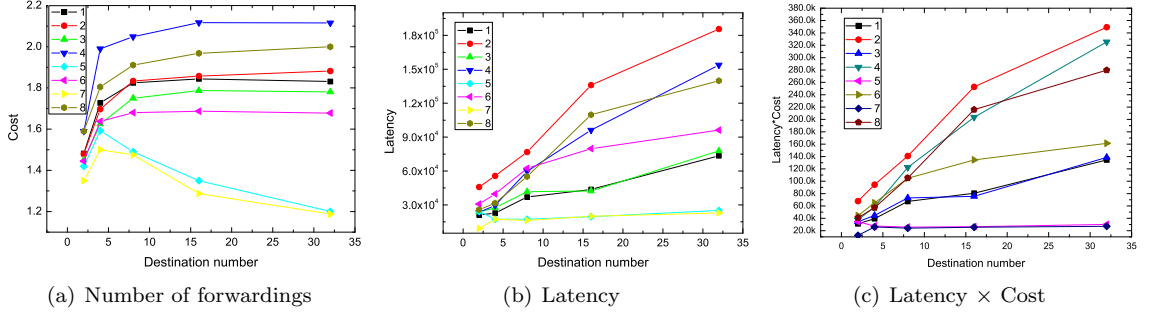


Figure 3.19: Comparison in Gaussian distribution model: compare-split-wait in N-RI.

destination subset.

### (b) Results in real traces

In the Intel trace, RS has a similar number of forwardings for each destination as PS, but much shorter latency, about 22% shorter, from Figs. 3.21 and 3.22. RS performs better than PS when considering the product of latency and cost, as shown in Figs. 3.21(c) and 3.22(c). Using the threshold-based condition in the compare step can reduce the number of forwardings and latency. In the final step, when we want to reduce the number of forwardings, we can choose wait, and if we want to reduce the delay, we can use the focus scheme.

In the Cambridge trace, RS and PS have similar performances in Figs. 3.23 and 3.24. RS has shorter latency while PS has a fewer number of forwardings and a smaller product of latency and cost. These two schemes are both better than BS.

### Considering recent information (RI)

In this part, we will compare all schemes by considering recent information, based on equations (3.12) and (3.13) where  $w$  is 0.5.

From Figs. 3.25 and 3.26, we can see our design split schemes perform similarly as in N-RI, as shown in Figs. 3.17 and 3.18. However, we take more into account recent information than in other schemes, which can provide more information to the Levy walks model; hence, TI reduces the number of forwardings by about 8% and the latency by about 5% as compared with other schemes. In Figs. 3.27 and 3.28, the results do not change a lot from RI, because the recent information has the same contribution to the active level and contact rate level as the long term information in the Gaussian distribution model. From the real traces, as shown in Figs. 3.29, 3.30, 3.31 and 3.32, we can see our designed schemes perform similarly as in N-RI. At the same time, in RI the cost, latency, and their product are reduced compared with N-RI. This means that recent information can present the nodes' mobility pattern better than the previously acquired information.

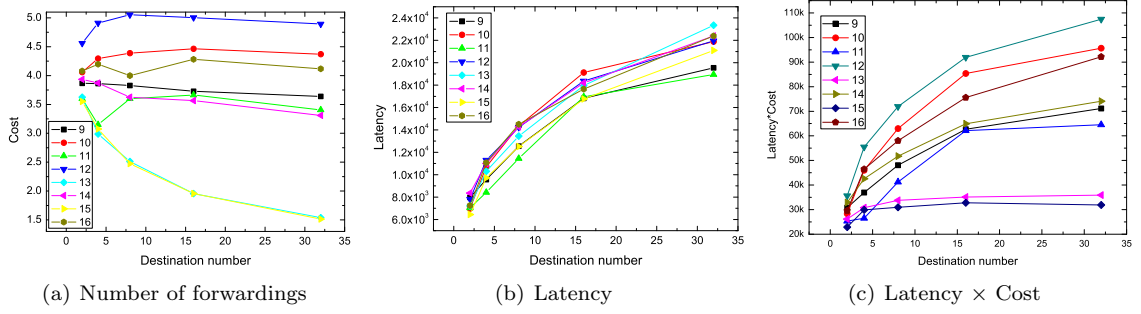


Figure 3.20: Comparison in Gaussian distribution model: compare-split-focus in N-RI.

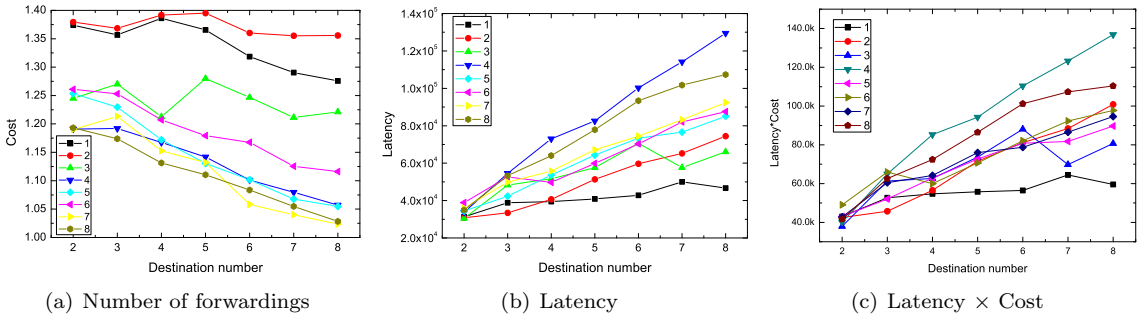


Figure 3.21: Comparison in Intel trace: compare-split-wait in N-RI.

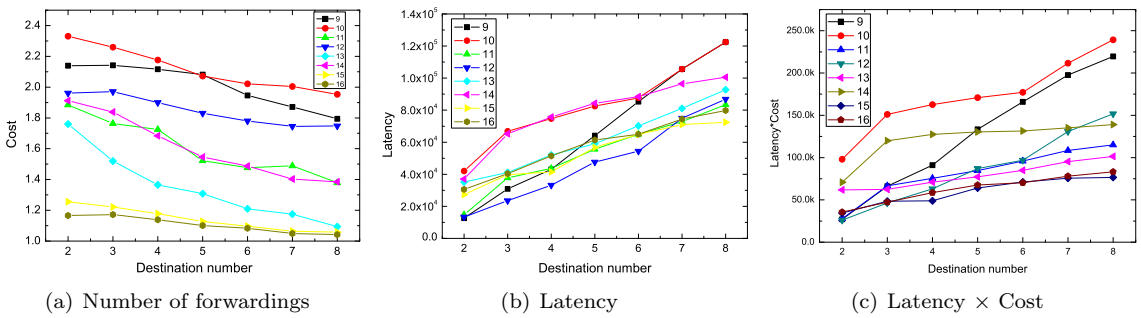


Figure 3.22: Comparison in Intel trace: compare-split-focus in N-RI.

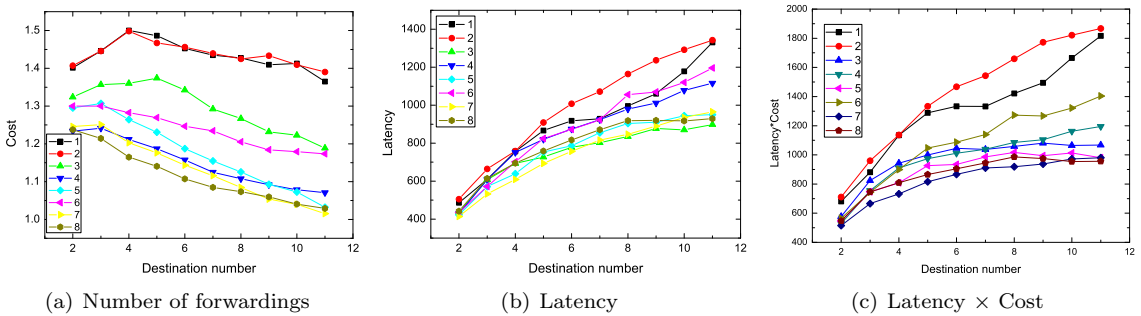


Figure 3.23: Comparison in Cambridge trace: compare-split-wait in N-RI.

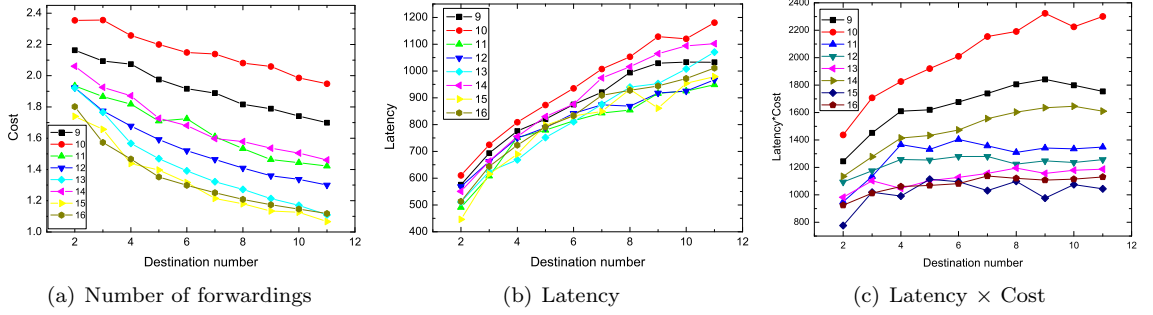


Figure 3.24: Comparison in Cambridge trace: compare-split-focus in N-RI.

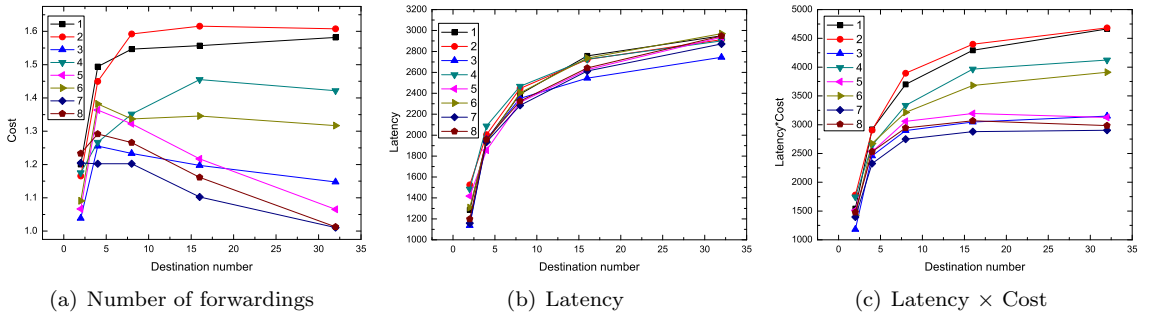


Figure 3.25: Comparison in Levy walks model: compare-split-wait in RI.

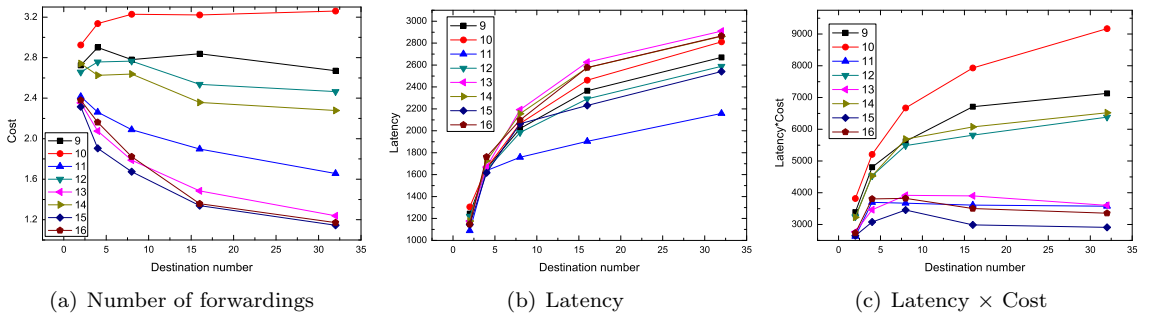


Figure 3.26: Comparison in Levy walks model: compare-split-focus in RI.

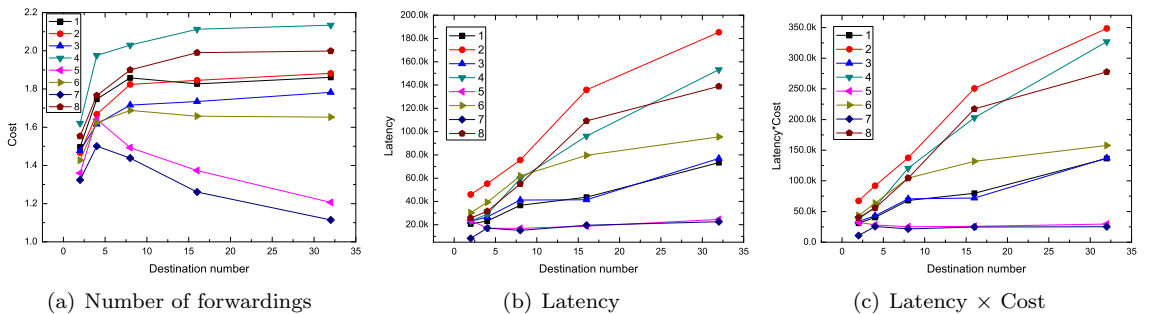
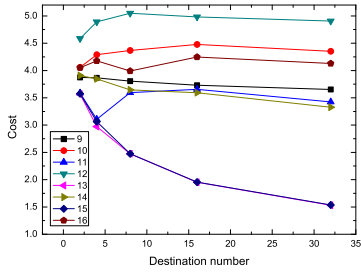
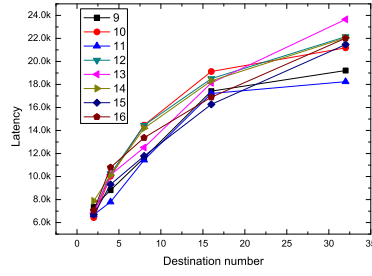


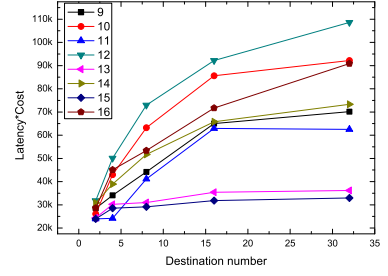
Figure 3.27: Comparison in Gaussian distribution model: compare-split-wait in RI.



(a) Number of forwardings

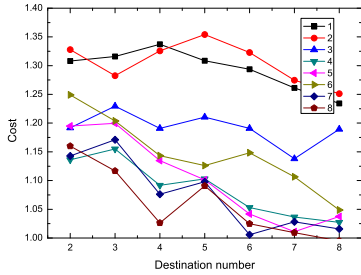


(b) Latency

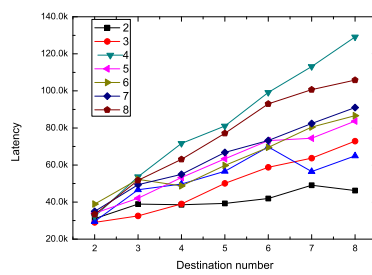


(c) Latency  $\times$  Cost

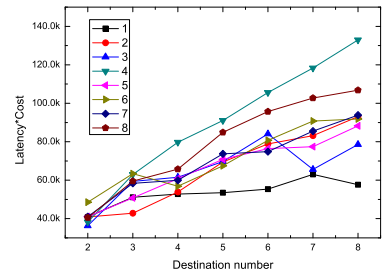
Figure 3.28: Comparison in Gaussian distribution model: compare-split-focus in RI.



(a) Number of forwardings

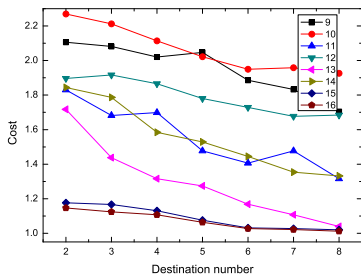


(b) Latency

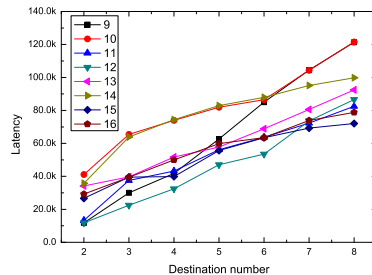


(c) Latency  $\times$  Cost

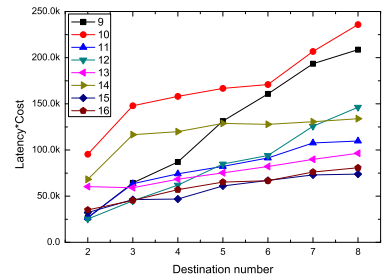
Figure 3.29: Comparison in Intel trace: compare-split-wait in RI.



(a) Number of forwardings

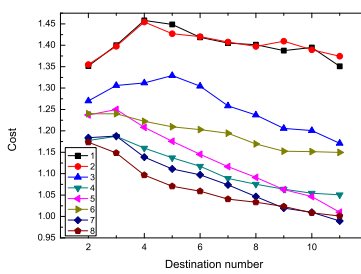


(b) Latency

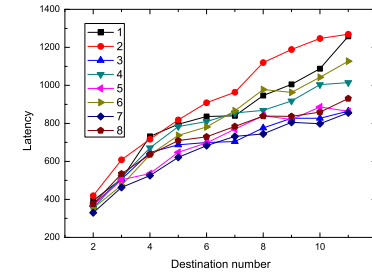


(c) Latency  $\times$  Cost

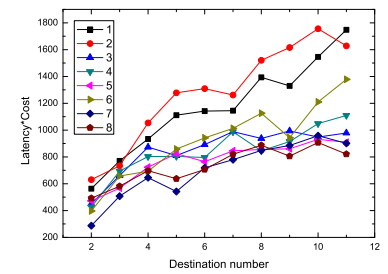
Figure 3.30: Comparison in Intel trace: compare-split-focus in RI.



(a) Number of forwardings



(b) Latency



(c) Latency  $\times$  Cost

Figure 3.31: Comparison in Cambridge trace: compare-split-wait in RI.

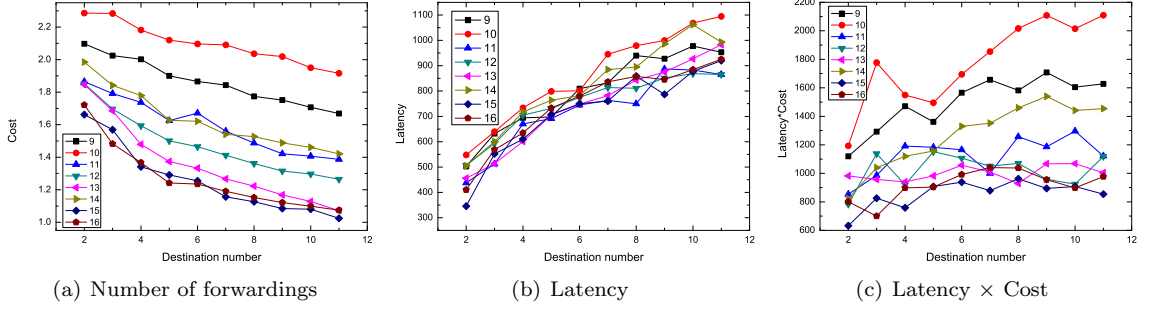


Figure 3.32: Comparison in Cambridge trace: compare-split-focus in RI.

### Summary of Simulation Results

We use non-replication multicasting schemes in DTNs. In the Levy walks model, RS is better than BS as the active levels of the nodes vary significantly. Using RS can assign the destinations to high active level nodes, while BS does not consider the active levels. In the Gaussian distribution model, RS is better than PS as active levels of the nodes are more uniform. This phenomenon is pervasive. In two real traces, the active levels vary significantly. It appears that the role of contact rates and active levels are both very important. Hence, using PS and RS is better than using BS. If the compare step with threshold is used before splitting the destination set, the number of forwardings and latency will both decrease. Table 3.5 shows the best split method in different models. When there is only one destination in the destination set, using the wait scheme can reduce the number of forwardings while using the focus scheme can reduce the latency. From the comparison of N-RI and RI, we can find that the role of recent information is very important.

### 3.2.8 Conclusion

In this report, we focused on developing a non-replication multicasting scheme in DTNs. Our *compare-split* scheme is based on the single copy model with the objective to reach destinations quickly while minimizing the total number of forwardings. We proposed using the node *active level* together with the *contact rate level* to determine when and how to split a destination set during a contact. The split will occur when the message holder has a contact with a node with the sum of the contact rate levels for all destinations being higher than the message holder. In the split process, we used *ratio-based-split* to split the destination set, then compared it with *random-binary-split*, *median-binary-split*, and *priority-based-split* schemes. When there is only one destination left in the destination set, we used *wait* or *focus* to forward the message to the destination.

We compared the performance of these schemes both in synthetic traces and in real traces. Trace driven simulation results showed that compare-split with ratio-based-split, which considers both the

Table 3.5: Conclusion of simulation results

Different models	Best split method
Levy walks mode	RS
Gaussian distribution model	RS
Two real traces	RS / PS

contact rates and active levels, has the best performance. Compare-split-wait has less forwardings while compare-split-focus has shorter latency. We believe that the results obtained from this section present the first step in exploiting the destination set split rule in single copy DTN multicasting. Future research can benefit from our results by developing specific applications based on the provided schemes in DTNs.

### 3.3 Cloud-based Multicasting with Feedback

Inspired by the homophily of social networks that friends are usually similar in characteristics, we present a novel concept – *cloud*, where the nodes in frequent contact with the destinations will form destination clouds. Neighbors in the destination cloud have a special status that can forward the message to the destination directly. We propose a cloud-based multicast scheme with feedback in MSNs with two phases: *pre-cloud* and *inside-cloud*. In the pre-cloud process, the message holder will forward the copy of the multicast message to the encountered node, based on a given forwarding metric. The forwarding metric can be iteratively refined from a feedback control mechanism. In the inside-cloud process, the message holder will wait until it meets with the destinations. We analytically formulate the multicast problem into a continuous Markov chain problem and formally analyze the latency in this model.

#### 3.3.1 Introduction

As stated in Nielson’s report: “as of Q4 2011, 46 percent of US mobile consumers had smartphones, and that figure is growing quickly. In fact, 60 percent of those who said they got a new device within the last three months chose a smartphone over a feature phone” [57]. The mobile social applications in smartphones consume a large amount of bandwidth within the cellular networks. However, the cellular networks are currently overloaded. Thus, it is imperative to develop novel architectures and protocols to solve this problem. A mobile social network (MSN) [8, 33] is a special type of delay tolerant network [1], where the mobile devices contact each other occasionally through opportunistic contacts. The content can be shared among the mobile devices instead of directly downloaded from the infrastructure.

There are multiple types of content to be delivered in MSNs, such as newspapers, online social networking updates, and weather information. The service providers may deliver the content to only a small fraction of mobile users. Then, these content holders walk around and contact other mobile users opportunistically, and exchange the content through either WiFi or Bluetooth. As far as delivery time and cost are concerned, designing an efficient routing protocol to deliver the content to the interested users becomes a multicast problem in MSNs.

In this report, we consider the multicast problem in MSNs. Multicast is a service where a source node sends the message to all of the destinations. In MSNs, a relay node is an encountered node that can store-carry-and-forward the message to the destinations quickly with a small overhead. Therefore, designing a suitable criterion to select good relay nodes becomes a challenging problem in MSN multicast routing.

The nodes that contact each other more frequently will have a higher probability of exchanging the messages, which we called friends. Each node has its frequent contact friends which, together, form a community. Here, we consider the destination community, which has been termed by a new concept – *cloud*. If the nodes contact the destination with high frequency, these nodes have more opportunities to forward the message to the particular destination. We call these kinds of nodes: *destination neighbors*. Destinations and destination neighbors will constitute *destination clouds*. One of the most basic notions governing the structure of social networks is *homophily* [58]. Friends are usually similar in their characteristics, including the places they live, their interests, etc. Therefore, the individuals with more common interests have more of a chance to contact each other. In our proposed multicast scheme, the nodes only obtain their local neighbor information. As shown in Fig. 3.33,  $N_1^1, N_2^1, N_3^1$ , and  $N_4^1$  are destination neighbors of the destination  $D_1$ . They form a destination cloud  $C_1$  for  $D_1$ .  $N_1^2, N_2^2$ , and  $N_3^2$  are destination neighbors of the destination  $D_2$ . They form a destination cloud  $C_2$  for  $D_2$ . We can see that  $N_3^1$  and  $N_1^2$  represent the same node, which belongs to both  $C_1$  and  $C_2$ .

As shown in Fig. 3.33, our proposed multicast scheme has two steps: *pre-cloud* and *inside-cloud*. In the pre-cloud process, initially, the source  $S$  carries a certain amount of copies. When it meets an encountered node, the message forwarding decision is based on the forwarding metric  $F$ , which measures the quality of reaching the destinations. The number of forwarding copies is proportional to the forwarding metrics of these two encountered nodes. The node with the new copies becomes a relay node. In Fig. 1.1,  $R_1$  is a good relay node for  $S$  ( $F_{R_1} > F_S$ ). Thus,  $S$  will forward portions of the copies of the message to  $R_1$ .  $S$  and  $NR_4$  come in contact with each other. However,  $NR_4$  is not a good relay node for  $S$  ( $F_{NR_4} < F_S$ ). Therefore,  $S$  will not forward any copy of the message to  $NR_4$ . When the message holder comes in contact with one of the destination neighbors, the number

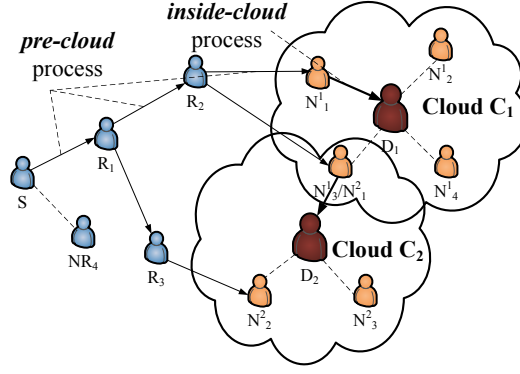


Figure 3.33: Cloud-based Multicast in MSNs.

of copies being forwarded to this destination neighbor depends on the number of destination clouds it belongs to.  $N_1^1$  belongs to cloud  $C_1$ . Therefore, it will receive one copy from  $R_2$ .  $N_3^1/N_1^2$  belongs to two destination clouds,  $C_1$  and  $C_2$ . Therefore,  $R_2$  forwards two copies to it. If the encountering node is one of the destinations, the message holder will forward one copy to it. In the inside-cloud process, the destination neighbors with the message only forward the message to the destinations directly.

Recently, many forwarding metrics have been introduced for routing guidance, such as contact frequency or available buffer space. However, there is no optimal metric. Thus, in this thesis, we present a novel feedback control mechanism. Initially, the forwarding metric is the total number of contacts with all of the destinations in a fixed time interval. After a successful multicast process, we can capture the shortest multicast time for each node to forward the message to the destinations. In the future multicast process, we will use the reciprocal of the shortest captured multicast time in the previous round as the new forwarding metric for the future routing guidance. In this feedback control mechanism, the shortest forwarding time from the node to the destinations can be iteratively refined and used for the future multicast process. The simulation results will show that the routing performance can be improved round by round until an optimal result is reached.

### 3.3.2 Cloud-based Multicast

#### System Model

Assume that there are  $n$  nodes in the whole network. The source node is denoted as  $S$ . The destination set of a multicast is represented as  $\{D_1, D_2, \dots, D_m\}$ . Therefore, there are  $n - m - 1$  relay nodes  $\{R_1, R_2, \dots, R_{n-m-1}\}$ . Each message has a timestamp, which records each time point at which it travels from the source to the destinations. From the historical contact information, each node records its number of contacts with other nodes. Each node has a forwarding metric table recording

the forwarding metrics for itself and other nodes.

### **Cloud Formation**

First, we define a new concept – *cloud*. The destination node is the center of the cloud. The cloud members are the nodes that have high contact frequency with the destination. Thus, we use a contact frequency threshold ( $\theta$ ) to control the size of the cloud. If the contact frequency of a node with the destination node is larger than a threshold, we let this node belong to the cloud of the destination. We call it a destination neighbor. We will assume that a destination neighbor with a copy of a message waits until meeting with the destination, and gives the message to the destination directly. Therefore, if the contact threshold is very small, it will form a large cloud. Although it can reduce the number of forwardings, it will also increase the latency. If the contact threshold is very large, it will form a small, tight cloud. Therefore, it will dilute the role of the cloud. Thus, our first problem is to find an effective threshold for the formation of a cloud. We will compare different values for the contact threshold in the simulation.

### **Forwarding Metric**

The forwarding metric is used to measure the capability of a node to forward the message to the destinations. Intuitively, to improve the performance of MSN, if a node  $i$  (with a copy of the message) encounters a relay node  $j$  (without a copy of the message),  $i$  should give a copy of the message to  $j$  only if the forwarding metric of  $j$  is larger than the forwarding metric of  $i$ . The question is, what is the definition of a good forwarding metric? Is there an optimal forwarding metric? Thus, our second problem is to find an effective definition of forwarding metric, or ultimately, an optimal forwarding metric.

We will use the feedback mechanism to approximate an optimal forwarding metric. We will update the definition of forwarding metric after each multicast process is nearly completed, and use the updated forwarding metric for the next round. We then iterate this updating procedure many times. The iterative updating procedure will eventually give us an approximation to the optimal forwarding metric. To this end, let us denote  $F_i(l)$  to be the forwarding metric of node  $i$  in the  $l$ -th round,  $l = 1, 2, \dots$ . The initial forwarding metric  $F_i(1)$  is defined as follows. We consider in a fixed time interval, a matrix representing quantities proportional to the contact probabilities between

---

**Algorithm 2** Pre-cloud Stage

---

```
/* A non-destination neighbor node  $i$  with  $Z(i)$  copies of a message, encounters node  $j$  without that
message. */
/* Their forwarding metrics are  $F_i$  and  $F_j$ , respectively. */
if  $j$  is one of the destinations then
    Forwarding one copy to  $j$ .
else
    if  $j$  belongs to one or more than one of the destination clouds then
        Forwarding  $\min(Z(i), e(j))$  number of copies to  $j$ , where  $e(j)$  is the number of destination clouds  $j$ 
        belongs to.
    end if
else
    if  $F_j > F_i$  then
        Forwarding  $\left\lceil \frac{F_j}{F_i + F_j} \cdot Z(i) \right\rceil$  number of copies to  $j$ .
    end if
end if
```

---

pairwise nodes:

$$\begin{pmatrix} a_{11} & \cdot & \cdot & \cdot & a_{1n} \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ a_{n1} & \cdot & \cdot & \cdot & a_{nn} \end{pmatrix}.$$

Here, for  $i \neq j$ ,  $a_{ij}$  can be the contact frequency, or inter contact time, etc., between nodes  $i$  and  $j$ . We also let  $a_{ii} = -\sum_{j \neq i} a_{ij}$ . For a concrete example, we may as well let  $a_{ij}$  be the contact frequency between  $i$  and  $j$  for the rest of the report.

Initially, we let the forwarding metric  $F_i(1)$  be the sum of  $a_{ij}$  over all destination nodes  $j$ . In the next two subsections, we will describe the multicast process and how to update the forwarding metric in the feedback mechanism.

### Multicast Process

There are two steps to complete the multicast process: *pre-cloud* and *inside-cloud*.

In the multicast process, we assume that each node  $i$  keeps its metadata, which contains its forwarding metric and  $e(i)$ , where  $e(i)$  is the number of destinations to which  $i$  belongs. Let  $Z(i)$  be the number of message copies possessed by node  $i$ . In the pre-cloud stage, if a message holder  $i$  (with  $Z(i) > 0$ ) is outside of every destination cloud, when it meets node  $j$  (with  $Z(j) = 0$ ), they first exchange their metadata. If  $j$  is a destination node, then  $i$  gives a copy to  $j$ . If  $j$  belongs to one or more destination clouds, then  $i$  gives  $j$   $\min(Z(i), e(j))$  copies. If  $j$  is not in any destination cloud and their forwarding metrics satisfy  $F_j > F_i$ , then  $i$  gives  $j$   $\left\lceil \frac{F_j}{F_i + F_j} \cdot Z(i) \right\rceil$  copies. If  $j$  is not in any destination cloud and satisfy  $F_j \leq F_i$ , then  $i$  does not give  $j$  any copies. Algorithm 2 shows the whole process of the pre-cloud stage.

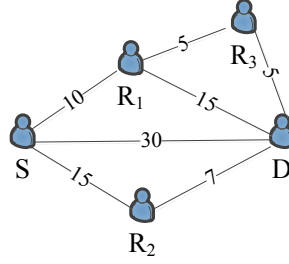


Figure 3.34: Motivation of the feedback mechanism: the values between the mobile nodes are the inter contact time.

We note that initially, the forwarding metric of node  $i$ ,  $F_i(1)$  is the total contact frequencies of node  $i$  with all destinations in the fixed time interval. When the multicast process is nearly completed (above a threshold numbers of destinations who have already received the message), the forwarding metric will be updated to  $F_i(2)$  for the next round. Repeating this process, we will have a sequence of the forwarding metric  $F_i(l)$ , for the  $l + 1$ -th round,  $l = 1, 2, \dots$ . The details will be described in the feedback mechanism in the next subsection.

In the inside-cloud stage, if a destination neighbor has a copy of a message, it only forwards one copy to the destination directly. This process can dramatically reduce the number of forwardings, which is considered to be the cost of the multicast process. Here, the destination node can also act as a relay node.

When the storage of the mobile node is full, the message discard policy will follow FIFO, which means that the first coming message will be discarded.

### Motivation of the Feedback Mechanism

Most of the previous DTN routing protocols are based on the deterministic metrics, such as inter contact time, or number of contacts. As shown in Fig. 3.34, the source  $S$  has a message for the destination  $D$ . When  $S$  meets with nodes  $R_1$  and  $R_2$ , if the forwarding metric is based on the inter contact time between each node to the destination,  $R_2$  is a better relay node (path  $S \rightarrow R_2 \rightarrow D$  with delay of 22, which is shorter than path  $S \rightarrow R_1 \rightarrow D$  with delay of 25). However,  $R_1$  has another path to  $D$  through  $R_3$ , which has a shorter delay (path  $S \rightarrow R_1 \rightarrow R_3 \rightarrow D$  with delay of 20). Since we considered the delivery delay as a measurement metric in this example, the path through  $S \rightarrow R_1 \rightarrow R_3 \rightarrow D$  has the shortest inter-contact time. Therefore, we chose this path as the forwarding path. If we can design a feedback mechanism which updates the forwarding metric according to the delay of the previous round, the routing performance can be improved. We assume that the contacts between each node are stable through the entire time period.

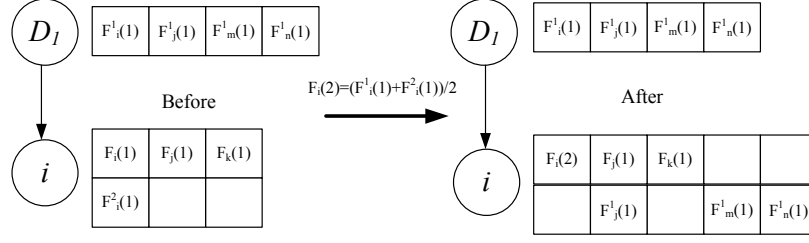


Figure 3.35: Destination feedback process.

### Feedback Control Mechanism

In this part, we describe in detail our proposed feedback control mechanism to improve the accuracy of the forwarding metric prediction.

As we will analyze in the previous part, that using the forwarding metric obtained from the previous multicast process can reduce the latency for the current multicast process. Here, we use an additional space to record the feedback information.

In each round, there is a new multicast message generated. Therefore, each round is considered as an independent multicasting process. For each message, we assign a timestamp, which records the exact time at which the message has been forwarded from one node to another. When the message arrives at one of the destinations, it calculates the latency for all nodes in the routing path from the source to the destination, and records the results in the destination's metadata. After that, the destination node gives the metadata to other nodes when they contact each other. Thus, each encountered node has the metadata that records its multicast latency of the previous round in an additional space. The metadata also contains the same information for other nodes. Therefore, the feedback information is quickly spread out through the whole network. After a node has obtained the metadata (of the previous round) from more than a given threshold number of destinations, it updates its forwarding metric by using the average latency to the destinations. Once the forwarding metric is updated, it is then used in the pre-cloud algorithm for the next round. In our simulation, we set the threshold to be half the number of the destinations.

In Fig. 3.35, for example, destination  $D_1$  calculates the latency for relay nodes  $i, j, m,$  and  $n$ ; all of these nodes have participated in the previous multicast process. After that,  $D_1$  has a contact with  $i$ . In this example, we suppose that the threshold to update the forwarding metric is 2. Suppose that before  $D_1$  and  $i$  contact,  $i$  has already received a feedback from destination  $D_2$ . Therefore,  $i$  will update its forwarding metric to  $F_i(2)$ , which is equal to the average of two feedbacks,  $F_i(2) = \frac{F_i^1(1) + F_i^2(1)}{2}$ , where  $F_i^j(l)$  is the latency of the delivery of a message sent from  $i$  to destination  $j$  using forwarding metric  $F_i(l)$  in the  $l$ -th round. Note that  $i$  also has the similar information of some other nodes.

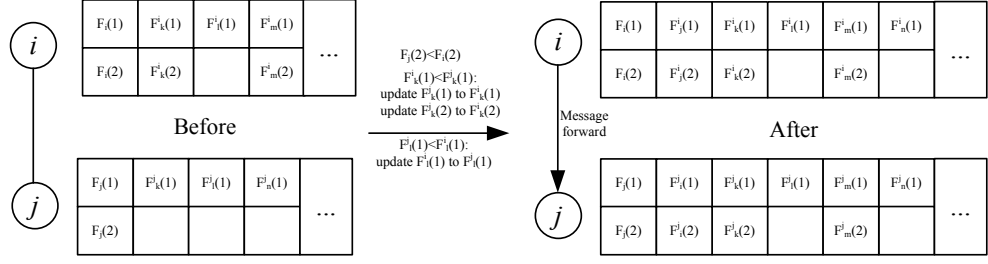


Figure 3.36: Metadata update process.

Fig. 3.36 illustrates the metadata exchange process between two non-destination nodes. After exchanging metadata, these two encountered nodes will record the updated information not only for themselves, but also for all other nodes previously encountered by these two nodes. For example, before this contact,  $i$  has the information about  $k$  of the first round,  $F_k(1)$ , and  $j$  has  $F_k(2)$ . After exchanging the metadata,  $i$  will update its forwarding metric table by recording the information of  $k$  in the second round in its additional buffer space. Hence, the metadata only consumes a small amount of bandwidth.

### 3.3.3 Analysis

In this section, we theoretically analyze the cloud-based multicast scheme with feedback, first without cloud and then with cloud.

#### Multicast without cloud

As discussed in Section 3.3.2, if we do not consider the destination cloud concept, the  $n$ -node MSN can be represented as  $\{N_1, N_2, \dots, N_n\} = \{S, R_1, R_2, \dots, R_{n-m-1}, D_1, D_2, \dots, D_m\}$ . For simplicity of writing formulas, we will relabel  $\{S, R_1, R_2, \dots, R_{n-m-1}, D_1, D_2, \dots, D_m\}$  as  $\{1, 2, \dots, n\}$ , so that  $i = 1$  is representing the source node  $S$ ,  $i = 2, \dots, n-m$  are representing the relay nodes  $R_1, R_2, \dots, R_{n-m-1}$ , and  $i = n-m+1, \dots, n$  represent the destination nodes  $D_1, D_2, \dots, D_m$ , respectively. Then we can model the multicast routing process as a continuous time Markov chain as follows.

We consider a fixed time interval, and for  $i \neq j$ , let  $a_{ij}$  be the contact frequency of node  $i$  to node  $j$  in the fixed time interval. We define  $a_{ii} = -\sum_{j \neq i} a_{ij}$ . Then the contact table  $A$  is an  $n \times n$  matrix,

$$A(i, j) = \begin{pmatrix} a_{11} & \cdot & \cdot & \cdot & a_{1n} \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ a_{n1} & \cdot & \cdot & \cdot & a_{nn} \end{pmatrix},$$

that satisfies  $a_{ij} \geq 0$ , if  $i \neq j$  and  $a_{ii} \leq 0$ , and  $\sum_j a_{ij} = 0, \forall i$ .

It is well-known that the matrix  $A$  can be considered as a generator of a continuous time Markov chain [59] where each node  $i$  waits for a random time  $\xi_i$ , then contacts a node  $j$ . Here,  $\xi_i$  follows an exponential distribution  $\xi_i \sim \exp(-a_{ii})$ , i.e.,  $P(\xi_i > t) = e^{-a_{ii}t}$ , and  $\xi_i, i = 1, 2, \dots, n$ , are independent. It is well-known that if, at a certain time  $i$  is contacting, then the conditional probability that node  $j$  is contacted by  $i$  is:

$$\begin{aligned} P(j|i) &= \frac{a_{ij}}{-a_{ii}}, \quad i \neq j, \\ P(i|i) &= 0. \end{aligned} \tag{3.18}$$

When there is no restriction on which node is contacting which nodes, the first contact time of the whole system after  $t = 0$  is  $\eta_1 = \min(\xi_k; k = 1, \dots, n)$ . Let  $\eta_0 = 0$  and  $\eta_{t+1}$  be the first contact time after  $\eta_t, t = 0, 1, 2, \dots$ . Then the successive contact times  $\eta_t, t = 0, 1, 2, \dots$ , form an increasing sequence of random times. Therefore, if we only observe what happens at contact times, then we obtain an embedded discrete-time Markov chain that describes the transitions of the number of copies of the messages in the multicast routing process. To this end, we let  $Z_t(i)$  be the number of copies that node  $i$  has at time  $t, t = 0, 1, 2, \dots$  and put  $Z_t = (Z_t(1), Z_t(2), \dots, Z_t(n))$ . Initially, we suppose that there are  $L$  copies in the source node, so  $Z_0 = (L, 0, 0, \dots, 0)$ . Since the total number of copies is unchanged,  $Z_t \in \Omega$ , for all  $t$ , where  $\Omega = \{Z; \sum_{i=1}^n Z(i) = L\}$  is the state space of the discrete-time Markov chain.

Let  $F_i$  be the forwarding metric of node  $i$ . Intuitively, in the routing process,  $i$  is more efficient than  $j$  if  $F_i > F_j$ . Therefore, to obtain an efficient routing algorithm,  $i$  should give more copies of the message to  $j$  only if  $F_i < F_j$  when  $i$  and  $j$  are in contact.

Let  $S_t = \{i = 1, 2, \dots, n - m; Z_t(i) \geq 1\}$  be the *message holder set* at time  $t$ . For each  $i \in S_t$ , let  $R_t(i) = \{j = 2, \dots, n - m, F_j > F_i, Z_t(j) = 0, Z_t(i) \geq 2\} \cup \{n - m + 1, \dots, n\}$  be the *receiver set of  $i$*  at time  $t$ . Note that if  $i \in S_t$  and  $Z_t(i) = 1$ , then  $R_t(i) = \{n - m + 1, \dots, n\}$ . Also if  $i \notin S_t$ , then  $R_t(i) = \emptyset$ .

Suppose at time  $t$ , nodes  $i$  and  $j$  come in contact with each other. If  $i$  is in the message holder set and  $j$  is in the receiver set of  $i$ , then  $i$  splits the message copies into two parts according to the ratio of the forwarding metrics, and gives one part to  $j$ . Suppose that  $Z_t = Z$  before  $i$  and  $j$  have a

contact, then after the contact, the state becomes  $Z_{t+1} = Z^{ij}$ , where:

$$Z^{ij}(k) = \begin{cases} Z(k), & \text{if } k \neq i, j, \\ \left\lfloor \frac{F_i}{F_i+F_j} \times Z(i) \right\rfloor, & \text{if } k = i, \\ \left\lceil \frac{F_j}{F_i+F_j} \times Z(i) \right\rceil, & \text{if } k = j, \end{cases} \quad (3.19)$$

for  $i \in S_t$ ,  $j \in R_t(i)$ . Thus  $Z^{ij}$  is the new state obtained by state  $Z$  after  $i$  and  $j$  have exchanged messages. In Eq. 2, if  $k = j$ , which means  $k$  is in the receiver set of  $i$ ,  $k$  will receive  $\left\lceil \frac{F_j}{F_i+F_j} \right\rceil$  in the third equation. At the same time, the message holder  $i$  will take the remainder.

Suppose that at time  $t$  with state  $Z_t$ ,  $i \in S_t$  is a message holder, and  $i$  is the first node in contacting with other nodes, then  $i$  has to wait until it meets with a node  $j \in R_t(i)$  to proceed. If  $j$  is not in  $R_t(i)$ , then  $i$  does not give any copy to  $j$ . In terms of Markov chain, this mechanism is equivalent to imposing a broken-link condition for the Markov chain on the links that go from  $i$  to any node in  $(R_t(i))^c$ , the complement of the receiver set of  $i$ , by setting  $A(i, j) = 0$ , for all  $j \in (R_t(i))^c$ . Then the Markov generator with such broken-link condition is given by

$$A_{Z_t}(i, j) = \begin{cases} A(i, j), & \text{if } i \in S_t, j \in R_t(i), \\ 0, & \text{if } i \in S_t, j \notin R_t(i), i \neq j, \\ \sum_{k \notin R_t(i)} A(i, k), & \text{if } i = j, i \in S_t, \\ A(i, j), & \text{if } i \notin S_t. \end{cases} \quad (3.20)$$

Here, as we can see from the second equation of Eq. 3.20,  $A_{Z_t}(i, j)$  is set to zero if  $i$  is a message holder and  $j$  is not in the message receiver set of  $i$ . Otherwise, if  $i \neq j$ , then  $A_{Z_t}(i, j)$  is unchanged, as seen from the first and the fourth equation of Eq. 3.20. Since the sum over a row of a Markov generators is always zero, the third equation of Eq. 3.20 is just for re-normalization. Algorithm 3 shows the whole process.

By the Markov property and Eq. 3.18, the probability of routing process that goes through a path of states  $(Z_0, Z_1, \dots, Z_n)$  is  $P(Z_0, Z_1, \dots, Z_n) = P(Z_0, Z_1)P(Z_1, Z_2) \cdots P(Z_{n-1}, Z_n)$ , where:

$$P(Z_t, Z_{t+1}) = \begin{cases} \frac{A_{Z_t}(i, j)}{-\sum_{k \in S_t} A_{Z_t}(k, k)}, & \text{if } Z_{t+1} = Z_t^{i, j}, \\ 0, & \text{if } Z_{t+1} \neq Z_t^{i, j}. \end{cases} \quad (3.21)$$

Given time  $t$  with state  $Z_t$ , the first contact time for the routing is  $\eta_{Z_t} = \min(\xi_k; k \in S_t)$ , where  $\xi_k$ 's are independent exponential distributed random variables,  $\xi_k \sim \exp(-A_{Z_t}(k, k))$ . The expected waiting time for the first contact after time  $t$  is  $T_{Z_t} = E(\eta_{Z_t}) = \frac{1}{-\sum_{k \in S_t} A_{Z_t}(k, k)}$ . Then, the total multicast process time is  $T = T_{Z_0} + T_{Z_1} + \dots + T_{Z_{\tau-1}}$ , where  $\tau = \min\{t; t \geq 0, Z_t(j) > 0, \forall j = n - m + 1, \dots, n\}$  is the completion time of the multicast routing process. The expected latency is

---

**Algorithm 3** Analysis of Multicast Routing Without Cloud

---

```
/* When node  $i \in S_t$  with  $Z_t(i)$  copies of a message encounters node  $j \in R_t(i)$  at time  $t$ . */
if  $k = i$  then
   $Z_{t+1}(k) = \lfloor \frac{F_i}{F_i + F_j} \times Z_t(i) \rfloor$ .
else
  if  $k = j$  then
     $Z_{t+1}(k) = \lceil \frac{F_j}{F_i + F_j} \times Z_t(i) \rceil$ .
  else
     $Z_{t+1}(k) = Z_t(k)$ 
  end if
end if
/* Update  $S_{t+1}$ ,  $R_{t+1}(i)$  and Markov generator  $A$  */
if  $i \in S_{t+1}, j \in R_{t+1}(i)$  then
   $A_{Z_{t+1}}(i, j) = A_{Z_t}(i, j)$ .
else
  if  $i \in S_{t+1}, j \notin R_{t+1}(i), i \neq j$  then
     $A_{Z_{t+1}}(i, j) = 0$ .
  end if
else
  if  $i = j, i \in S_{t+1}$  then
     $A_{Z_{t+1}}(i, j) = \sum_{k \notin R_{t+1}(i)} A(i, k)$ .
  else
     $A_{Z_{t+1}}(i, j) = A_{Z_t}(i, j)$ 
  end if
end if
```

---

then given by:

$$E(T) = \sum_{Z_0 \dots Z_{\tau-1}} (T_{Z_0} + T_{Z_1} + \dots + T_{Z_{\tau-1}}) P(Z_0 \dots Z_{\tau-1}). \quad (3.22)$$

### Multicast with cloud

When we include the cloud concept, there is a new type of node: destination neighbor. There are many ways to define what the cloud of a destination is. Here we consider a threshold model. Let  $\theta$  be a preset (fixed) threshold value. We say that a node  $j$ ,  $j = 2, \dots, n-m$ , is in the cloud of a destination node  $d$  if  $a_{jd} > \theta$ . Let  $C_d = \{j = 2, \dots, n-m; a_{jd} > \theta\}$  denote the cloud of the destination node  $d$ ,  $d = n-m+1, \dots, n$ . Note that  $C_d$ 's may be overlapped. Let  $p$  be the cardinality of  $\cup_{d=n-m+1}^n C_d$ . Then the set of the relay nodes can be rewritten as  $\{R_1, R_2, \dots, R_{n-m-1}\} = \cup_{d=n-m+1}^n C_d \cup \{\mathbb{R}_1, \dots, \mathbb{R}_{n-m-p-1}\}$ , here we have used a new notation (in open-faced R) and relabeled the non-destination neighbor relay nodes. Let  $\mathbb{C} = \cup_{d=n-m+1}^n C_d$  be the destination neighbor set and  $\mathbb{R} = \{\mathbb{R}_1, \dots, \mathbb{R}_{n-m-p-1}\}$  be the set of non-destination neighbor relay nodes.

The differences between using cloud and not using cloud lie in the message partition, the receiver set  $R_t(i)$ , and where to set the broken-link conditions for the Markov generator at time  $t$ . In what follows, we continue to use the notation  $e(i)$  for the number of destination clouds to which node  $i$  belongs.

For the cloud model, we first need to define the receiver set of  $i$  at time  $t$ ,  $R_t(i)$ . Suppose at time

$t$  the state is  $Z_t = Z$ . The message holder set  $S_t = \{i = 1, 2, \dots, n - m; Z_t(i) \geq 1\}$  is the same as in the non-cloud model. Let  $i \in S_t$ . If  $i \in C_d$  for some  $d$ , then  $R_t(i) = \{d = n - m + 1, \dots, n; i \in C_d\}$  is the set of all destinations such that  $i$  belongs to their clouds. If  $i \in S_t$  but not in any cloud, then  $R_t(i) = \{j = 2, \dots, n - m, F_j > F_i, Z_t(j) = 0, Z_t(i) \geq 2\} \cup \{n - m + 1, \dots, n\}$  is the same as in the non-cloud model.

Next, we set the rules for the message partition. Suppose  $i$  contacts  $j$  at time  $t$ , then after the contact the state is changed to a new state  $Z^{ij}$  according to the following rules.

Let  $i \in S_t$  and  $j \in R_t(i)$  ( $i \neq j$ ). Then

1. If  $k \neq i, j$ , then the value of  $Z^{ij}(k) = Z(k)$ , which means that  $k$  is not the contact node, so the copy in  $k$ 's buffer will not change.
2. If  $k = j$ , then there are 2 situations:
  - (a) If  $i$  is in the cloud of  $d$  for some  $d$ , then  $j$  must be a destination node and  $Z^{ij}(j) = 1$  and  $Z^{ij}(i) = Z(i) - 1$ ;
  - (b) If  $i$  is not in any cloud, then
    - i. if  $j$  belongs to one or more destination clouds, then  $i$  forwards  $j$   $\min(Z(i), e(j))$  copies;
    - ii. if  $j$  is a destination node ( $j \in D$ ), then  $i$  forwards one copy to  $j$ ;
    - iii. if  $j$  is not in any destination cloud, then  $i$  gives  $j$   $\left\lceil \frac{F_j}{F_i + F_j} \cdot Z(i) \right\rceil$  copies.
3. If  $k$  is the sender ( $k = i$ ), then the number of copies remaining in  $i$ 's buffer is the remaining part of the initial  $Z(i)$  subtracted by the amount forwarded to receiver  $j$ , as discussed above.

In summary, if at time  $t$  the state is  $Z_t = Z$  and  $i, j$  are the first pair in contact after time  $t$ , then after contact the new state  $Z^{ij}$  becomes

1. For  $i \in \cup_d C_d, j \in R_t(i)$ ,

$$Z^{ij}(k) = \begin{cases} Z(k), & \text{if } k \neq i, j, \\ 1, & \text{if } k = j, \\ Z(i) - 1, & \text{if } k = i; \end{cases} \quad (3.23)$$

2. For  $i \notin \cup_d C_d, j \in R_t(i)$ ,

$$Z^{ij}(k) = \begin{cases} Z(k), & \text{if } k \neq i, j, \\ \min(Z(i), e(j)), & \text{if } k = j, j \in \mathbb{C}, \\ 1, & \text{if } k = j, j \in D, \\ \lceil \frac{F_j}{F_i + F_j} \times Z(i) \rceil, & \text{if } k = j, j \notin \mathbb{C} \cup D, \\ Z(i) - \min(Z(i), e(j)), & \text{if } k = i, j \in \mathbb{C}, \\ Z(i) - 1, & \text{if } k = i, j \in D, \\ Z(i) - \lceil \frac{F_j}{F_i + F_j} \times Z(i) \rceil, & \text{if } k = i, j \notin \mathbb{C} \cup D. \end{cases} \quad (3.24)$$

Since a node  $i$  has to wait until it contacts a node  $j$  in  $R_t(i)$  to proceed, any other contacts before that time do not contribute to our multicast process. Therefore, the Markov generator with broken-link conditions is given by

$$A_{Z_t}(i, j) = \begin{cases} A(i, j), & \text{if } i \in S_t, j \in R_t(i), \\ 0, & \text{if } i \in S_t, j \notin R_t(i), i \neq j, \\ \sum_{k \notin R_t(i)} A(i, k), & \text{if } i = j, i \in S_t, \\ A(i, j), & \text{if } i \notin S_t. \end{cases} \quad (3.25)$$

Note that the Markov generator with broken-link conditions has the same form as that of the non-cloud model, Eq. 3.20. However, since the receiver sets are different, the Markov generators are actually different between cloud and non-cloud models.

The formula for the expected latency for the cloud model is the same as the one in the non-cloud model, Eq. 3.22.

By using the feedback control mechanism, the reciprocal of the calculated expected latencies of all nodes are used as the next round forwarding metrics.

We suppose there are 100 nodes in the network. We create a Markov generator  $A$  in a  $100 \times 100$  matrix. The value of  $A_{ij}$  is a randomly generated real number in the interval  $[1, 15]$ , which represents the number of contacts between nodes  $i$  and  $j$ . Then we can numerically analyze the expected latency iteratively.

In Figs. 3.37 and 3.38, we show the latency and the number of forwardings in situations with 4 and 32 destinations, as well as in different cloud sizes. When the contact threshold is 0, the nodes are in the same cloud. The multicast scheme becomes a 2-hop routing transmission, where the source node will forward all copies to an encounter node that has a higher forwarding metric; then, this node will wait to forward the message to the destinations. Although, in this situation, the number of forwardings can be reduced, the latency increases dramatically. Accompanied by the decreasing cloud size, the latency decreases and the number of forwardings increases. When the

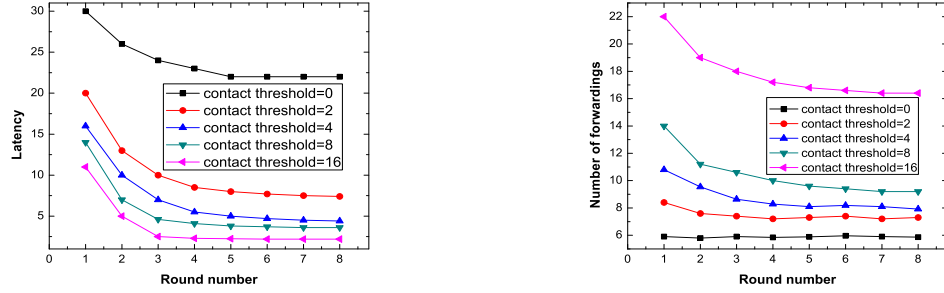


Figure 3.37: Comparing the performance in the 4 destinations situation: ( $L$ ): latency and ( $R$ ): number of forwardings.

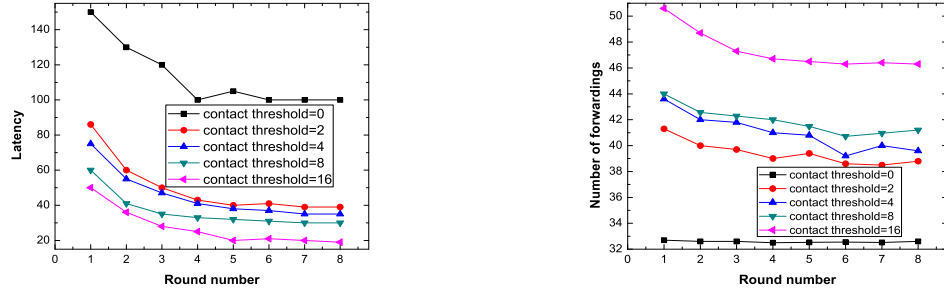


Figure 3.38: Comparing the performance in the 32 destinations situation: ( $L$ ): latency and ( $R$ ): number of forwardings.

contact threshold reaches 16, which exceeds the maximum number of contacts between each node, no destination cloud exists. The message will be forwarded to an encounter node with a higher forwarding metric than the message holder until the destination is reached.

In both sparse (Fig. 3.37) and dense (Fig. 3.38) destination situations, initially the performance is not perfect because of high latency and a large number of forwardings. After the first round of multicast, the destinations will give the feedback information, with the feedback information as the forwarding metric, and the latency and the number of forwardings can be decreased dramatically. The latency decreases by about 33%, and the number of forwardings decreases by about 15% from round 1 to round 2. The latency decreases by about 21%, and the number of forwardings decreases by about 6% from round 2 to round 3. After about four rounds, the network performance converts to a stable situation. Therefore, we believe that our cloud-based multicast with feedback can improve the network performance.

### 3.3.4 Simulation

We compare the performance of the proposed cloud-based multicast scheme with feedback to several state-of-the-art ones, including the delegation forwarding multicast scheme [60], spray-and-focus-based multicast scheme [54], and the epidemic-based multicast scheme [26], in Matlab using two real traces in the following categories:

1) *Various cloud sizes comparison*: when the contact threshold ( $\theta$ ) is large, the destination cloud will disappear while the contact threshold becomes smaller, and the cloud size becomes larger. We compare the performance in different contact thresholds.

2) *Various proportions of destinations*: we compare the performances when the number of destinations varies.

3) *Speed of feedback control information spread*: we show how quickly the feedback control information can be spread to the nodes in the whole network, and the impact of feedback control information for multicast performance.

In the simulation, we consider two performance metrics:

1) *Latency*: the average delivery time for all of the delivered destinations to receive the message. The end-to-end delay is an important concern in cloud-based multicast with feedback design.

2) *Number of forwardings*: the average number of forwardings for all destinations to receive the message. This value represents the overhead in the network, in terms of how many times a message must take in order to reach all the destinations.

The delivery rate performs similarly in all compared schemes, therefore, we only compare the latency and number of forwardings in our report.

## Simulation Methods

We evaluate the performance of our proposed *cloud-based multicast scheme with feedback (CM)*, which is compared with the following multicast schemes:

1) *Delegation forwarding multicast scheme (DM)* [60]: a message holder only replicates a copy of the message to a node with a higher forwarding metric for a destination, unless it is one of the destinations. Then the message holder also raises its own forwarding metric to the higher one it has successfully contacted.

2) *Spray-and-focus-based multicast scheme (SM)*: it is an extension of spray-and-focus [54] in the multicast version. There are two phases in SM: *spray* and *focus*. In the spray phase, the message holder forwards half of the copies of the message to any node it meets without this message. After the spray phase, if any of the destinations have not received the message, it continues to the focus phase. In the focus phase, the message holder only forwards half of the copies of the message to a node with a higher forwarding metric for a destination, unless it is one of the destinations.

3) *Epidemic-based multicast scheme (EM)* [26]: the message holder will replicate a copy of the message to any node it meets without this message. It is a pure flooding-based multicast.

In the simulation, we compare the performances of the routing schemes in two real traces: Infocom 2006 [61] and Intel traces [50].

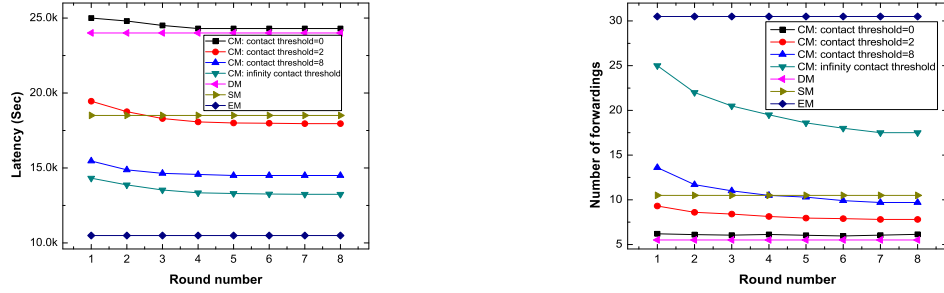


Figure 3.39: Comparing the performance in the Intel trace in a variety of cloud sizes: ( $L$ ): latency and ( $R$ ): number of forwardings.

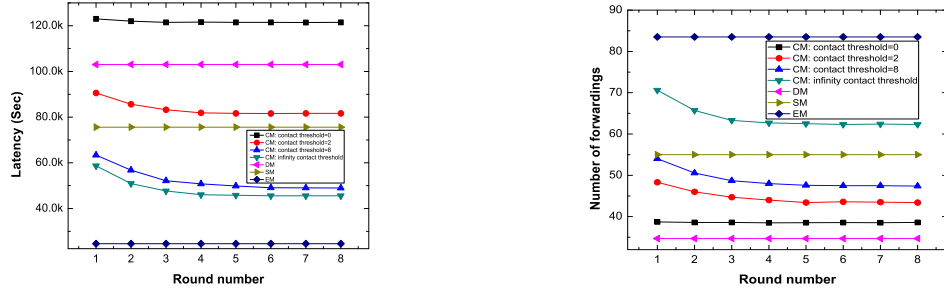


Figure 3.40: Comparing the performance in the Infocom 2006 trace in a variety of cloud sizes: ( $L$ ): latency and ( $R$ ): number of forwardings.

### Various cloud sizes comparison

in this part, we set the number of destinations to 4 in the Intel trace and 32 in the Infocom 2006 trace. From Figs. 3.39 and 3.40, we find that our proposed cloud-based multicast with feedback routing scheme performs best among all of the schemes. Using the feedback control mechanism to update the forwarding metric in each round can improve the multicasting performance. After about five rounds, the two performance parameters stabilize to stable values.

In the Intel trace, by comparing the latency in Fig. 3.39, when we set the contact threshold to 0, the cloud-based multicast scheme with feedback has a longer latency than the delegation-based multicast scheme. When the contact threshold is 0, all nodes in the whole network are considered to be neighbors in one cloud. Therefore, the cloud-based multicast scheme with feedback is operating in an inefficient way. When we increase the contact threshold, the destination clouds become smaller, and the end-to-end latency decreases dramatically. The latency in the 2-contact threshold case reduces by about 26% compared with the latency in the 0-contact threshold case. The 2-contact threshold case has a similar performance as the spray-and-focus-based multicast scheme. It further reduces by about 19.3% from the 2-contact threshold case to the 8-contact threshold case. When we set the contact threshold to infinity, which means that there is no destination cloud, the cloud-based multicast scheme with feedback will always work in the pre-cloud phase. By comparing

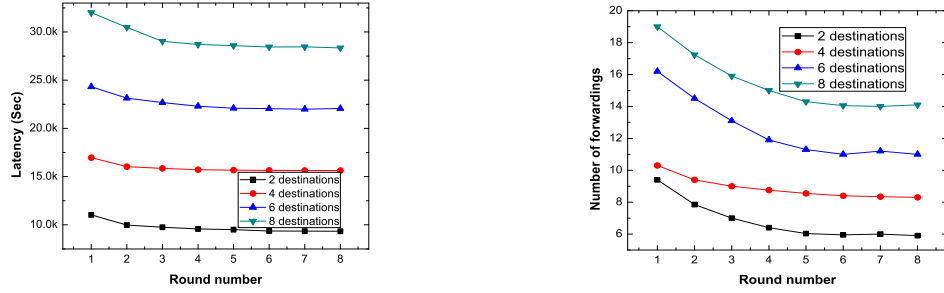


Figure 3.41: Comparing the performance in the Intel trace in different numbers of destinations in each round: ( $L$ ): latency and ( $R$ ): number of forwardings.

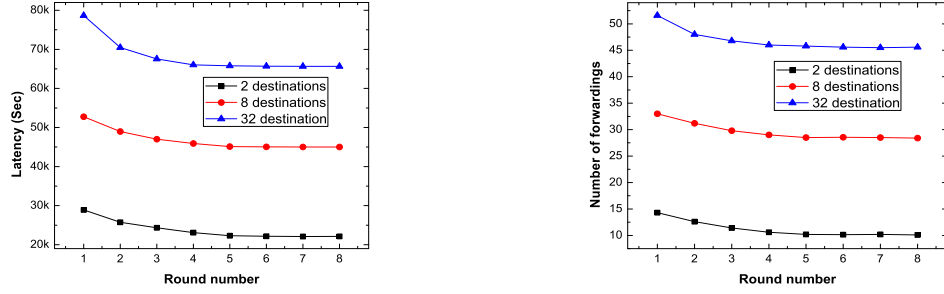


Figure 3.42: Comparing the performance in the Infocom 2006 trace in different numbers of destinations in each round: ( $L$ ): latency and ( $R$ ): number of forwardings.

the latency in this situation with the epidemic-based multicast scheme, we find that our proposed scheme only improves by about 26% in end-to-end latency. However, in Fig. 3.39, the epidemic-based multicast scheme has a much higher number of forwardings than the cloud-based multicast scheme with feedback in the infinity contact threshold case, by about 74%. When the contact threshold reduces, the average number of forwardings also reduces. The 8-contact threshold case has a similar number of forwardings as the spray-and-focus multicast scheme. When the contact threshold reduces to 0, the cloud-based multicast scheme with feedback has a similar number of forwardings as the delegation-based multicast scheme. The results from the Infocom trace show the same trend as in the Intel trace in Fig. 3.40.

In both the Intel and Infocom traces, we find that the overall latency decreases by about 5.6%, and the number of forwarding reduces by about 2.8% from round 1 to round 2. In round 3, the latency decreases by about 11.3%, and the number of forwardings reduces by about 5.7%, compared with round 2.

### Various proportions of destinations

in this part, we set the contact threshold to 4 in both the Intel and Infocom 2006 traces. *In each round, there is a new multicast message generated. Therefore, each round is considered as an independent multicasting process.* From both the Intel and Infocom 2006 traces, we find that the

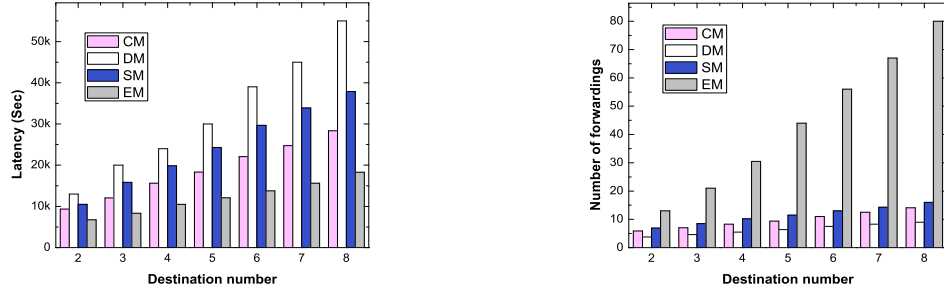


Figure 3.43: Comparing the performance in the Intel trace in different number of destinations: ( $L$ ): latency and ( $R$ ): number of forwardings.

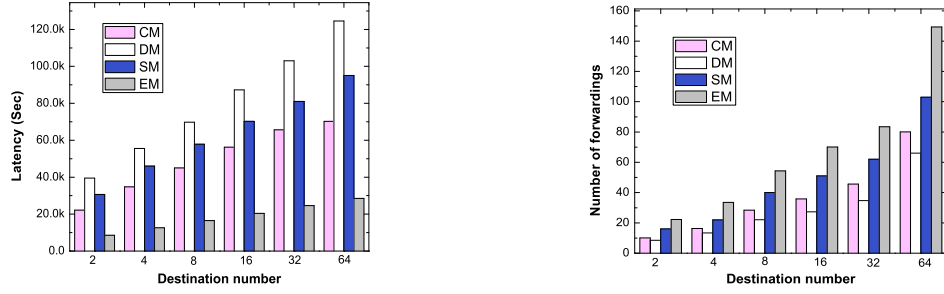


Figure 3.44: Comparing the performance in the Infocom 2006 trace in different number of destinations: ( $L$ ): latency and ( $R$ ): number of forwardings.

multicast routing performance improves by applying the feedback control mechanism in cases with varying numbers of destinations. As shown in Fig. 3.41, in the Intel trace, round 1 can reduce the latency by about 5.2% and reduce the number of forwardings by about 10.4% overall. After about five rounds, the performance metrics convert to stable values. The Infocom 2006 trace shows the same results as the Intel trace in Fig. 3.42. These results show the robustness of our proposed scheme.

From Figs. 3.43 and 3.44, we find that the cloud-based multicast scheme with feedback can decrease the latency dramatically, compared to the delegation-based multicast scheme; at the same time, the number of forwardings only increases slightly in both the Intel and Infocom traces. Our scheme can reduce the latency and number of forwardings compared with the spray-and-focus-based scheme. Compared with the epidemic-based scheme, our proposed cloud-based multicast scheme with feedback can reduce the number of forwardings dramatically.

In the Intel trace, it shows that the cloud-based multicast scheme with feedback can reduce the latency by about 38.7% and 23.2%, overall, compared with the delegation-based multicast scheme and spray-and-focus-based multicast scheme, respectively. In the number of forwardings comparison, the cloud-based multicast scheme with feedback reduces about 49% and 28.4% of the overhead overall, compared with the epidemic-based multicast scheme and spray-and-focus-based multicast scheme, respectively. The results in the Infocom trace show the same trend as the Intel trace.

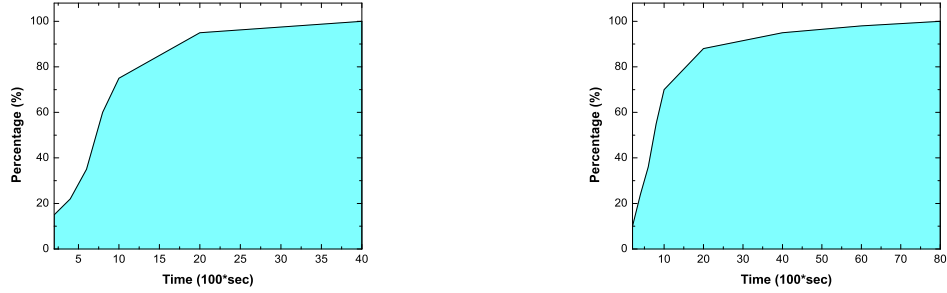


Figure 3.45: Percentage of the nodes that received the feedback information: (*L*): Intel and (*R*): Infocom.

### Feedback information spread speed

in this part, we will show how fast the feedback information can be spread to the nodes in MSNs. Fig. 3.45 shows the percentage of nodes that received the feedback information in both the Intel and Infocom 2006 traces. In the Intel trace, it needs about 2,000 seconds to spread the information to the majority of nodes (95%). After 4,000 seconds, all of the nodes updated their forwarding metrics. In the Infocom 2006 trace, it needs about 8,000 seconds to spread the information to all of the nodes.

### Summary of Simulation

Although the cloud-based multicast scheme with feedback increases the number of forwardings compared with the delegation-based multicast scheme, it significantly reduces the latency, especially when the number of destinations is large. It also reduces the number of forwardings and latency at the same time, compared with the spray-and-focus-based multicast scheme. Compared with the epidemic-based multicast scheme, the cloud-based multicast scheme significantly decreases the number of forwardings, while slightly increasing the latency. When the destination cloud size is large, the cloud-based multicast scheme with feedback performs with a long latency and a small number of forwardings. When we increase the contact threshold, the destination cloud size becomes small. It reduces the latency while increasing the number of forwardings. The feedback control mechanism refines the forwarding metric and improves the performance step by step. The improvement is significant at the beginning, but then, the improvement reduces. After several rounds, the performance metrics will stay with stable values. Considering the characteristics of MSNs, we do not use broadcasting to spread the feedback information. Our feedback control mechanism can spread the information quickly by using the local information exchanging scheme only. The competitive performances in both the Intel and Infocom 2006 traces verify that the cloud-based multicast scheme with feedback performs effectively under different conditions.

### 3.3.5 Conclusion

In this thesis, we proposed a cloud-based multicast scheme with feedback in mobile social networks. Our scheme has two parts: multicast process and feedback control process. First, we introduced a new concept: destination cloud. If the number of contacts, the nodes encountering the destination, is larger than a contact threshold, this node is called the destination neighbor in the destination cloud. In the multicast process, there are two phases: pre-cloud and inside-cloud. In the pre-cloud phase, the message holder will only forward the message to an encountered node with a larger forwarding metric. The forwarding metric updates each round by the feedback control mechanism. In the inside-cloud phase, the message holder will only forward the message to the destination directly. We formally analyzed the delivery latency by modeling our problem into a continuous time Markov chain model. Trace-driven simulation results showed that our proposed cloud-based multicast scheme with feedback performs better than the delegation-based, the spray-and-focus-based and the epidemic-based multicast schemes. We believe that the destination cloud concept will play an important role in multicast routing in mobile social networks. The feedback control mechanism can improve the performance of MSN multicast. Our proposed cloud-based multicast scheme with feedback can be implemented in real world situations. Our future work will focus on implementing real world cloud-based multicast with feedback applications in mobile social networks.

## 3.4 Conclusion of the Chapter

In this chapter, we present three multicast schemes in DTNs. The first one is an extension of delegation forwarding approach. The second one is non-replication multicast scheme which is a compare-split scheme based on the single copy model. Finally, we introduce a cloud-based multicast scheme by leverage the community structure of the network.

In the next chapter, we will study broadcast problem in DTNs.

## Chapter 4

# BROADCAST IN DTNS

Broadcast (or called content distribution or dissemination) is an important application in wireless networks. Data sharing in DTNs poses some unique challenges when the bandwidth is limited for each contact (for example, one transmission per contact) while there are multiple broadcast packets in the local queue.

In this chapter, we first propose greedy solutions for content distribution in Section 4.1. Then, a ticket-based multiple packet broadcasting scheme is introduced in Section 4.2. Finally, in Section 4.3, we apply the social tie concept for information dissemination.

### 4.1 Making Many People Happy: Greedy Solutions for Content Distribution

The increase in multimedia content makes providing good quality of service in wireless networks a challenging problem. Consider a set of users, with different content interests, connected to the same base station. The base station can only broadcast a limited amount of content, but wishes to satisfy the largest number of users. We approach this problem by considering each user as a point in a 2-D space, and each type of broadcast content as a circle. A point that is covered by a circle will be satisfied, and the closer the point is to the center of the circle, the higher the satisfaction.

#### 4.1.1 Introduction

The growth in the amount of mobile devices, such as smartphones and tablets, coupled with the popularity of multimedia content, places a significant strain on existing wireless networks. The field of content distribution for wireless networks [62–65] has emerged in an attempt to address this

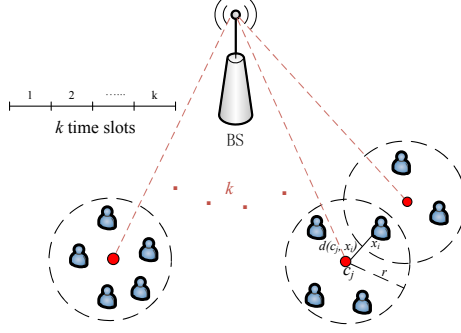


Figure 4.1: Content distribution in a sequence of  $k$  broadcast in the  $m$ -D space.

problem. In this section, we propose to formulate the content distribution problem as an optimization problem which considers the interest distance between the content and the user's interest.

Fig. 4.1 illustrates our problem setting. We assume the base station (BS) can only broadcast  $k$  times in a given period of time to  $n$  number of users. The BS is limited to  $k$  broadcasts for performance reasons, where  $k < n$ . Each user will be happy if the information that was broadcast is close to his interest. We seek to answer the question of choosing the content to broadcast to make the most number of users happy.

This problem can be abstracted as follows. Suppose there are  $n$  points in a 2-dimensional (2-D) space to be covered by  $k$  circles of radius  $r$ . Each point  $x_i$  has a maximum reward of  $w_i$ , and a point will return a reward if it is covered by a circle. The amount of reward is determined by its distance to the center of the circle. Our objective is get the maximum reward for all of the points. In this abstraction, each point corresponds to a particular user's interest. Two dimensions is considered to be two attributes of an interest. Radius determines the boundary of closeness between the content and matching interest. The reward denotes happiness, but with a given bound.

In the real world, people (users here) have their own interests. If the broadcast contents meet their interests, users will be happy and gain some rewards. For example, one user is interested in classic music. If the BS broadcasts light music, this user will be happy to hear this content. Otherwise, if rock music is broadcast, the user will not gain any rewards. The total rewards that each user can gain is capped. The interest distance is the difference between the broadcasting contents and the users interests, which can be used to measure the gained rewards. Our objective is to maximize the total rewards gained by all of the users. We use multi-dimensional vectors ( $m$  keywords in  $m$ -D space) to represent the contents and the users' interests.

In this report, we first formulate our objective into an optimization problem and prove its NP-hardness. Then we propose three local greedy algorithms to solve the problem. These algorithms can be implemented in the  $m$ -D space. The interest distance can be calculated in a general  $p$ -norm.

Here, we only consider the 1-norm and 2-norm (physical distance in the 2-D space) models. We also design weighted and unweighted schemes to better reflect the relative importance of each node in the network. An analysis of the approximation ratio of the greedy algorithms is also given. Trace-driven evaluations show the good performance of our local greedy algorithms.

#### 4.1.2 Optimization problem

Suppose there are  $n$  points in a 2-D space to be covered by  $k$  circles of radius  $r$ . Each point  $i$  has a maximum reward of  $w_i$ . A point will return a reward if it is covered by a circle. The reward can be thought of as the satisfaction of the nodes by receiving the content. The amount of reward is based on the distance between the point and the center of the circle. A point  $i$  can return multiple rewards, but not exceeding  $w_i$ , if it is covered by multiple circles. Note that a larger value of  $k$  tends to have a higher average of satisfiability, but it will also have less frequent service in a time-slotted content distribution system.

As shown in Fig. 4.1,  $c_j$  is the center  $j$ , which has coverage range  $r$ .  $x_i$  is the point  $i$  and  $d(c_j, x_i)$  is the interest distance between  $x_i$  and  $c_j$ . If  $x_i$  is in  $c_j$ 's coverage range, the reward that point  $x_i$  received is the inverse of the interest distance between  $x_i$  and  $c_j$ . Otherwise, point  $x_i$  cannot get any reward. The problem can be presented as the equation below:

$$\psi(c_j, x_i) = \begin{cases} w_i(1 - \frac{d(c_j, x_i)}{r}) & d(c_j, x_i) \leq r \\ 0 & otherwise \end{cases} \quad (4.1)$$

One point can belong to different centers, but its received reward can not exceed the maximum reward. Hence,

$$g(c_i) = \sum_{j=1}^k \psi(c_j, x_i). \quad (4.2)$$

$$f(x_i) = \begin{cases} g(x_i), & g(x_i) \leq w_i \\ w_i, & otherwise \end{cases} \quad (4.3)$$

Let  $[\cdot]_+$  be a function defined as  $[\cdot]_+ = \max(\cdot, 0)$ . This target problem can be formulated as the following maximization problem:

$$\text{maximize} \quad \sum_{i=1}^n f(x_i). \quad (4.4)$$

Then an equivalent equation can be formulated:

$$\max_{\{c_j\}_{j=1}^k} \sum_i w_i \min \left( \sum_j \max \left( 1 - \frac{d(c_j, x_i)}{r}, 0 \right), 1 \right). \quad (4.5)$$

Our target problem can be formulated as the following maximization optimization problem:

$$\max_{C:|C|=k, C \subset \mathcal{V}} f(C) \quad (4.6)$$

where  $\mathcal{V}$  is an infinite set of indices that point to the real vectors in  $\mathbb{R}^m$  space, and  $C$  is a subset of  $\mathcal{V}$ . For simplicity, from now on we will directly view  $\mathcal{V}$  as an infinite set of real vectors in  $\mathbb{R}^m$  space, and represent  $C$  as a subset of real vectors in  $\mathbb{R}^m$ :  $C = \{c_1, \dots, c_k\}$ . The objective function in (4.6) is a function of  $C$ , expressed as:

$$f(C) = \sum_{i=1}^n w_i \min \left\{ \sum_{j=1}^k \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, 1 \right\} \quad (4.7)$$

**Theorem 0.** *The target optimization problem (4.6) is an NP-hard problem.*

**Proof.** It is well known that maximizing a submodular function subject to a size constraint is NP-hard [66]. Below we will show that this objective function is a submodular function by proving two lemmas. ■

**Lemma 0a.** *Given real numbers  $a \geq 0$ ,  $b \geq 0$ , and  $y \geq 0$ , we have:*

$$\begin{aligned} g &= \min\{y + a, 1\} - \min\{a, 1\} \\ &- \min\{y + a + b, 1\} + \min\{a + b, 1\} \geq 0. \end{aligned} \quad (4.8)$$

**Proof.** Considering different values for  $a$ ,  $b$  and  $y$ , the problem above can be solved in three cases.

We give the proof below for each case.

- Case 1 —  $y + a + b < 1$ :  
we have  $g = (y + a) - a - (y + a + b) + (a + b) = 0 \geq 0$ .
- Case 2 —  $y + a + b \geq 1$ ,  $y + a \leq 1$ :  
if  $a + b \leq 1$ , we have  $g = (y + a) - a - 1 + (a + b) = y + a + b - 1 \geq 0$ ;  
if  $a + b > 1$ , we have  $g = (y + a) - a - 1 + 1 = y \geq 0$ .
- Case 3 —  $y + a > 1$ :  
if  $a \leq 1$  and  $a + b \leq 1$ , we have  $g = 1 - a - 1 + (a + b) = b \geq 0$ ;

if  $a \leq 1$  and  $a + b > 1$ , we have  $g = 1 - a - 1 + 1 = 1 - a \geq 0$ ;

if  $a > 1$ , we have  $g = 1 - 1 - 1 + 1 = 0 \geq 0$ ;

By combing these three cases, (8) is proved. ■

**Lemma 0b.** *Let  $f(\emptyset) = 0$ ; then the function  $f(C)$ , defined in (4.7), is a submodular function.*

**Proof.** Consider any two subsets  $A, B \subseteq \mathcal{V}$  and  $A \subset B$ . Without loss of generality, we can assume  $A = \{a_1, \dots, a_k\}$  and  $B = \{a_1, \dots, a_k, b_1, \dots, b_{k'}\}$  for  $k \geq 0$  and  $k' > 0$ . Then for any  $s \in \mathcal{V}$  and  $s \notin B$ , we have:

$$\left( f(A \cup \{s\}) - f(A) \right) - \left( f(B \cup \{s\}) - f(B) \right) = \sum_{i=1}^n w_i \nabla R_i$$

where:

$$\begin{aligned} \nabla R_i &= \min \left\{ \left[ 1 - \frac{d(s, x_i)}{r} \right]_+ + \sum_{j=1}^k \left[ 1 - \frac{d(a_j, x_i)}{r} \right]_+, 1 \right\} \\ &- \min \left\{ \sum_{j=1}^k \left[ 1 - \frac{d(a_j, x_i)}{r} \right]_+, 1 \right\} \\ &- \min \left\{ \left[ 1 - \frac{d(s, x_i)}{r} \right]_+ + \sum_{j=1}^k \left[ 1 - \frac{d(a_j, x_i)}{r} \right]_+ \right. \\ &+ \left. \sum_{j=1}^{k'} \left[ 1 - \frac{d(b_j, x_i)}{r} \right]_+, 1 \right\} \\ &+ \min \left\{ \sum_{j=1}^k \left[ 1 - \frac{d(a_j, x_i)}{r} \right]_+ + \sum_{j=1}^{k'} \left[ 1 - \frac{d(b_j, x_i)}{r} \right]_+, 1 \right\} \end{aligned} \quad (4.9)$$

According to Lemma 0a,  $\nabla R_i \geq 0$ . Therefore,  $f(A \cup \{s\}) - f(A) \geq f(B \cup \{s\}) - f(B)$ , and function  $f(C)$  is a submodular function [66]. ■

It is well known that maximizing a submodular function subject to a size constraint is NP-hard [66]. Thus our target maximization problem is an NP-hard problem.

### Round-based heuristic algorithm

Because this is an NP-hard optimization problem, we introduce a round-based heuristic algorithm. There are  $k$  rounds ( $k$  is same as the number of centers), in each round the reward value can be optimized. The round-based heuristic algorithm is described in Algorithm 4.

---

**Algorithm 4** Round-based Heuristic Algorithm
 

---

- 1: let  $y_i^1 = 1$  for  $i = 1 \dots n$ .
  - 2: **for**  $j = 1$  to  $k$  **do**
  - 3:    $\{g(j), c_j, z_1^j, \dots, z_n^j\} \leftarrow$  maximizing  $\sum_{i=1}^n w_i z_i^j$  by solving Equation (4.10).
  - 4:   update  $y_i^{j+1} = y_i^j - z_i^j$ , for  $i = 1 \dots n$ .
  - 5: **end for**
- 

In Algorithm 4, the optimization problem involved in the  $j$ th round is as follows:

$$\begin{aligned}
 g(j) &= \max_{c_j} \sum_{i=1}^n w_i \min \left\{ \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, y_i^j \right\} \\
 &= \max_{c_j, z_1^j, \dots, z_n^j} \sum_{i=1}^n w_i z_i^j \\
 &\quad \text{s.t.} \quad z_i^j \leq y_i^j, \forall i = 1, \dots, n \\
 &\quad \quad z_i^j \leq \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, \forall i = 1, \dots, n
 \end{aligned} \tag{4.10}$$

The optimization problem Equation (4.10) is equivalent to the following optimization problem.

$$\begin{aligned}
 \max_{c_j, z_1^j, \dots, z_n^j, s_1, \dots, s_n} \sum_{i=1}^n w_i s_i z_i^j \\
 \text{s.t.} \quad z_i^j \leq y_i^j, \forall i = 1, \dots, n \\
 z_i^j \leq 1 - \frac{d(c_j, x_i)}{r}, \forall i = 1, \dots, n \\
 0 \leq s_i \leq 1, \forall i = 1, \dots, n
 \end{aligned} \tag{4.11}$$

It is straightforward to show that for any feasible solution  $(c, z_1, \dots, z_n)$  of (4.10), we can simply construct  $s_i$  by setting it to 1 if  $1 > d(c, x_i)/r$  and to 0 otherwise. Then  $(c, z_1, \dots, z_n, s_1, \dots, s_n)$  is one feasible solution for (4.11) with the same objective value. On the other hand, for any optimal solution  $(c^*, z_1^*, \dots, z_n^*, s_1^*, \dots, s_n^*)$  of (4.11), it is straightforward to see that  $(c^*, z_1^*, \dots, z_n^*)$  is one feasible solution for (4.10) with the same objective value, and therefore is also an optimal solution of (4.10).

The optimization problem (4.11) is a quadratic programming. Rewrite its objective in a quadratic form  $\alpha Q \alpha^\top$ , where  $\alpha = [z_1^j, \dots, z_n^j, s_1, \dots, s_n, c_j]$  and:

$$Q = \begin{bmatrix} \text{zeros}(n, n) & 0.5 \text{diag}(\mathbf{w}) & \text{zeros}(n, 1) \\ 0.5 \text{diag}(\mathbf{w}) & \text{zeros}(n, n) & \text{zeros}(n, 1) \\ \text{zeros}(1, n) & \text{zeros}(1, n) & \text{zeros}(1, 1) \end{bmatrix} \tag{4.12}$$

for  $\mathbf{w} = [w_1, \dots, w_n]$ .

It is easy to see that  $Q$  has both positive and negative eigenvalues. Thus Equation (4.11) is NP-hard according to [67], and Equation (4.10) is NP-hard as well accordingly.

In the following, we present a theorem for the approximation ratio of the round-based heuristic algorithm.

**Theorem 1** *Algorithm 1 (round-based heuristic) achieves an approximation ratio of  $1 - (1 - 1/k)^k$ , where  $k$  is the number of selected centers.*

**Proof of Theorem 1:** We will prove Theorem 1 by proving the following Lemma.

**Lemma 1** *Let  $g(j) = \sum_{i=1}^n w_i z_i^j$  be the optimal objective function value for the optimization problem (4.10) in the  $j$ th round of the round-based heuristic algorithm. We then have the following results:*

(a)  $g(1) \geq \frac{1}{k} f_{opt}$ ; (b)  $g(j) \geq \frac{1}{k} (f_{opt} - \sum_{\ell=1}^{j-1} g(\ell - 1))$ , for  $j = 2 \dots k$ .

**Proof.** The proof for (a) is straightforward. From the round-based heuristic algorithm, we can see that  $kg(1)$  can be obtained by:

$$\begin{aligned}
kg(1) &= k \max_{c_1} \sum_{i=1}^n w_i \min \left\{ \left[ 1 - \frac{d(c_1, x_i)}{r} \right]_+, 1 \right\} \\
&= \max_{c_1, \dots, c_j} \sum_{i=1}^n w_i \sum_{j=1}^k \min \left\{ \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, 1 \right\} \\
&\geq \max_{c_1, \dots, c_j} \sum_{i=1}^n w_i \min \left\{ \sum_{j=1}^k \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, 1 \right\} \\
&= f_{opt}
\end{aligned} \tag{4.13}$$

Thus (a)  $g(1) \geq \frac{1}{k} f_{opt}$  is proved.

Let the reward obtained on point  $i$  from the first  $j - 1$  rounds of the round-based heuristic algorithm is  $h_i^{j-1} = \sum_{\ell=1}^{j-1} z_i^\ell$ , thus  $y_i^j = 1 - h_i^{j-1}$ , and  $\sum_{i=1}^n h_i^{j-1} = \sum_{\ell=1}^{j-1} g(\ell - 1)$ . Let  $\{c_1^*, \dots, c_k^*\}$  be the optimal solution returned by the optimization. Therefore:

$$\begin{aligned}
kg(j) &= k \max_{c_j} \sum_{i=1}^n w_i \min \left\{ \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, y_i^j \right\} \\
&= \max_{c_{j_1}, \dots, c_{j_k}} \sum_{i=1}^n w_i \sum_{\ell=1}^k \min \left\{ \left[ 1 - \frac{d(c_{j_\ell}, x_i)}{r} \right]_+, y_i^j \right\} \\
&\geq \sum_{i=1}^n w_i \sum_{j=1}^k \min \left\{ \left[ 1 - \frac{d(c_j^*, x_i)}{r} \right]_+, y_i^j \right\} \\
&\geq \sum_{i=1}^n w_i \min \left\{ \sum_{j=1}^k \left[ 1 - \frac{d(c_j^*, x_i)}{r} \right]_+, y_i^j \right\} \\
&= \sum_{i=1}^n w_i \min \left\{ \sum_{j=1}^k \left[ 1 - \frac{d(c_j^*, x_i)}{r} \right]_+, 1 - h_i^{j-1} \right\} \\
&\geq \sum_{i=1}^n w_i \left( \min \left\{ \sum_{j=1}^k \left[ 1 - \frac{d(c_j^*, x_i)}{r} \right]_+, 1 \right\} - h_i^{j-1} \right) \\
&= f_{opt} - \sum_{\ell=1}^{j-1} g(\ell - 1)
\end{aligned} \tag{4.14}$$

Thus (b)  $g(j) \geq \frac{1}{k}(f_{opt} - \sum_{\ell=1}^{j-1} g(\ell - 1))$  is proved. ■

Now let  $f(j) = \sum_{\ell=1}^j g(\ell)$ . Then  $f(k)$  is the reward value, i.e. the sum of objective values in all  $k$  rounds, returned by the round-based heuristic algorithm above. According to Lemma 1, we have:

$$f(1) = g(1) \geq \frac{1}{k}f_{opt}; \quad (4.15)$$

$$\begin{aligned} f(j) &= f(j-1) + [f(j) - f(j-1)] \\ &= f(j-1) + g(j) \\ &\geq f(j-1) + \frac{1}{k}(f_{opt} - \sum_{\ell=1}^{j-1} g(\ell - 1)) \\ &= f(j-1) + \frac{1}{k}(f_{opt} - f(j-1)) \\ &= (1 - \frac{1}{k})f(j-1) + \frac{1}{k}f_{opt}, \quad \text{for } j = 2, \dots, k. \end{aligned} \quad (4.16)$$

Combining the base case inequation (4.15) and the recursive inequation (4.22), we can get:

$$f(k) \geq (1 - (1 - 1/k)^k)f_{opt} \geq (1 - 1/e)f_{opt}. \quad (4.17)$$

Therefore, the approximation ratio of the round-based heuristic algorithm is  $(1 - (1 - 1/k)^k)$ , and it is bounded by  $(1 - 1/e)$ .

### 4.1.3 Greedy heuristic algorithm

Since the problem, as we discussed in the previous section, is NP-hard, here we will introduce three local greedy algorithms to solve the problem. The idea is to use the greedy approach at each point locally when seeking an optimal solution. To simplify the discussion, we use 2-D and 2-norm to illustrate. “The scope of a content” is represented by a 2-D disk.

We assume that there are  $n$  number of points in the whole network  $\{x_1, x_2, \dots, x_n\}$ , and each point’s maximum reward  $\{w_1, w_2, \dots, w_n\}$ . We need to find  $k$  number of disks to cover them. The radius of the disks is predefined as  $r$ . Our objective is to find the “best” disks which can get the largest reward as we discussed in the previous section.

#### Local greedy algorithm

The approximation ratio we obtained in Theorem 1 is for the round-based heuristic algorithm; Equation (4.10) is still an NP-hard problem. Here we solve a local greedy algorithm instead, which picks the point that leads to the largest reward as the center in each round. This algorithm is shown in Algorithm 5.

---

**Algorithm 5** Local Greedy Algorithm
 

---

- 1: let  $y_i^1 = 1$  for  $i = 1 \dots n$ .
  - 2: **for**  $j = 1$  to  $k$  **do**
  - 3:    $\{g(j), c_j, z_1^j, \dots, z_n^j\} \leftarrow$  maximizing  $\sum_{i=1}^n w_i z_i^j$  by solving Equation (4.18).
  - 4:   update  $y_i^{j+1} = y_i^j - z_i^j$ , for  $i = 1 \dots n$ .
  - 5: **end for**
- 

In Algorithm 5, the optimization problem involved in the  $j$ th round is as follows:

$$\begin{aligned}
 g(j) &= \max_{c_j \in \{x_1, \dots, x_n\}} \sum_{i=1}^n w_i \min \left\{ \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, y_i^j \right\} \\
 &= \max_{c_j \in \{x_1, \dots, x_n\}, z_1^j, \dots, z_n^j} \sum_{i=1}^n w_i z_i^j \\
 \text{s.t.} \quad & z_i^j \leq y_i^j, \forall i = 1, \dots, n \\
 & z_i^j \leq \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, \forall i = 1, \dots, n
 \end{aligned} \tag{4.18}$$

In each round of Algorithm 5, all of the points are candidates for the center of the disk. The point that gives the maximum *coverage reward*, which includes all of the rewards contributed by the points covered by this disk, will be selected as the center of the disk. If there are a number of points, which have the same maximum coverage reward, our selection will be based on the index of the points (index refers to the ID of the point).

The approximation ratio of this local greedy algorithm is given in Theorem 2.

**Theorem 2** *Algorithm 5 (local greedy) achieves an approximation ratio of  $1 - (1 - 1/n)^k$ , where  $k$  is the number of selected centers and  $n$  is the number of points.  $n > k$  is assumed by default.*

**Proof of Theorem 2:** We will prove Theorem 2 by proving the following Lemma.

**Lemma 2** *Let  $g(j) = \sum_{i=1}^n w_i z_i^j$  be the optimal objective function value for the optimization problem (4.18) in the  $j$ th round of the local greedy algorithm. We then have the following results: (a)  $g(1) \geq \frac{1}{n} f_{opt}$ ; (b)  $g(j) \geq \frac{1}{n} (f_{opt} - \sum_{\ell=1}^{j-1} g(\ell - 1))$ , for  $j = 2 \dots k$ .*

**Proof.** It is obvious  $f_{opt} \leq \sum_{i=1}^n w_i$ . The proof for (a) is straightforward. From the local greedy algorithm, we have  $g(1) \geq \max_i w_i$ . Therefore:

$$g(1) \geq \frac{\max_i w_i}{\sum_{i=1}^n w_i} f_{opt} \geq \frac{1}{n} f_{opt} \tag{4.19}$$

Thus (a)  $g(1) \geq \frac{1}{n} f_{opt}$  is proved.

Let the reward obtained on point  $i$  from the first  $j - 1$  rounds of the round-based heuristic algorithm is  $h_i^{j-1} = \sum_{\ell=1}^{j-1} z_i^\ell$ , thus  $y_i^j = 1 - h_i^{j-1}$ , and  $\sum_{i=1}^n h_i^{j-1} = \sum_{\ell=1}^{j-1} g(\ell - 1)$ . Let  $\{c_1^*, \dots, c_k^*\}$  be the optimal solution returned by the optimization.

Therefore:

$$\begin{aligned}
g(j) &= \max_{c_j \in \{x_1, \dots, x_n\}} \sum_{i=1}^n w_i \min \left\{ \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, y_i^j \right\} \\
&\geq \max_i w_i y_i^j \\
&\geq \max_i w_i \min \left\{ \sum_{j=1}^k \left[ 1 - \frac{d(c_j^*, x_i)}{r} \right]_+, y_i^j \right\} \\
&\geq \frac{1}{n} \sum_{i=1}^n w_i \min \left\{ \sum_{j=1}^k \left[ 1 - \frac{d(c_j^*, x_i)}{r} \right]_+, y_i^j \right\} \\
&= \frac{1}{n} \sum_{i=1}^n w_i \min \left\{ \sum_{j=1}^k \left[ 1 - \frac{d(c_j^*, x_i)}{r} \right]_+, 1 - h_i^{j-1} \right\} \\
&\geq \frac{1}{n} \sum_{i=1}^n w_i \left( \min \left\{ \sum_{j=1}^k \left[ 1 - \frac{d(c_j^*, x_i)}{r} \right]_+, 1 \right\} - h_i^{j-1} \right) \\
&= \frac{1}{n} \left( f_{opt} - \sum_{\ell=1}^{j-1} g(\ell - 1) \right)
\end{aligned} \tag{4.20}$$

Thus (b)  $g(j) \geq \frac{1}{n}(f_{opt} - \sum_{\ell=1}^{j-1} g(\ell - 1))$  is proved. ■

Now let  $f(j) = \sum_{\ell=1}^j g(j)$ . Then  $f(k)$  is the reward value, i.e. the sum of objective values in all  $k$  iterations, returned by the local greedy algorithm above. According to Lemma 2, we have:

$$f(1) = g(1) \geq \frac{1}{n} f_{opt}; \tag{4.21}$$

$$\begin{aligned}
f(j) &= f(j-1) + [f(j) - f(j-1)] \\
&= f(j-1) + g(j) \\
&\geq f(j-1) + \frac{1}{n} (f_{opt} - \sum_{\ell=1}^{j-1} g(\ell - 1)) \\
&= f(j-1) + \frac{1}{n} (f_{opt} - f(j-1)) \\
&= (1 - \frac{1}{n})f(j-1) + \frac{1}{n} f_{opt}, \\
&\quad \text{for } j = 2, \dots, k.
\end{aligned} \tag{4.22}$$

Combining the base case equation (4.21) and the recursive in equation (4.22), we can get:

$$f(j) \geq (1 - (1 - 1/n)^j) f_{opt}. \tag{4.23}$$

Thus:

$$f(k) \geq (1 - (1 - 1/n)^k) f_{opt} \tag{4.24}$$

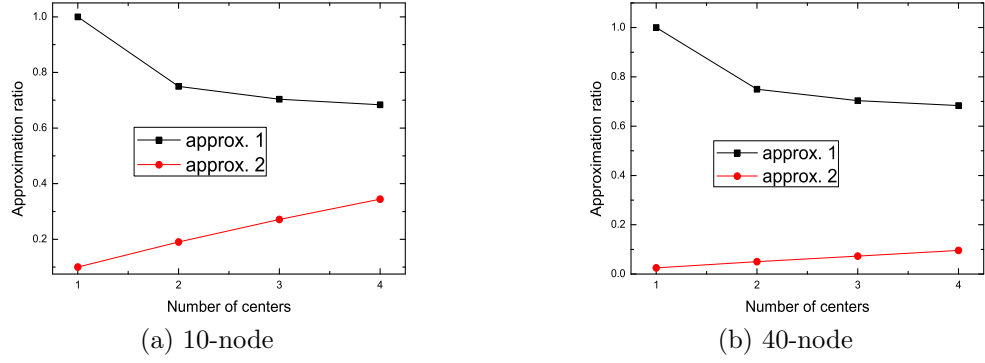


Figure 4.2: Comparison of approximation ratio in 10-node and 40-node environments.

and the approximation ratio of the local greedy algorithm is  $(1 - (1 - 1/n)^k)$ .

In each round, it takes up to  $n$  steps to select the center point in the approach, and there are up to  $n$  steps to get the coverage reward by each candidate center. Hence, the complexity of this algorithm is  $O(kn^2)$ .

### Alternative local greedy algorithms

In Fig. 4.2, approx. 1 is the approximation ratio  $1 - (1 - 1/k)^k$  from Theorem 1 using Algorithm 1, and approx. 2 in the figures is the approximation ratio  $1 - (1 - 1/n)^k$  from Theorem 2 using Algorithm 2, where  $k$  is the number of centers and  $n$  is the number of points. We compare these two approximation ratios in 10-node and 40-node environments. We can see that approx. 1 is much larger than approx. 2. In the following, we propose the other two algorithms: one reduces the asymptotic complexity (simple local greedy algorithm) and another one increases the approximation ratio (complex local greedy algorithm).

**Simple local greedy algorithm:** picks the largest single reward point as the center in each round, which is described in Algorithm 6:

---

#### Algorithm 6 Simple Local Greedy Algorithm

---

- 1: let  $y_i^1 = 1$  for  $i = 1 \dots n$ .
  - 2: **for**  $j = 1$  to  $k$  **do**
  - 3:    $c_j \leftarrow x_{i^*}$  for  $i^* = \arg \max_i w_i y_i^j$ .
  - 4:    $\{g(j), z_1^j, \dots, z_n^j\} \leftarrow$  maximizing  $\sum_{i=1}^n w_i z_i^j$  by solving Equation (4.25).
  - 5:   update  $y_i^{j+1} = y_i^j - z_i^j$ , for  $i = 1 \dots n$ .
  - 6: **end for**
-

In Algorithm 6, the optimization problem involved in the  $j$ th round is as follows:

$$\begin{aligned}
g(j) &= \sum_{i=1}^n w_i \min \left\{ \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, y_i^j \right\} \\
&= \max_{z_1^j, \dots, z_n^j} \sum_{i=1}^n w_i z_i^j \\
\text{s.t.} \quad & z_i^j \leq y_i^j, \forall i = 1, \dots, n \\
& z_i^j \leq \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, \forall i = 1, \dots, n
\end{aligned} \tag{4.25}$$

In each round, we will pick the point with the maximum *single-point reward* as the center of the disk. If there are a number of points, which have the same maximum single-point reward, our selection will be based on the index of the nodes.

**Theorem 3** *The complexity of Algorithm 6 (simple local greedy) is  $O(kn)$ .*

**Proof.** In each round, there are up to  $n$  steps to get the coverage reward by the selected center. Hence, the complexity of this algorithm is  $O(kn)$ . ■

It is easy to show that the approximation ratio in Theorem 3 still holds for the simple local greedy algorithm.

**Complex local greedy algorithm:** the complex local greedy algorithm is described in Algorithm 7:

---

**Algorithm 7** Complex Local Greedy Algorithm

---

- 1: let  $y_i^1 = 1$  for  $i = 1 \dots n$ .
  - 2: **for**  $j = 1$  to  $k$  **do**
  - 3:   **for**  $i = 1$  to  $n$  **do**
  - 4:     Initially  $x_i^1 \leftarrow x_i$ .
  - 5:     update  $x_i^{l+1} = \text{new-center}(x_i^l)$ , for  $l = 1 \dots (n-1)$ .
  - 6:   **end for**
  - 7:    $\{g(j), c_j, z_1^j, \dots, z_n^j\} \leftarrow$  maximizing  $\sum_{i=1}^n w_i z_i^j$  by solving Equation (4.26).
  - 8:   update  $y_i^{j+1} = y_i^j - z_i^j$ , for  $i = 1 \dots n$ .
  - 9: **end for**
- 

In Algorithm 7, the optimization problem involved in the  $j$ th round is as follows:

$$\begin{aligned}
g(j) &= \max_{c_j \in \{x_1^n, \dots, x_n^n\}} \sum_{i=1}^n w_i \min \left\{ \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, y_i^j \right\} \\
&= \max_{c_j \in \{x_1^n, \dots, x_n^n\}, z_1^j, \dots, z_n^j} \sum_{i=1}^n w_i z_i^j \\
\text{s.t.} \quad & z_i^j \leq y_i^j, \forall i = 1, \dots, n \\
& z_i^j \leq \left[ 1 - \frac{d(c_j, x_i)}{r} \right]_+, \forall i = 1, \dots, n
\end{aligned} \tag{4.26}$$

We use a 2-D disk to illustrate the new-center ( $x_i^l$ ) in a 2-D and 2-norm system:

1. Start with the disk  $D$  centered at an  $x_i^l$ .
2. Consider the remaining heaviest point  $j$  (i.e.,  $\max w_j z_j^l$ ).
3. If  $j$  is outside  $D$ , return the center of  $D$  and stop.
4. Otherwise, define the center for the new disk  $D'$  by including  $j$  in  $D$ . This center is the smallest disk that covers all points in  $D$  plus point  $j$ .
5. If the coverage reward of  $D'$  is larger than the coverage reward of  $D$ , then return the center of  $D'$ ; otherwise, return the center of  $D$ .

In Algorithm 7,  $g(j)$  is the coverage reward. It is so presented to keep all of the algorithms in a uniform way. The new center ( $x_i^l$ ) in 1-norm can be easily calculated through projections on different dimensions. Note that the major difference here is that the position of a center can be anywhere in the complex local greedy algorithm.

The complexity for this greedy heuristic algorithm is described in Theorem 4:

**Theorem 4** *The complexity of Algorithm 7 (complex local greedy) is  $O(kn^3)$  for 2-norm in a 2-D space. The complexity is  $O(kmn^3)$  for 1-norm in the  $m$ -D space.*

**Proof.** For 2-norm in a 2-D space, there are  $k$  rounds in the outer loop. Also, each point (of  $n$ ) performs (1), (2) and (3), that we discussed above, of the following:

Suppose the size of  $D$  is  $i$  in the current round. (2) takes  $(n - i)$  steps. (3) consumes  $(i + 1)$  steps (to find the smallest disk to cover  $D$  and the newly selected point). Up to  $n$  rounds of (2) and (3) have a total of  $n^2$  steps. Therefore, the overall complexity is  $O(kn^3)$ .

For 1-norm in the  $m$ -D space, in each round, up to  $n^2$  steps are needed as all remain the same except (3), which consumes  $m \cdot (i + 1)$ . Along each dimension, the the boundary can be determined through a projection on the dimension. The min and max values are determined. The center position along this dimension is  $(\min + \max)/2$ . Therefore, its complexity is  $O(kmn^3)$ . ■

The approximation ratio for the complex local greedy algorithm is still an open problem.

### **$p$ -norm in the $m$ -D space**

Previously, we used 2-norm and a 2-D space to explain our algorithms. The disk and its radius we mentioned are based on 2-norm to calculate the interest distance between two points in a 2-D space.

In the 2-D space, if we use the 1-norm to calculate the interest distance, we can use a square to cover the points instead of the disk, and the “radius” would then be the square’s side length.

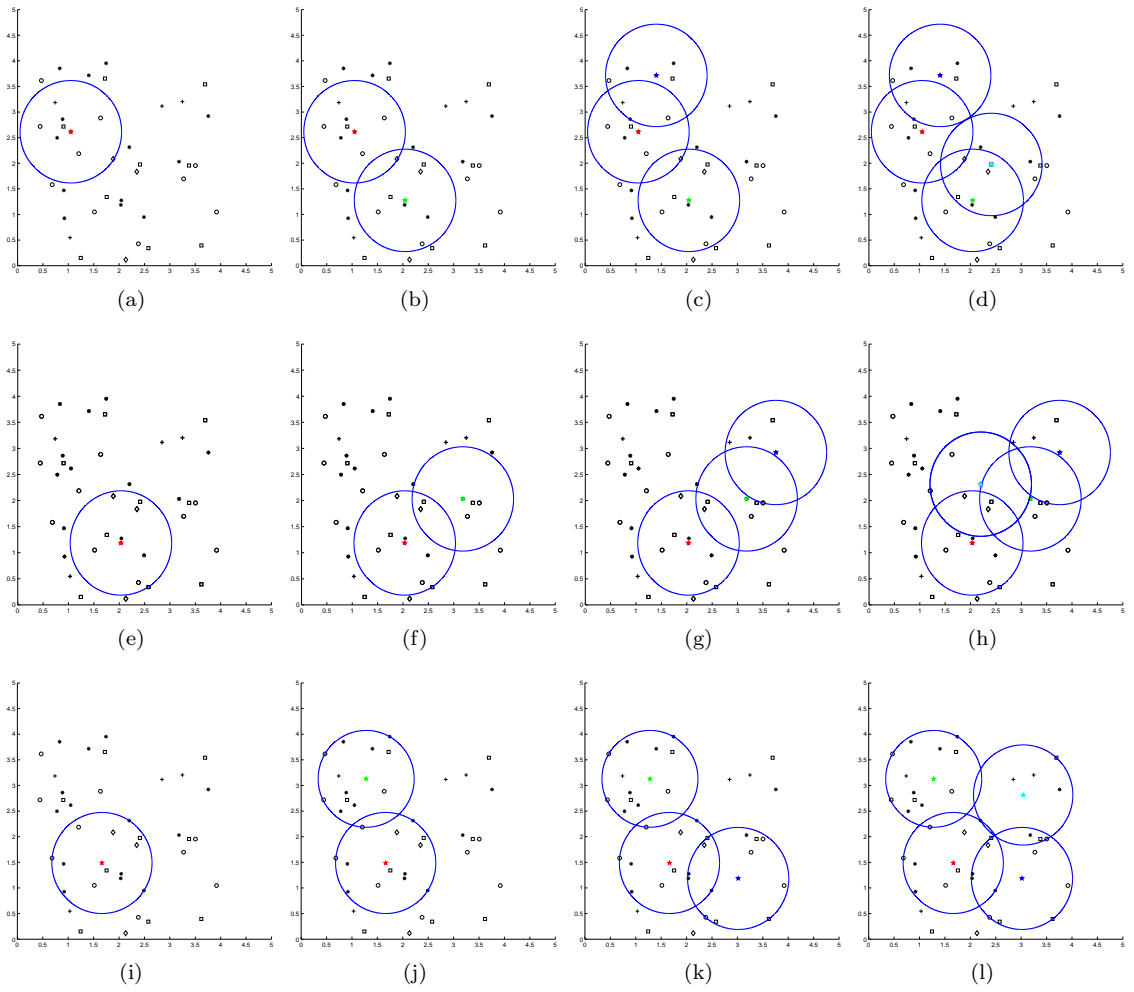


Figure 4.3: Greedy algorithms: greedy 2: (a) - (d); greedy 3: (e) - (h); greedy 4: (i) - (l) (different symbols of the points mean different weights: 5: \*, 4: □, 3: ◇, 2: +, 1: ○. \* is the centers).

Table 4.1: Comparison of three greedy heuristic algorithms.

Coverage reward	1	2	3	4	Total
Greedy 2	14.3145	11.2969	9.778	9.2406	44.6301
Greedy 3	11.2969	9.7395	8.4636	8.3435	37.8435
Greedy 4	20.3867	19.1588	13.5481	10.4635	63.5571

In the 3-D space, in a 1-norm system, we use a cube to cover the points, while in a 2-norm system, we use a ball to cover the points.

Based on the complexity of the algorithms, we rename these three greedy heuristic algorithms we discussed above. The simple local greedy algorithm is greedy heuristic algorithm 1, original greedy local algorithm is greedy heuristic algorithm 2, and the complex local greedy algorithm is greedy heuristic algorithm 3.

In Figs. 4.3. We implement our three greedy heuristic algorithms in  $4 \times 4$  2-D space with 40 points based on 2-norm for interest distance calculation. Table 4.1 shows that the coverage reward gain in each round using three greedy heuristic algorithms. It is clearly that the third one is much better than other two in each round.

### Simulation methods and setting

We randomly put the nodes into a  $(4 \times 4)$  2-D space and a  $(4 \times 4 \times 4)$  3-D space. In a 2-D space, we have 10-node and 40-node environments. In a 3-D space, we have 40-node and 160-node environments.

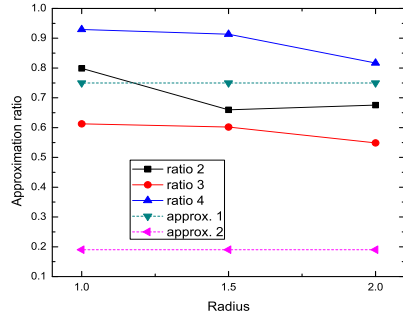
The weight of each node is an integer. We also have two schemes: one is that the weight of each node is the same, which is 1; another is that the weight of each node is different, which is a random integer between 1 and 5.

In our simulation, the calculation of the interest distance is based on 1-norm and 2-norm. We compare the greedy algorithms in different number of centers (2, 4) and different radius of the centers (1, 1.5, 2).

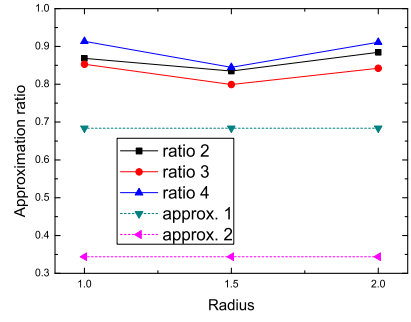
We denote that the original local greedy algorithm (Algorithm 5) is called greedy 2, the simple local greedy algorithm (Algorithm 6) is called greedy 3, and the complex local greedy algorithm (Algorithm 7) is called greedy 4.

#### 4.1.4 Simulation

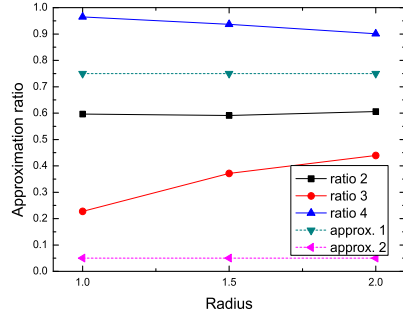
In the 2-D space comparison, ratios 2, 3, and 4 present the approximation ratio among these three greedy algorithms (Algorithms 2, 3, and 4) with the exhaustive optimal solution, respectively. We also compare these three ratios with approx. 1 from Theorem 1 and approx. 2 from Theorem 2. Note that approx. 2 is the worst case for Algorithm 2. Therefore it should correspond to the



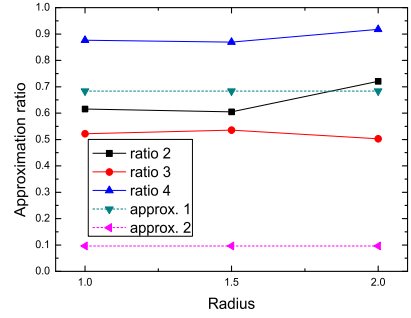
(a) 10-node with 2 centroids



(b) 10-node with 4 centroids

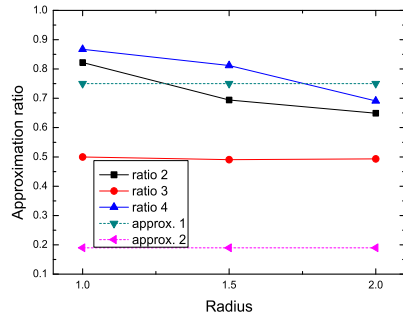


(c) 40-node with 2 centroids

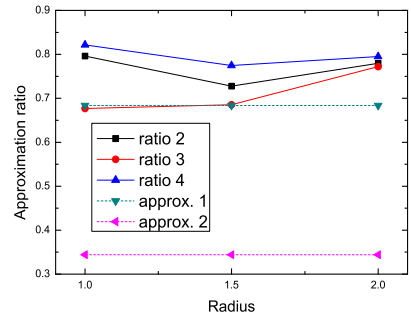


(d) 40-node with 4 centroids

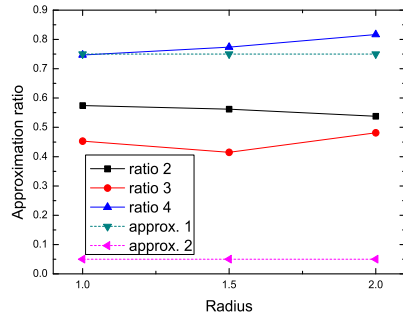
Figure 4.4: Comparison in 2-norm in a 2-D space using different weights.



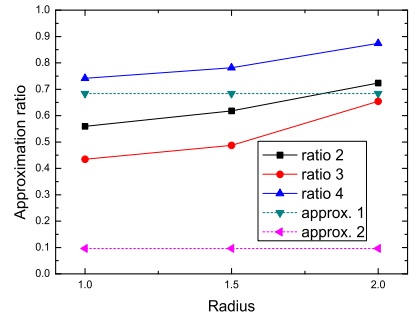
(a) 10-node with 2 centroids



(b) 10-node with 4 centroids



(c) 40-node with 2 centroids



(d) 40-node with 4 centroids

Figure 4.5: Comparison in 2-norm in a 2-D space using the same weights.

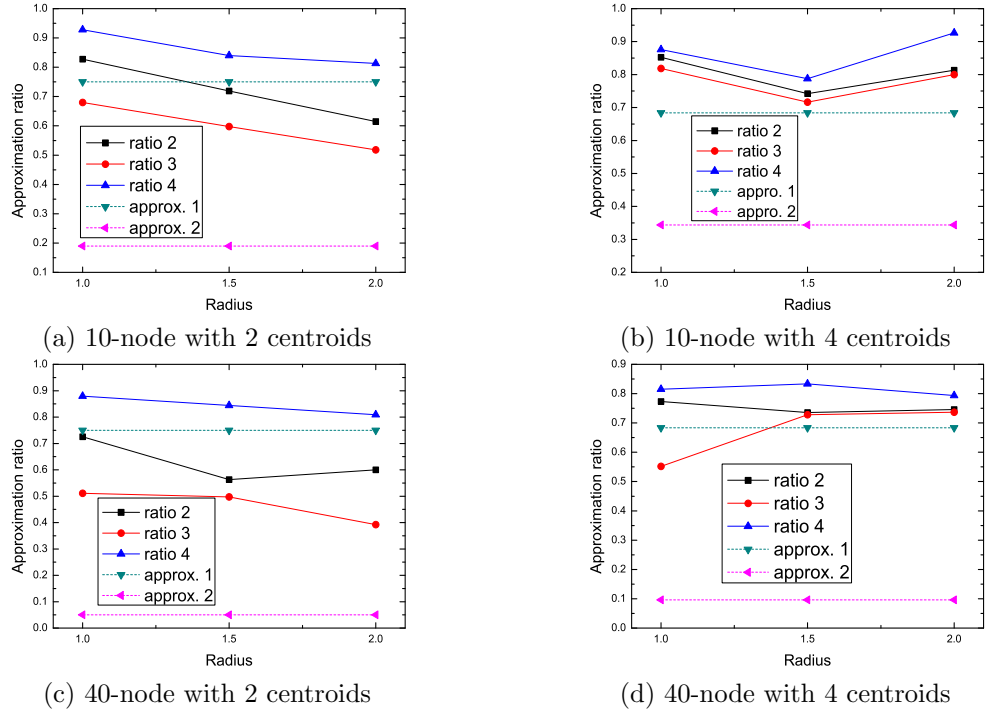


Figure 4.6: Comparison in 1-norm in a 2-D space using different weights.

smallest number (in ratio). Approx. 1 is the worst case for an iterative approach with the optimal local solution. In the 3-D space comparison, we just compare these three greedy algorithms' gained rewards.

### Results in a 2-D space using 2-norm

in a 2-D space, we first use 2-norm to calculate the interest distance between two points. We compare the approximation ratio among these three algorithms with the exhaustive solution, as shown in Figs. 4.4 and 4.5.

From Figs. 4.4 and 4.5, we can see that our proposed three local greedy algorithms' approximation ratios are all larger than approx. 2. This validates Theorem 2. Greedy 3 is better than the other two, and its approximation ratios are above approx. 1 most times in different conditions.

Overall, in a 2-D space for 2-norm, we find that with greedy 3, the approximation ratio is about 84.22% which is the best out of all three of the greedy algorithms. Greedy 1's the approximation ratio is about 68.87% and approximation ratio for greedy 2 is about 55.97%.

### Results in a 2-D space using 1-norm

In a 2-D space, we use 1-norm to calculate the interest distance between two nodes. We compare the approximation ratio among these three algorithms with the exhaustive solution.

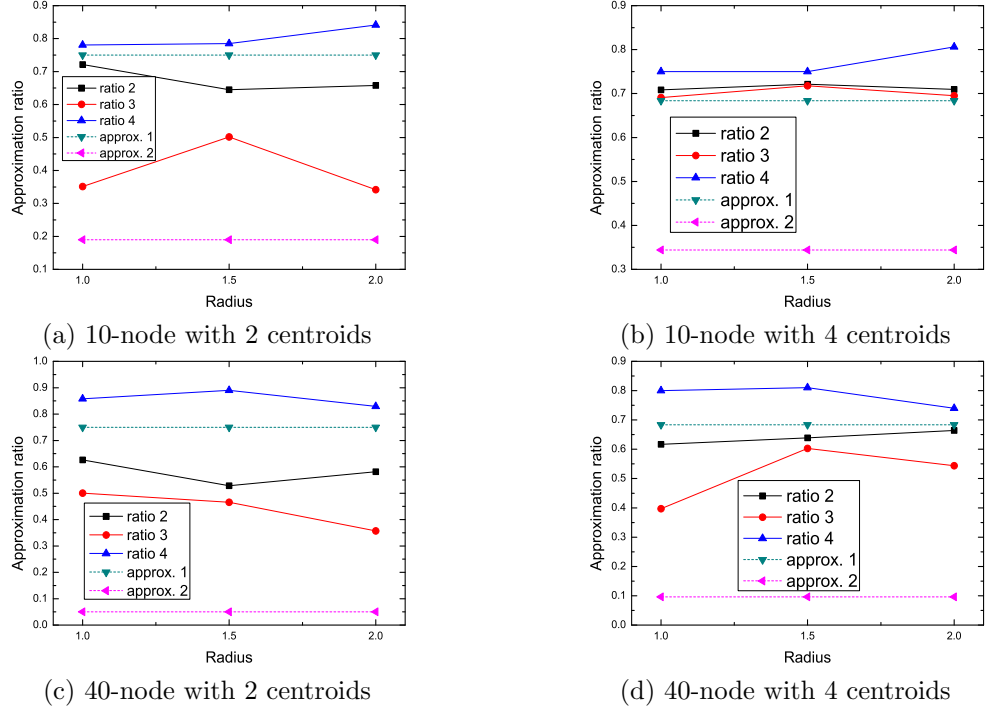


Figure 4.7: Comparison in 1-norm in a 2-D space using the same weights.

From Figs. 4.6 and 4.7, we find that with greedy 3, the approximation ratio is about 82.76% which is the best out of all three of the greedy algorithms. Greedy 1's approximation ratio is about 68.77% and the approximation ratio for greedy 2 is about 57%.

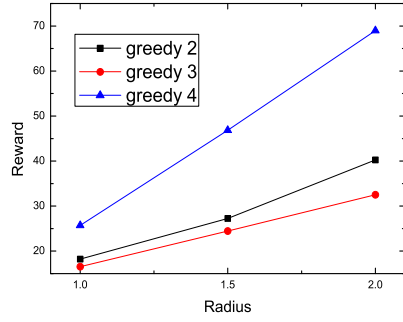
### Results in a 3-D space using 1-norm

in a 3-D space, we use 1-norm to calculate the interest distance between two nodes. We compare the total reward that these three greedy algorithms gain with same weight and different weight schemes, as shown in Figs. 4.8 and 4.9.

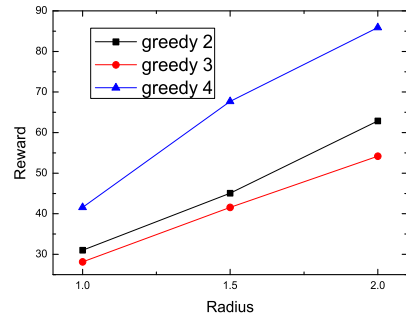
We find using greedy 3 will get the highest reward. Using greedy 1 gets about 61.04% of the reward that greedy 3 gets, and greedy 2 gets about 31.14%.

### Summary of simulation

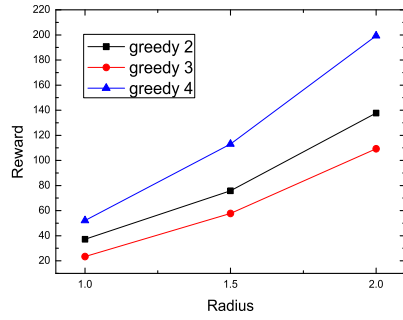
We use these three greedy algorithms to calculate the approximation value for the optimization problem. In 2-D and 3-D spaces for 1-norm and 2-norm, we can see that greedy 3 gets the best results. Its approximation ratio is above 80%, which is higher than Theorem 1's approximation ratio, which considers that each round is optimal. Greedy 1's approximation ratio is larger than 60%, while greedy 2's approximation ratio is about 56% in a 2-D space and 31% in a 3-D space. These two greedy algorithms' results also reflect the analytical results (Theorem 2). Although our optimization problem is an NP-hard problem, our proposed greedy 3 still gets an acceptable approximation ratio.



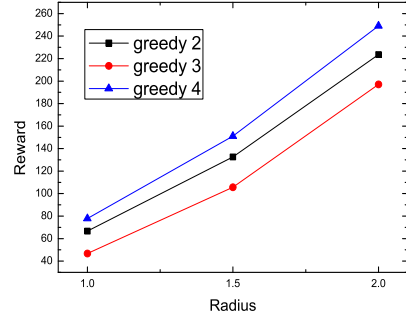
(a) 40-node with 2 centroids



(b) 40-node with 4 centroids

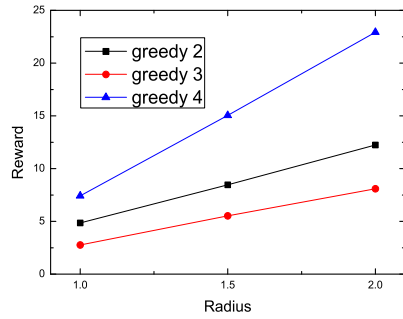


(c) 160-node with 2 centroids

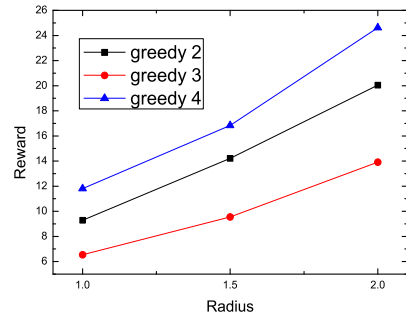


(d) 160-node with 4 centroids

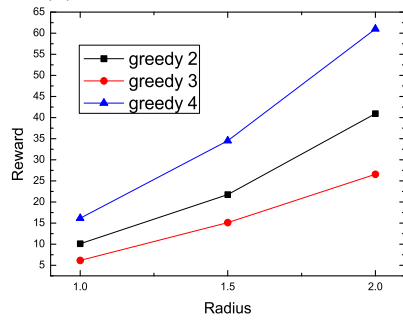
Figure 4.8: Comparison in 1-norm in a 3-D space using different weights.



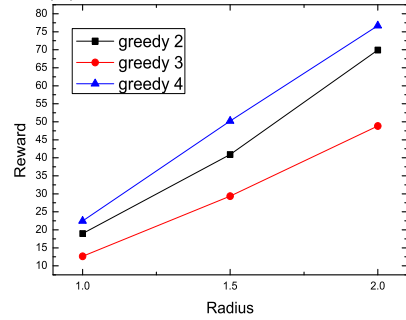
(a) 40-node with 2 centroids



(b) 40-node with 4 centroids



(c) 160-node with 2 centroids



(d) 160-node with 4 centroids

Figure 4.9: Comparison in 1-norm in a 2-D space using the same weights.

### 4.1.5 Conclusion

In this section, we studied the content distribution in wireless networks. We first formulated the problem into an optimal content distribution problem and proved it as an NP-hard problem. Three greedy algorithms have been proposed to solve the optimal problem. The greedy algorithms can be implemented in the  $m$ -D space using  $p$ -norm to calculate the interest distance. We analyzed the approximation ratio of the greedy algorithms and its complexity. We then turned to studying the performance of these three local greedy algorithms in 2-D and 3-D space with 1-norm and 2-norm. The simulation results have shown that our proposed algorithms perform well. To the best of our knowledge, this is the first study on the optimal content distribution problem in wireless networks.

## 4.2 Ticket-based Multiple Packet Broadcasting

In delay tolerant networks (DTNs), broadcasting is an important routing function that supports the distribution of data to all users in the network. Efficient broadcasting in DTNs is a challenging problem due to the lack of continuous network connectivity. In this section, we consider the limited bandwidth scenario for DTN broadcasting.

### 4.2.1 Introduction

Recent years have shown rapid growth in the popularity and capabilities of handheld devices, such as mobile phones and laptops. Delay tolerant networks (DTNs) technologies have been proposed to allow mobile nodes in such extreme networking environments to communicate with one another. In DTNs, most of the time, there does not exist an end-to-end path between some or all of the nodes in the network. Existing broadcast routing protocols in DTNs [28, 68] do not consider situations with limited bandwidth and a choice of multiple packets to transmit. In this report, we propose a novel DTN broadcasting scheme which considers multiple packet ranking where there are limited bundles of packets that can be forwarded in each contact.

In recent years, biologists have found that *Levy walks* can be used to describe the mobility patterns of foraging animals [69]. Computer scientists also paid attention to this area of research. They studied Levy walks in human mobility [55, 70], which can help us to analyze wireless mobile networks, such as DTNs. Human movements have patterns to them. For example, we go to work, meetings, favorite places, etc. These are not random movements. Recently, there has been some research done with the community-based mobility model [71, 72]. The mobile nodes tend to move and stay at local sets of frequently visited places for most of the time, while occasionally roaming to other places. Thus, nodes often meet other nodes that also move and stay within the same set

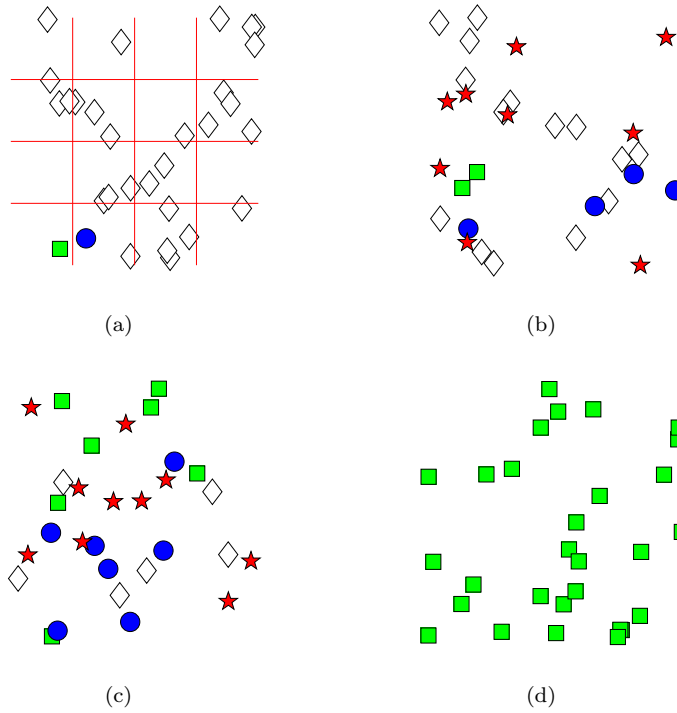


Figure 4.10: An example for multiple packet broadcasting: circle nodes with packet 1, star nodes with packet 2, square nodes with both packets, and diamond nodes with no packets.

of frequently visited places while, by chance, meeting nodes of other areas. Hence, considering Levy walks and the community-based mobility model, we divide the network into grids, and nodes in the same grid have more of a chance to meet with each other.

Broadcasting in DTNs poses some unique challenges when the bandwidth is limited for each contact (for example, one transmission per contact) while there are multiple broadcast packets in the local queue. One key issue is to determine packet ranking and its dynamic ranking adjustment during the broadcast. For example, a relatively “new” packet usually has a higher rank than an “old” packet that has been in the system for a while. Another issue is how to control packet replication during the broadcast process. Various uncertainties in DTNs, including movement and contact distribution, make the distributed ranking adjustment and replication process harder.

In DTNs, it is observed that contacts tend to be clustered in relatively short-term time scales. This clustering effort occurs either in physical space where the physical mobility pattern follows Levy walks or community-based mobility, or in logical space where people congregate in an online social network, such as Facebook and Twitter. We focus only on physical space (although it can be extensible to logical space as well) by partitioning a given 2-D space into square regions (also called grids). We first propose a *quad-grid division* scheme, which divides the network into multiple grids based on recursively dividing a grid into 4 small grids. In the proposed approach, each node maintains its activity levels of all of the grids. Initially, the source node has a given set of tickets

for all grids, usually  $k$  copies for each grid. In our simulation, the value of  $k$  is the number of nodes in the network divided by the grid number. Our protocol contains two steps: the first step is *packet selection*. When two nodes make contact, the forwarding node will select the highest priority packet based on the packet *priority value*, which is jointly decided by three parameters: number of tickets currently held (or simply “ticket”), time in the system (or “time”), and already committed hop count (or “hop-count”). The second step is *relay node selection*. When a ticket carrier  $a$  is in contact with another node  $b$ , if  $b$  has a higher (global) activity level among grids, where  $b$  currently holds tickets, than  $a$ , then  $a$  will redistribute its tickets of certain grids where  $b$  has higher (local) activity levels.

Ticket-based packet priority ranking not only considers the packets’ characteristics, but also links the global and local active level of the nodes, which is more efficient in DTN broadcasting. The time-based scheme is based on the period the packet travels in the network, which has the global information. Hence, the time-based scheme is considered more useful than the hop-count-based scheme, which only has the local information. We verify the effectiveness of our approach through synthetic and real human mobility trace simulations.

#### 4.2.2 Multiple Packet Broadcasting

Here, we will first explain the area division method. Then, we will present the two steps of our scheme and give an example to explain the whole process.

##### System Model

Due to the characteristics of DTNs and the limited bandwidth, we assume that in each contact, the forwarding node can forward limited packets to a receiving node. Assume there are  $n$  nodes in the network. Initially, there is one source node which holds  $m$  packets. The area of the network will be equally divided into  $g$  grids generated by partitioning the 2-D broadcast space along both dimensions. Each packet will be associated with  $t$  number of tickets, and each grid has  $t/g$  number of tickets. Each node  $a$  is associated with a *global active level*  $G_a$  and  $g$  number of *local active levels*  $L_a(i)$  for grid  $i$  ( $i \in (1, 2, \dots, g)$ ). Each packet will also be associated with a *priority value*  $P$  to prioritize packets and to help select the highest priority packet to be forwarded.

Fig. 4.10 is an example of our protocol. In Fig. 4.10(a), the square node is the source node, which takes two packets initially. When it encounters a neighbor node, it will decide which packet will be forwarded. Circle nodes represent nodes that hold packet 1, star nodes are nodes that hold packet 2, and diamond nodes represent nodes that have no packets. After a while, both packets will be broadcasted to all of the nodes, as shown in Fig. 4.10(d).

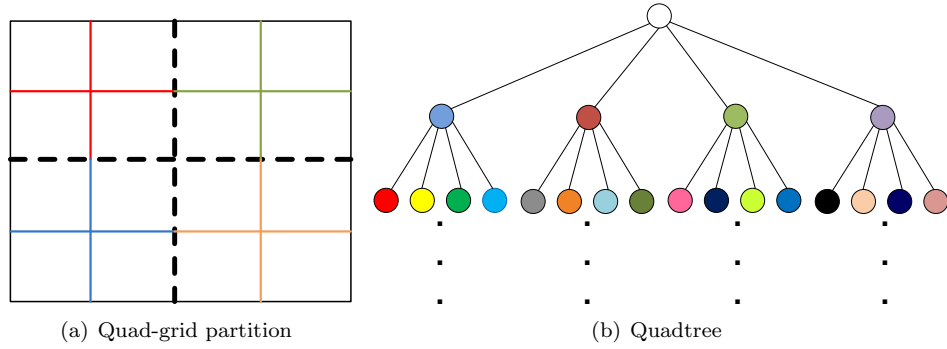


Figure 4.11: Area division.

### Quad-grid Division and Ticket Distribution in 2-D Space

Given a 2-D space network, we first divide the area into multiple grids. As shown in Figs. 4.11 and 4.10(a), the area will be recursively divided into 4 small grids. Each node belongs to only one grid. This idea can use a quadtree to be explained as Fig. 4.11(b). Tickets are assigned to grids,  $k$  copies for each.  $k$  depends on the number of nodes in the 2-D space and the number of grids.

As the recursion gets deeper, the number of tickets also increases. To reduce overhead in piggybacking these tickets, we can judiciously control the depth of the partition to balance cost (in maintaining tickets) and efficiency (in maintaining a certain level of ticket granularity).

### Packet Selection

The first step of the proposed scheme is packet selection. When node  $a$ , with  $m$  number of packets, has a contact with node  $b$ , node  $a$  will select the highest priority packet to forward. There are three parameters affecting the priority of the packet. (1) *Tickets ( $C$ )*: the number of tickets this packet holds. The more tickets the packet holds, the fewer the nodes in the network already received this packet. Hence, the packet with more tickets will have higher priority. (2) *Time ( $T$ )*: the time that this packet traveled in the network. If a packet has been traveling in the network for a long amount of time, it is likely that it has already been forwarded to other nodes. Hence, we propose *first in first out* (FIFO) for packet selection based on the parameter *time* in our simulation. (3) *Hop-count ( $H$ )*: the number of times the packet has been forwarded. The larger the hop-count the packet has, the more of a chance other nodes have to already be covered. Thus, we assign the packet which has a larger hop-count and a lower priority.

### Relay Node Selection

The second step is to select a good relay node. A good relay node is the node that can cover more grids and has a higher frequency of contact with other nodes in the network.

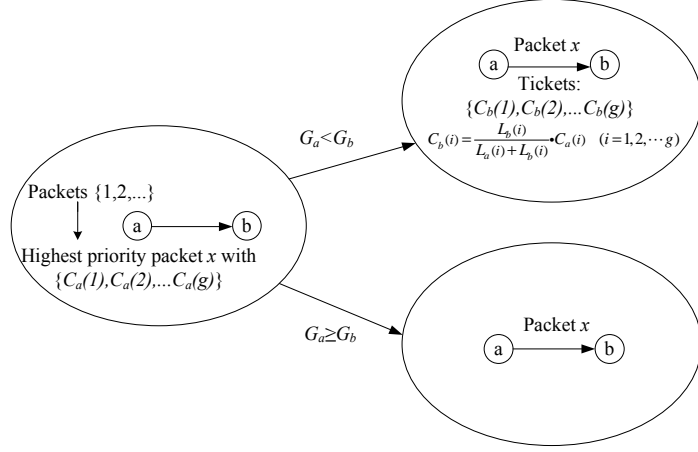


Figure 4.12: Ticket-based multiple packet DTN broadcasting.

There are two parameters to measure the node's value. (1) Global active level ( $G$ ): a priori knowledge or estimation of the total number of contacts within the network in a given period; (2) Local active level ( $L$ ): a priori knowledge or estimation of the number of contacts this node has with other nodes in this grid in a given period.

In our simulation, the given period to calculate the active level is the length of the whole period before the current contact. Hence, the active level is based on the historical information. The highest priority packet selected in the previous step will be forwarded to node  $b$ . Whether node  $b$  has the ability to forward this packet is based on the tickets. The ticket partition strategy is explained in the following.

First, node  $a$  and node  $b$  will exchange their global active level  $G_a$  and  $G_b$ . Node  $a$  only forwards the tickets to node  $b$  with a higher global active level ( $G_b$ ) than its own ( $G_a$ ). This approach does not need global knowledge. Each node decides whether it should or should not forward the message by itself. This is suitable for a distributed environment such as DTNs. In addition, node  $a$  will raise its own level to the higher level of node  $b$ . This idea is based on *delegation forwarding* [49], which means the packet holder will just assign the tickets to the relay node which has the highest global active level it has ever seen.

Then, we use the local active levels to decide how many tickets for each grid should be assigned to the receiving node.

$$C_b(i) = \frac{L_b(i)}{L_a(i) + L_b(i)} \cdot C_a(i) \quad (i = 1, 2, \dots, g),$$

where  $C_a(i)$  is the number of tickets for the selected packet  $x$ , held by node  $a$  for grid  $i$ , and  $L_a(i)$  is the local active level of node  $a$  for grid  $i$ . At the same time, we will leave  $\frac{L_a(i)}{L_a(i) + L_b(i)} \cdot C_a(i)$  number of tickets for grid  $i$  in node  $a$ .

The whole process can be explained by Algorithm 8. Fig. 4.12 illustrates our entire solution.

---

**Algorithm 8** Multiple packet broadcasting in DTNs

---

- 1: When node  $a$  with global active level  $G_a$  encounters node  $b$  with  $G_b$ .
  - 2: Local active level for grid  $i$ , ( $i = 1, 2, \dots, g$ ),  $L_a(i)$  and  $L_b(i)$ .
  - 3: Node  $a$  selects the highest priority packet  $x$  with  $C_a(i)$  tickets to be forwarded.
  - 4: **if**  $G_a < G_b$  **then**
  - 5:   Node  $b$  will be the relay for packet  $x$ .
  - 6:    $G_a \leftarrow G_b$
  - 7:    $C_b(i) : \frac{L_b(i)}{L_a(i)+L_b(i)} \cdot C_a(i)$  for node  $b$
  - 8:    $C'_a(i) : \frac{L_a(i)}{L_a(i)+L_b(i)} \cdot C_a(i)$  for node  $a$
  - 9: **else**
  - 10:   Node  $b$  will just receive packets without tickets
  - 11: **end if**
- 

When node  $a$  and node  $b$  have a contact, they will first exchange their holding packet lists. Then, node  $a$  will sort the packets which are not in node  $b$ 's list based on the tickets that the packet is holding. Packet  $x$  is the highest priority packet that can be chosen to be forwarded. After that, these two nodes will compare their global active levels. If  $G_a < G_b$ , node  $a$  will duplicate packet  $x$  with some tickets, which will be based on the local active level, and be forwarded to node  $b$ . Otherwise, node  $b$  will just receive packet  $x$  without any tickets, which means node  $b$  cannot forward the packet to other nodes later.

### 4.2.3 Simulation

The metrics, which are calculated in our simulation, are *average latency* and *useless contacts*.

1. *Average latency*: the average duration of multiple packets between the time that they enter the network and the time that they finish broadcasting to all of the nodes within the network.

2. *Useless contacts*: the number of contacts that have no packet forwarded when two nodes have a contact.

#### Simulation Methods and Setting

We used synthetic traces with GPS information for the nodes' mobility. Based on the GPS information of each node, we assign each node to one grid. We use Levy walks and the community-based mobility model as these nodes' mobility patterns. Each time two nodes make contact with each other, we give a contact time and a GPS address. In the Levy walks mobility pattern, we use TLW MATLAB [56] to generate the human mobility model. In the community-based mobility model, we use the movement patterns generator [73], which is implemented by the University of Cambridge's computer laboratory, to generate the nodes' mobility patterns.

For the average latency comparison, we set up a 100-node environment. The initial number of

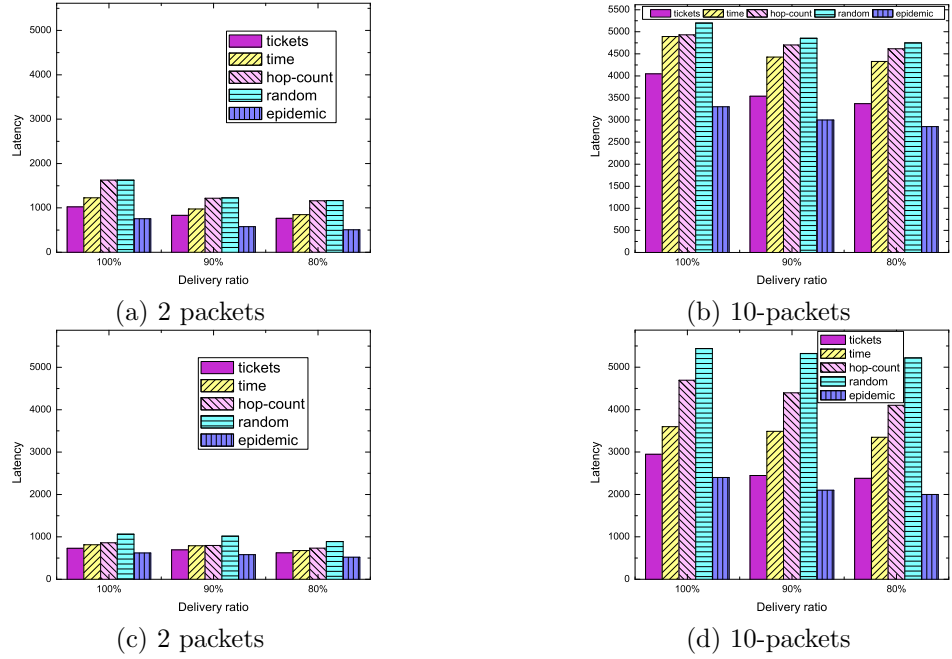


Figure 4.13: Average latency comparisons in Levy walks model ((a) and (b)) and community-based mobility model ((c) and (d)).

packets in the source node will be set to 2 and 10. We also set up the delivery ratio <sup>1</sup> as 100%, 90%, and 80%. We compare the performance of the schemes using the tickets, time, hop-count, or random <sup>2</sup>, as a primary key for packet priority ranking.

For the useless contacts comparison, we also set up a 100-node environment. The initial number of packets in the source node are set to 2, 4, 6, 8, and 10 respectively. We will compare the useless contacts using tickets, time, or hop-count for packet priority ranking. In the relay node selection step, we will also compare our ticket-based scheme with the no ticket assignment scheme, when the packets are ordering during the whole process.

We use NCSU’s human mobility traces [74] in our simulation. These traces are collected human mobility traces from five different sites - two university campuses (NCSU and KAIST), New York City, Disney World (Orlando), and the North Carolina state fair. We use the NCSU campus trace and the North Carolina state fair trace to evaluate our schemes.

For the average latency comparison, the initial number of packets in the source node will be set to 2 and 4. <sup>3</sup> For the NCSU campus trace, we set up the delivery ratio as 85%, 80%, and 75% <sup>4</sup>. For the North Carolina state fair trace, we set up the delivery ratio as 100%, 90%, and 80%.

For the useless contacts comparison, the initial number of packets in the source node are set to 2, 3, and 4 respectively. We will compare the useless contacts using tickets, time, or hop-count

<sup>1</sup>Delivery ratio: the rate of the packets received by all the nodes in the network.

<sup>2</sup>When there are multiple packets in the node’s list, we will randomly pick one to forward.

<sup>3</sup>Because of the limited contact times, a large number of packets will reduce the delivery ratio.

<sup>4</sup>In the NCSU campus trace, it is hard to reach 100% delivery ratio.

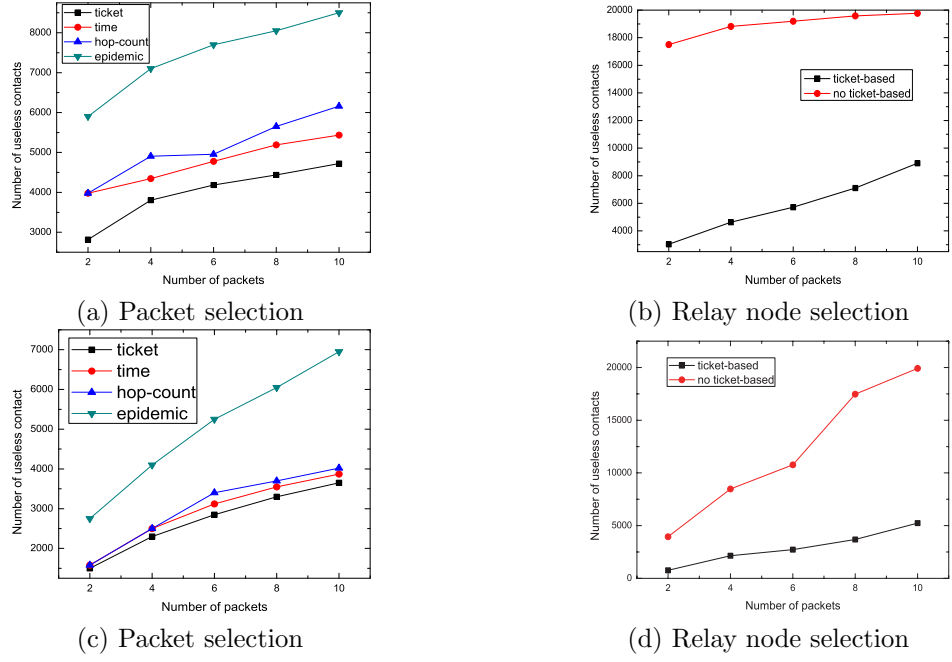


Figure 4.14: Useless contacts comparison in synthetic traces: Levy walks model ((a) and (b)) and community-based mobility model ((c) and (d)).

for packet priority ranking and epidemic routing scheme. In the relay node selection step, we will also compare our ticket-based scheme with the no ticket assignment scheme when the priority of the packets is not changing during the whole process.

## Simulation Results for Synthetic Traces

### Average latency comparison

We compare the latency in 4 parameters: tickets, time, hop-count, or random, to prioritize packets, as shown in Fig. 4.13. The results of the epidemic routing scheme are also including in these figures. Both Levy walks and the community-based mobility models show that the ticket-based scheme has the shortest latency out of all three delivery ratios. The schemes that control the packet priority ranking are all better than the random scheme.

In the Levy walks model, the ticket-based scheme has about 16.7% shorter latency than the time-based schemes, as shown in Figs. 4.13(a) and 4.13(b). In the community-based mobility model, the ticket-based scheme saves 22% of time compared with the time-based scheme, as shown in Figs. 4.13(c) and 4.13(d). That’s because the ticket-based scheme not only has the global information about the packets, but also considers the mobility patterns, which we think is Levy walks or community-based mobility in this report. Hence, ticket-based schemes have the best performance.

In both Levy walks and community-based mobility models, the epidemic routing scheme can reduce the latency by about 20% compared with ticket-based scheme in both 2 and 10 packets cases,

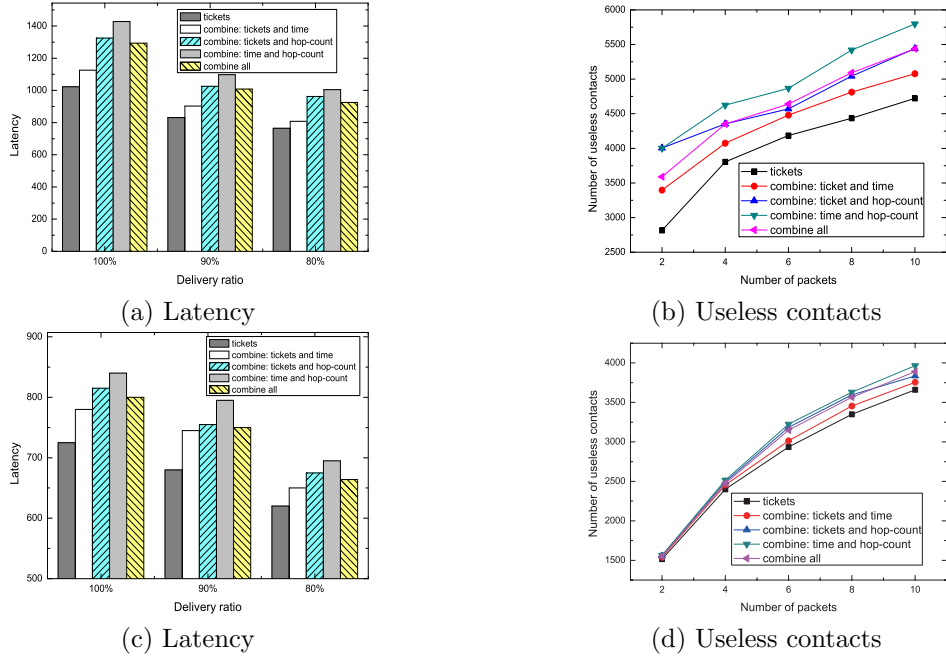


Figure 4.15: Comparison of ticket-based scheme and combined scheme in synthetic traces: Levy walks model ((a) and (b)) and community-based mobility model ((c) and (d)).

as shown in in Fig. 4.13.

We also want to see the performance of the scheme that combines tickets, time, and hop-count together to calculate the packet priority. In the combined schemes, we use the equation below to calculate the packet priority  $P$ :

$$P = \alpha \cdot C - \beta \cdot T - \gamma \cdot H, \quad (4.27)$$

where  $C$  is the number of tickets,  $T$  is the time, and  $H$  is the hop-count.  $\alpha$ ,  $\beta$ , and  $\gamma$  are constant parameters combining weight and scaling factors, and  $\alpha + \beta + \gamma$  should be 1. For example, when using ticket and time as a combined scheme,  $\alpha = \beta = 1/2$ ,  $\gamma = 0$ , and the combined three parameters,  $\alpha = \beta = \gamma = 1/3$ .

We compare our ticket-based scheme with 4 types of combined schemes, as shown in Figs. 4.15(a) and 4.15(c). We can see that the combined schemes cannot improve the performance in synthetic mobility models. Hence, we do not discuss combined schemes further.

#### Useless contacts comparison

In Figs. 4.14(a) and 4.14(c), when comparing three parameters for packet priority ranking, we can see that using tickets as the primary key is better than considering time or hop-count. Our ticket-based scheme has a smaller number of useless contacts when there are more packets in the source node initially. When the density of the network is higher, our ticket-based scheme performs

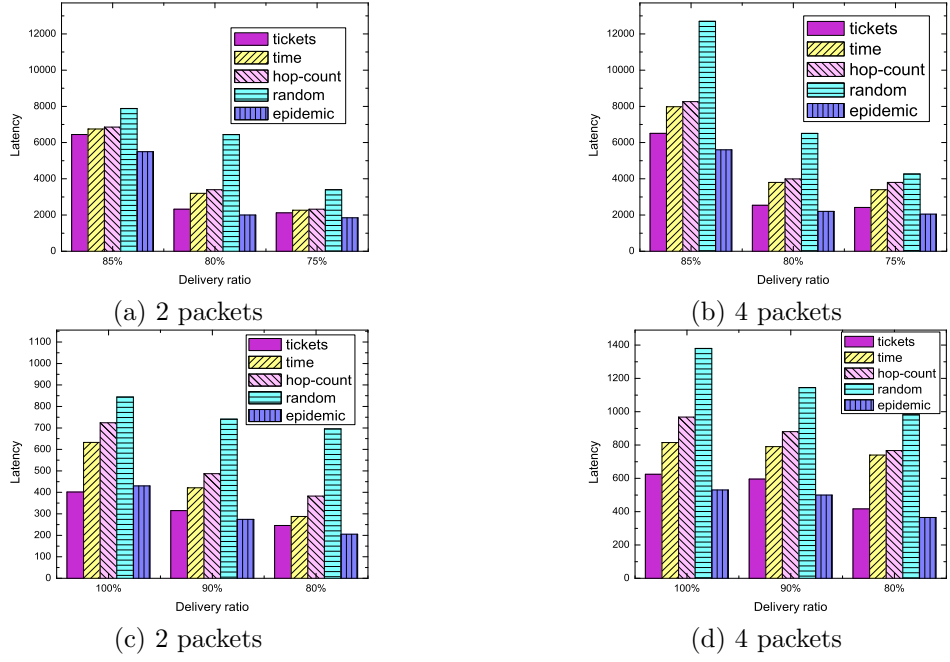


Figure 4.16: Average latency comparisons in the NCSU trace ((a) and (b)) and the North Carolina state fair trace ((c) and (d)).

better. In the Levy walks model, the ticket-based scheme decreases useless contacts by about 40%, shown in Fig. 4.14(a). In Fig. 4.14(c), the ticket-based scheme reduces useless contacts by about 35%. In Figs. 4.14(a) and 4.14(c), we find that epidemic routing generates the number of useless contacts dramatically compared with the ticket-based scheme by about 100%, both in Levy walks and community-based mobility models.

We compare the ticket-based scheme with the no ticket-based scheme in relay node selection, as shown in Figs. 4.14(b) and 4.14(d). Obviously, the ticket-based scheme has a much smaller number of useless contacts in all conditions.

We also use Equation 4.27 to compare our ticket-based scheme with the combined schemes, both in the Levy walks and the community-based mobility models. We find that the combined schemes cannot reduce the useless contacts, as shown in Figs. 4.15(b) and 4.15(d). Hence, we will not compare the combined schemes with our schemes in this section.

## Simulation Results for Real Traces

### Average latency comparison

We compare the latency in 4 parameters: tickets, time, hop-count, or random, as a primary key for packet priority ranking, as shown in Fig. 4.16. The results of the epidemic routing scheme also show in Fig. 4.16. Fig. 4.16 shows that the ticket-based scheme has the shortest latency out of all three delivery ratios. The ticket-based scheme reduces the latency by almost half compared to the

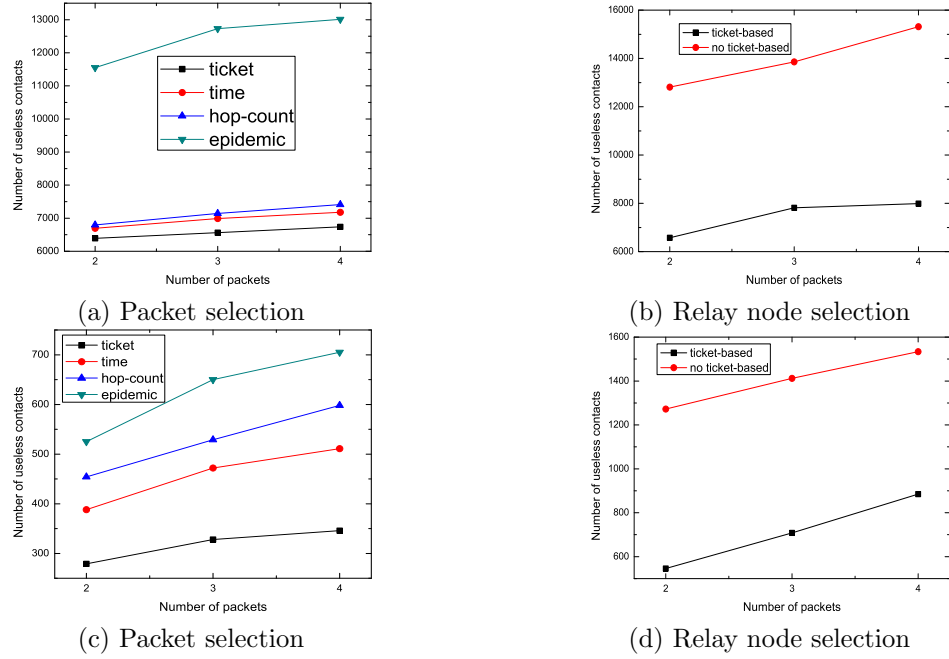


Figure 4.17: Useless contacts comparison in real traces: the NCSU trace ((a) and (b)) and the North Carolina state fair trace ((c) and (d)).

random scheme. The schemes using some parameters to control the packet priority ranking are all better than the random scheme. Both the NCSU trace and the North Carolina state fair trace show that the ticket-based scheme will slightly increase the latency compared with the epidemic scheme.

The ticket-based schemes have about 30% shorter latency than the time-based schemes, as shown in Fig. 4.16. We also find that our ticket-based scheme performs better when there are more packets initially. That's because more packets need more of an efficient packet priority ranking scheme.

We also use Equation 4.27 as the combined schemes to sort the packets. From Figs. 4.18(a) and 4.18(c), we find that our ticket-based scheme is better than the combined schemes. Hence, we do not use the combined schemes to compare the performance in real traces.

### Useless contacts comparison

In Figs. 4.17(a) and 4.17(c), when comparing three parameters for packet priority ranking, we can see that using tickets as the primary key is better than considering time or hop-count. In Fig. 4.17(a), the time-based scheme gives a similar performance when compared to the hop-count-based scheme, and the ticket-based scheme reduces useless contacts by about 16.5% when compared to the other two schemes. In Fig. 4.17(c), in the North Carolina state fair trace, the ticket-based scheme has a 45.5% smaller number of useless contacts than the other two schemes. Our ticket-based scheme has a smaller number of useless contacts when there is a larger number of packets in the source node initially. Using epidemic routing will generate the number of useless contacts dramatically in real traces.

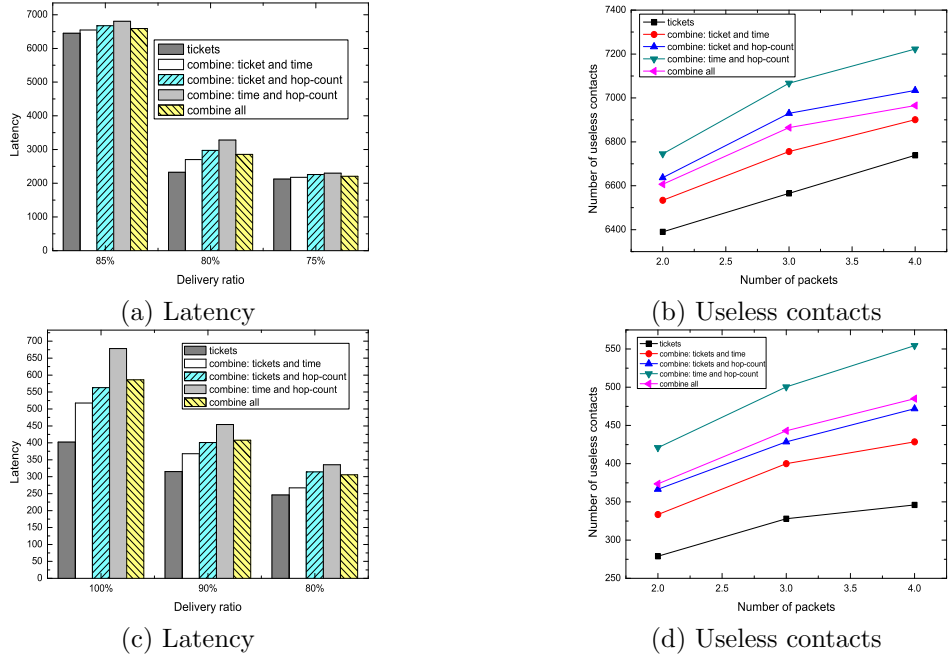


Figure 4.18: Comparison of the ticket-based scheme and the combined schemes in real traces: the NCSU trace ((a) and (b)) and the North Carolina state fair trace ((c) and (d)).

We compare the ticket-based scheme with the no ticket-based scheme in relay node selection, as shown in Figs. 4.17(b) and 4.17(d). Obviously, our scheme has a much smaller number of useless contacts under all conditions.

Using Equation 4.27 as the combined schemes to compare with our ticket-based scheme, we find that the combined schemes cannot improve the performance, as in Figs. 4.18(b) and 4.18(d). Hence, we will not use the combined schemes in the useless contacts comparison.

### Summary of Simulation

Although epidemic routing can slightly reduce the latency compared with the ticket-based scheme, it generates the number of useless contacts dramatically. In both synthetic and real traces, our ticket-based scheme performs well not only in the packet selection stage, but also in the relay node selection stage. The ticket-based scheme can reduce the latency, while at the same time reduce the number of useless contacts, which can have an impact on reducing the broadcasting cost. We also find that our ticket-based scheme performs better when there are more packets initially. That is because more packets require a more efficient packet priority ranking scheme. The ticket-based scheme not only has the global information about the packets, but also considers the mobility patterns. Hence, the ticket-based schemes have the best performance. In the packet selection stage, using time as the primary parameter is better than using hop-count, to control the packet priority. The time-based scheme has global information due to us knowing the total time that the packet has traveled in the

network, while the hop-count-based scheme just knows the neighbors' information. Hence, using the time-based scheme will have shorter latency and fewer useless contacts compared to the hop-count-based scheme. Future research can benefit from our results by developing specific applications based on our proposed schemes in DTN broadcasting.

#### 4.2.4 Conclusion

In this section, we focused on developing a ticket-based multiple packet broadcasting scheme in DTNs. Based on the human mobility pattern, which is proven to follow Levy walks or community-based mobility models in recent research, we use the quad-grid division scheme to divide the whole network into small grids, which can help the message to be forwarded quickly in the same grid. Our ticket-based scheme has two steps: packet selection and relay node selection, with the objective of reaching all nodes in the network quickly while minimizing the total number of useless contacts. We proposed the use of the number of tickets the node currently holds to decide the packet priority value. In the relay node selection step, we use nodes' global active level and local active level to determine whether the tickets of the selection packet should be assigned to the next relay node. We compared our ticket-based scheme with the time-based and hop-count-based schemes. Synthetic and real trace-driven results showed that our ticket-based scheme has the best performance, resulting in the shortest latency and the smallest number of useless contacts. We believe that the results obtained from this thesis present the first step in exploiting the ticket-based scheme in DTN broadcasting.

### 4.3 Social-Tie-Based Information Dissemination

In this section, we propose a distributed social tie strength calculation mechanism to identify the relationship between each set of pairwise mobile nodes. Following arguments originally proposed by Mark Granovetter's seminal 1973 paper [75], *The Strength of Weak Ties*, the majority of the novel information dissemination is generated by weak ties.

We first evaluate the strength of weak ties in *MIT reality mining data*. Then, a social-tie-based information dissemination protocol is presented, which is a token-based information dissemination scheme, including two phases: *weak tie-driven forwarding* and *strong tie-driven forwarding*. In the weak tie-driven forwarding phase, the susceptible nodes with more weak ties will receive more tokens for future forwarding. The number of forwarding tokens is related to the number of weak ties of two encountered nodes. After a while, the information will have been spread to multiple communities. Our scheme switches to a strong tie driven forwarding phase, in which the influential nodes are more important. The number of forwarding tokens is proportional to the number of strong ties of two

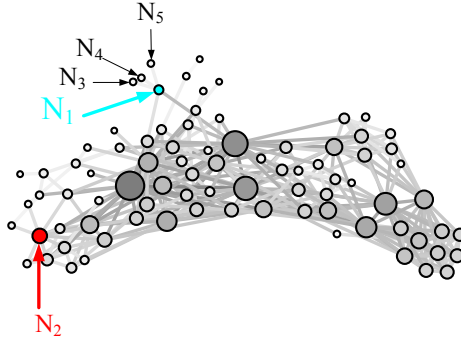


Figure 4.19: An illustration of the Facebook friends network: the nodes are the Facebook users. The size of the nodes show the degree of the nodes. Larger nodes have larger degrees. The thickness of the links show the social tie strength between pairwise nodes. Thicker links have larger tie strength.

encountered nodes.

### 4.3.1 Introduction

Social influence is empirically elusive in the social sciences. Scholars from different fields as diverse as business, computer science, physics, and sociology are interested in who influences whom, how to efficiently disseminate information, and how to prevent virus contagion. The answers to these questions, which are critical to policies, depend on the robustness of estimations of the degree to which contagion is at work during the social information influence [76].

With the popularization of smart phones, mobile opportunistic social networks (MOSNs), a new type of DTN, have recently become popular. In MOSNs, the individuals carrying smart phones walk around and communicate with each other via Bluetooth or WiFi, when they are in each other's transmission range. Because of the short contact duration and intermittent connectivity, designing an efficient information dissemination policy in such a network environment becomes a challenging problem.

There has also been some work on information dissemination in DTNs [29, 31, 32]. Existing work focuses on two directions: (1) interest-driven broadcast [29, 31], in which the information dissemination is based on the interest of the users; (2) centrality-driven data dissemination [32], which introduced a social centrality metric for routing guidance. In this report, we propose a social-tie-based information dissemination scheme in MOSNs, in which we leverage the social tie strength between each set of pairwise users, and thus ensure effective relay selections.

One of the powerful roles that networks play is to bridge the local and global, which guide the information flows through a social network. The strength of the weak ties hypothesis from sociology [75], illustrates the importance of weak ties in information dissemination. The tie strength has been modeled in many online social network researches [77–79]. Fig. 4.19 illustrates a Facebook

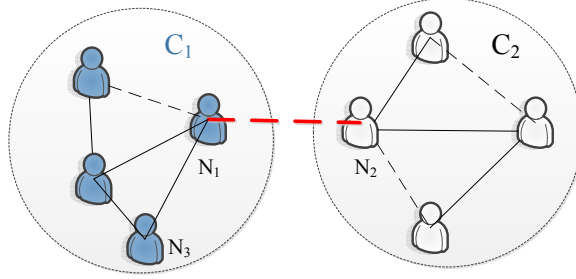


Figure 4.20: An illustration of the local bridge linking two communities: *solid* lines are the *strong* ties and dashed lines are the weak ties.

friends network.  $N_2$  has a larger degree than  $N_1$ , while  $N_1$  has more weak ties than  $N_2$ . Without  $N_1$ , three individuals  $N_3$ ,  $N_4$ , and  $N_5$  cannot receive the information, while the neighbors of  $N_2$  do not have this problem. Therefore,  $N_1$  is considered more important than  $N_2$  in information dissemination. Weak ties play an important role in information dissemination. However, these studies need the global information of whole social networks, which is not suitable in MOSNs. In this section, we propose a distributed tie strength measurement mechanism. Every mobile node maintains a *tie strength table*, which records the social tie relationship with its encountered nodes.

One particularly controversial argument is the “influentials” hypothesis that influential individuals (with many strong ties) catalyze the information dissemination in society [80]. Despite this popular argument, a variety of researches suggest that susceptibility (with many weak ties) is the key trait that drives the diffusion of novel social information [81–83]. In this thesis, we make use of the social tie table to identify the *influential* and *susceptible* members in MOSNs, as to enhance the efficiency of information dissemination.

In [84], Easley and Kleinberg claimed that *If a node A in a network satisfies the Strong Triadic Closure Property<sup>5</sup> and is involved in at least two strong ties, then any local bridge<sup>6</sup> it is involved in must be a weak tie*. As shown in Fig. 4.20, if the link between  $N_1$  and  $N_2$  is a strong tie, following the Strong Triadic Closure, there must be a link between  $N_2$  and  $N_3$ ; however, the definition of the local bridge says it cannot. If the local bridge is a strong tie, then all other links connected to the endpoints of the local bridge must be weak ties based on the Strong Triadic Closure Property. Hence, in a large network, the local bridges are more likely to be weak ties. If a node has many weak ties, the likelihood of one being a local bridge is relatively high. Therefore, to search for a local bridge, which can connect multiple communities, the focus should be on searching the susceptible nodes with many weak ties.

In this section, we consider the information dissemination problem as a token-based broadcast

<sup>5</sup>If a node  $A$  has links to nodes  $B$  and  $C$ , then the  $B - C$  link is especially likely to form if  $A$ 's links to  $B$  and  $C$  are both strong ties [84].

<sup>6</sup>A link joining two nodes in a graph is a *local bridge*, if its endpoints have no friends in common [84].

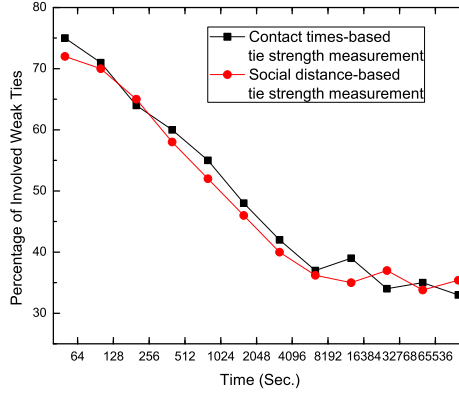


Figure 4.21: The percentage of involved weak ties in MIT reality mining data.

routing problem. In order to reduce the cost, only the mobile nodes with the tokens can forward the message to the encountered nodes. The routing process we proposed has two phases: *weak tie-driven forwarding* and *strong tie-driven forwarding*.

In the weak tie-driven forwarding phase, we propose to spread the message to more susceptible nodes, which is like an *inter-community* information spread. Hence, forwarding more tokens to the nodes with more weak ties can increase the spread speed. Therefore, at the beginning of information dissemination, the number of tokens assigned to the relay node is proportional to the number of weak ties of the two encountered nodes.

This idea is motivated from the strength of weak ties [75], that at the beginning of the information influence process, the relay nodes with more weak tie *acquaintances* have more chances to spread the information to different communities. The susceptible individuals (with many weak tie neighbors) play key roles in the novel information diffusion [85].

After a while, the information has been propagated to multiple communities. Hence, the strong ties will play a more important role, which means influential individuals (with many strong ties) can disseminate the information to many individuals in a short period. Therefore, the next stage is a strong-tie driven forwarding, which is like an *intra-community* information dissemination.

### 4.3.2 Social-Tie-Based Information Dissemination

Here, we first introduce the motivation of our work. The datasets we used will be presented next. Then, we discuss our proposed tie strength calculation mechanism and two phase token-based information dissemination protocol.

Table 4.2: Social features in datasets.

Infocom Feature	MIT Feature
<i>Affiliation</i>	<i>Neighborhood</i>
<i>City</i>	<i>Daily commute</i>
<i>Nationality</i>	<i>Hangouts</i>
<i>Language</i>	<i>Working hour</i>
<i>Country</i>	<i>Affiliation</i>
<i>Position</i>	<i>Research group</i>

## Motivation

Mark Granovetter’s seminal 1973 paper [75], *The Strength of Weak Ties*, demonstrated that weak ties play a key role in the novel information dissemination. We verify the strength of weak ties in real mobile datasets. Fig. 4.21 shows the percentage of involved weak ties in MIT reality mining data by using message flooding. We use the number of contacts or social feature distance to identify the weak and strong ties, which we will discuss the details later. We find that, at the early stage, more weak ties are involved in the information dissemination than the strong ties. Later on, strong ties become the dominant factor.

## Datasets

For our work, we exploit two datasets – MIT reality mining data [86] and Infocom 2006 conference trace [61]. These two datasets include *activity-based* and *survey-based* data. The activity-based data includes the contact information between pairwise nodes. We built an activity-based network whereby participants act as mobile nodes, and the number of contacts between two nodes act as contact information for tie strength calculation. In the MIT reality mining data, the contact information is recorded by the *call logs* by phone, and the *proximity data* by Bluetooth. The Infocom 2006 trace only includes the proximity data. The data from a survey provides self-reported personality, which we consider to be the social features of the participants.

## Tie Strength Calculation

In the MOSNs, there are many factors that affect the tie strength, such as the number of contacts, contact frequencies, contact durations, last contact time, social distance, and so on, between two encountered nodes.

**Definition:** *The strength of a tie is a (probably linear) combination of the amount of time, the emotional intensity, the intimacy (mutual confiding), and the reciprocal services which characterize the tie [75].*

In this report, we use two factors to measure the tie strength: *contact information*, which is from

the activity-based data; and *social information*, which is from survey-based data. For the contact information, we use the number of contacts between pairwise nodes as the measurement metric. For the social information, we extract the social features from two real datasets: MIT reality mining data and Infocom2006 trace, as shown in Table 4.2. Each node has a social feature vector, which indicates its characteristic in the social features. We use a 2-dimensional social feature vector as an example. Dimension 1 corresponds to *city* with three distinct values: New York (0), London (1), and Paris (2); and dimension 2 corresponds to *gender*, with two distinct values: male (0) and female (1). A user with social feature vector  $[0, 1]$  represents a female working in New York. If two nodes have exactly the same value in one dimension, we assume their distance in this dimension is 0. Otherwise, their distance is 1. The social feature distance is the summation of the distances in all dimensions. For example, a female working in New York with social feature vector  $[0, 1]$ , has social feature distance 1 to a female working in Paris with social feature vector  $[2, 1]$ , and social feature distance 2 to a male working in London with social feature vector  $[1, 0]$ . We use the social feature distance as another metric to measure the tie strength.

In this thesis, we model tie strength as a linear combination of the number of contacts and social feature distance:

$$w_{ij} = \alpha C_{ij} + \beta \frac{1}{1 + D_{ij}}, \quad (4.28)$$

where  $w_{ij}$  represents the tie strength of nodes  $N_i$  and  $N_j$ .  $C_{ij}$  ( $C_{ij} \in [0, 1]$ ) represents the normalized number of contacts between these two nodes, which includes the call logs and proximity data.  $D_{ij}$  ( $D_{ij} = 0, 1, 2, 3, \dots$ ) represents the social feature distance between these two nodes, which is used to measure the closeness between two nodes.  $\alpha$  and  $\beta$  represent the impact of contact information and social feature information, respectively. At the same time,  $\alpha + \beta = 1$ .

In the learning process, each node calculates the tie strength with the encountered nodes, and creates a weighted adjacency matrix. Then, the definition of strong and weak ties was established as follows: following [87, 88], from the weighted adjacency matrix, we used as a threshold the 59th percentile of the link weights cumulative distribution; then, links weighted higher than or equal to the threshold were considered as strong ties, while links with a weight less than the threshold were marked as weak ties.

For example, Fig. 4.22 illustrates a five-nodes social network,  $G(V, E)$ , where each link shows the condition of the linked pairwise nodes. As shown in Fig. 4.22,  $x/y$  labeled on each link represents *the number of contacts/the social feature distance*. In this example, each node has 5 social features. Here, we assume  $\alpha$  and  $\beta$  in Eq. 4.28 are the same, with value  $\frac{1}{2}$ . Therefore, we can create a  $5 \times 5$

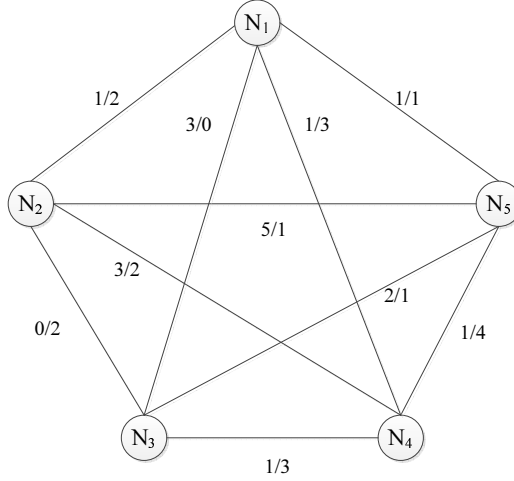


Figure 4.22: An illustration of a five-nodes social network. The value  $(x/y)$  on the links represent the number of contacts/the social feature distance of linked vertices.

weighted symmetric adjacency matrix according to Eq. 4.28:

$$\begin{bmatrix}
 0 & \frac{4}{15} & \frac{4}{5} & \frac{9}{40} & \frac{7}{20} \\
 \frac{4}{15} & 0 & \frac{1}{6} & \frac{7}{15} & \frac{3}{4} \\
 \frac{4}{5} & \frac{1}{6} & 0 & \frac{9}{40} & \frac{9}{20} \\
 \frac{9}{20} & \frac{7}{15} & \frac{9}{40} & 0 & \frac{1}{5} \\
 \frac{7}{20} & \frac{3}{4} & \frac{9}{20} & \frac{1}{5} & 0
 \end{bmatrix} \quad (4.29)$$

Based on this adjacency matrix, we can calculate the threshold  $((\frac{4}{5} - \frac{1}{6}) \times 59\% \approx 0.374)$ . Finally, we can distinguish the strong and weak ties. In this example, the links  $(N_1, N_3)$ ,  $(N_2, N_4)$ ,  $(N_2, N_5)$ , and  $(N_3, N_5)$  are strong ties, while the links  $(N_1, N_2)$ ,  $(N_1, N_4)$ ,  $(N_1, N_5)$ ,  $(N_2, N_3)$ ,  $(N_3, N_4)$ , and  $(N_4, N_5)$  are weak ties.

Then, each node can maintain its social tie table, recording the relationship to its encountered nodes. In each contact, the encountered nodes will exchange their social tie tables.

## Two-Phase Token-based Message Forwarding

### Influential and susceptible nodes

Local bridges linking multiple communities can disseminate the information among these communities. In a large social network, the local bridges are more likely to be the weak ties [84]. The *susceptible* nodes with many weak ties have high probability, located on the local bridges. A variety of literatures have claimed that susceptible nodes play a key role in novel information dissemination. In this section, we propose a token-based message forwarding with two phases: *weak tie-driven for-*

---

**Algorithm 9** Weak Tie-Driven Forwarding

---

```
/* When a message holder  $N_i$  with  $c$  tokens meets node  $N_j$  without the message. */  
/* Forward the message to  $N_j$ . */  
if  $W_j > W_i$  then  
    Give  $\left\lceil \frac{W_j}{W_i+W_j}c \right\rceil$  number of tokens to  $N_j$   
end if
```

---

warding and *strong tie-driven forwarding*. The mobile nodes with the message tokens can forward the message to the encountered nodes. In the early stage, susceptible nodes will receive more tokens, which we call 'weak tie-driven forwarding'. After a while, the message has been spread to multiple communities. It is more important that *influential* nodes, with more strong ties, deliver the message to the group members. Therefore, we change to a strong tie-driven forwarding scheme, which forwards more tokens to influential nodes.

**Weak tie-driven forwarding**

The individuals with many weak ties are considered to be the susceptible members in the MOSNs, who do not cluster in the network, while the influential individuals do [76]. A node with many weak ties has a relatively high probability of locating on a local bridge, which links different communities; therefore, susceptible members play a key role in novel information dissemination. Here, we propose a weak tie-driven forwarding algorithm in the early stage of information dissemination. When a message holder  $N_i$ , with  $c$  number of tokens, encounters a node  $N_j$  without that message, it forwards the message to  $N_j$ . If  $N_j$  has more weak ties than  $N_i$ ,  $N_i$  will give  $\left\lceil \frac{W_j}{W_i+W_j}c \right\rceil$  number of tokens to  $N_j$ , where  $W_i$  is the number of weak ties of  $N_i$ . Algorithm 9 shows the whole process of weak tie-driven forwarding.

**Strong tie-driven forwarding**

The influential individuals, with many strong ties, cluster in the network and can influence the other infected nodes with a high probability in the local communities. When the information has been spread to multiple communities, our forwarding strategy will turn to a strong tie-driven forwarding.

There are many factors that affect the two phases' switch time. However, considering the properties of MOSNs, every node does not have the global information, such as how many nodes in the network have received the new information. Therefore, the token holders can only use the number of tokens they have to estimate the information dissemination situation.

When the number of tokens held by the mobile nodes is below a predefined threshold, our information dissemination process switches to the second phase: strong tie-driven forwarding. The strong tie-driven forwarding phase is also a token-based broadcast process. When a message holder  $N_i$ , with  $c$  number of tokens, encounters a node  $N_j$  without a message, it forwards the message to

$N_j$ . The number of tokens assigned to  $N_j$  is  $\left\lceil \frac{S_j}{S_i+S_j}c \right\rceil$ , if  $N_j$  has more strong ties than  $N_i$ . Here,  $S_i$  is the number of strong ties of  $N_i$ .

### 4.3.3 Simulation and Evaluation

We evaluate the performance of the proposed two-phase token-based message forwarding scheme (**TTF**) through trace-driven simulations.

#### Simulation Setting and Comparison Scheme

In all experiments, the first half of the trace is used for the learning process, which is for the accumulation of the network information, the process of collecting the contact information, and calculating the tie strength combined with the social feature information. After the first half learning process, we can create an adjacency matrix, then distinguish the weak ties and strong ties following the method we discussed in Section 4.3.2. The new information generation and information dissemination happens during the second half of both traces.

In the simulation, we compare our proposed scheme with the following information dissemination schemes in MOSNs:

- **Flooding (F)**, in which the message holder will forward half of the token to the encountered node without the message.
- **Weak tie-driven forwarding (WF)**, which is the same as the first phase of our proposed two-phase token-based scheme.
- **Strong tie-driven forwarding (SF)**, which is the same as the second phase of our proposed two-phase token-based scheme.

#### Different Schemes for Tie Strength Calculation

First, we compare the performance of different tie strength calculation schemes. As we discussed previously, we will compare the performance in different values of  $\alpha$  and  $\beta$  in Eq. 4.28. We set the initial number of tokens ( $C$ ) created by the source to  $\frac{n}{2}$ , where  $n$  is the number of nodes in the whole network. When the number of tokens held by the message holder is below a threshold  $\lceil \log_2 C \rceil$ , the token forwarding strategy will switch from the weak tie-driven forwarding phase to a strong tie-driven forwarding phase.

As shown in Tables 4.3 and 4.4, we can see that when one set of the information is excluded from learning process ( $\alpha = 1, \beta = 0$  or  $\alpha = 0, \beta = 1$ ), the delivery ratio will reduce, and latency

Table 4.3: The performance of different tie strength calculation schemes in MIT reality mining trace

	$\alpha = 0.5$ $\beta = 0.5$	$\alpha = 0.75$ $\beta = 0.25$	$\alpha = 0.25$ $\beta = 0.75$	$\alpha = 1$ $\beta = 0$	$\alpha = 0$ $\beta = 1$
<b><i>Delivery ratio</i></b>	81.3%	83%	80.8%	79.3%	76.5%
<b><i>Latency (Sec.)</i></b>	328k	310k	323k	327k	343k

Table 4.4: The performance of different tie strength calculation schemes in Infocom2006 trace

	$\alpha = 0.5$ $\beta = 0.5$	$\alpha = 0.75$ $\beta = 0.25$	$\alpha = 0.25$ $\beta = 0.75$	$\alpha = 1$ $\beta = 0$	$\alpha = 0$ $\beta = 1$
<b><i>Delivery ratio</i></b>	89.5%	89.1%	91%	87.8%	88.3%
<b><i>Latency (Sec.)</i></b>	40.3k	41.2k	40k	44.3k	42.5k

will increase. In the MIT trace, contact information can predict the tie strength more accurately, especially when  $\alpha = 0.75$  and  $\beta = 0.25$ , in Table 4.3. Therefore, in the rest of the simulation, we set  $\alpha$  to 0.75 and  $\beta$  to 0.25 in the MIT trace. In the Infocom2006 trace, the social feature information is more important than the contact information. When  $\alpha = 0.25$ ,  $\beta = 0.75$ , the performance is best among all schemes, as shown in Table 4.4. Therefore, in the rest of the simulation, we set  $\alpha$  to 0.25 and  $\beta$  to 0.75 in the Infocom2006 trace.

### Performance in Limited Initial Tokens

Then, we evaluate different protocols in different initial numbers of tokens ( $C$ ). The initial number of tokens created by the source is  $\left\lceil \frac{\sqrt{n}}{2} \right\rceil$ ,  $\lceil \sqrt{n} \rceil$ ,  $\frac{n}{2}$ , and  $n$ . When the number of tokens held by the message holder is below a threshold  $\lceil \log_2 C \rceil$ , in **TTF**, the token forwarding strategy will switch from the weak tie-driven forwarding phase to the strong tie-driven forwarding phase. Here, we compare the delivery ratio in different settings, and latency when the delivery ratio reaches 50%.

From Figs. 4.23 and 4.24, we can see that our proposed two-phase token-based information dissemination scheme has a much higher delivery ratio compared with the other three forwarding schemes, especially when the initial number of tokens is smaller. The simulation results indicate the robustness of our proposed scheme, and it performs much better than other schemes in limited resource conditions (here, we consider the initial number of tokens as the resource).

Since some schemes can not achieve 50% delivery ratio, when the initial number of tokens is  $\left\lceil \frac{\sqrt{n}}{2} \right\rceil$  in both MIT reality mining and Infocom2006 traces, we consider that the latency of these schemes in this condition is infinite. In Figs. 4.23 and 4.24, we find that our scheme can dramatically reduce the latency in all conditions. Flooding-based scheme can also achieve smaller latency when the initial number of tokens are increasing.

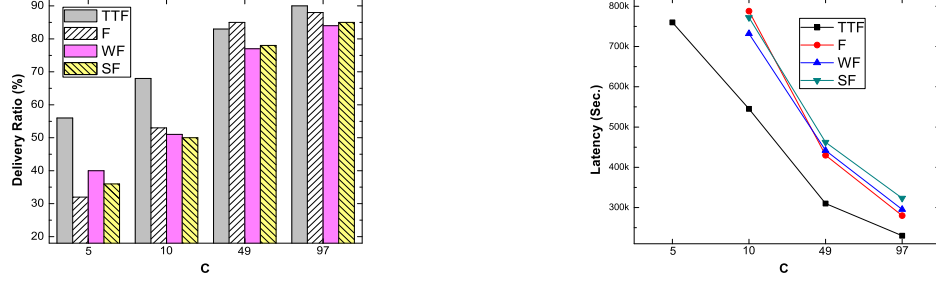


Figure 4.23: Comparing the performance of different schemes in different initial numbers of tokens in the MIT reality mining trace: ( $L$ ): delivery ratio; ( $R$ ): latency.

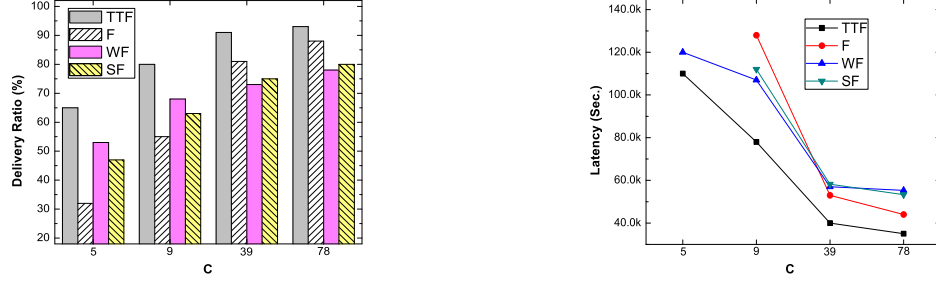


Figure 4.24: Comparing the performance of different schemes in different initial numbers of tokens in the Infocom2006 trace: ( $L$ ): delivery ratio; ( $R$ ): latency.

### Performance in Different Delivery Ratios

Here, we compare the latency of different protocols in different delivery ratios. We set the delivery ratio to 50%, 60%, 70%, and 80%. The initial number of tokens created by the source is  $\frac{n}{2}$  in this part. The probability vectors are the same as the previous part. The two phases' switch threshold is also  $\lceil \log_2 C \rceil = \lceil \log_2 n/2 \rceil$  in **TTF**.

Fig. 4.25 shows the latency comparison of different schemes in different delivery ratio constraints. Compared to weak tie-driven forwarding and strong tie-driven forwarding, our proposed two-phase token-based message forwarding scheme reduces the latency by about 15% in the MIT reality mining trace, and 21% in the Infocom2006 trace in Fig. 3.10. Our scheme performance is much better in the lower delivery ratio condition, compared to the flooding scheme. This means that our scheme can spread the new information quickly at the beginning of information dissemination, while flooding scheme may waste the contacts among the friend nodes, who contact each other in high frequency.

### Impact of The Two Phases' Switch Threshold

We also compare the performance of our proposed two-phase token-based message forwarding in different switch thresholds. The comparison switch thresholds are set as  $\lceil \frac{\log_2 C}{2} \rceil$ ,  $\lceil \log_2 C \rceil$ , and  $\lceil 2 \times \log_2 C \rceil$ , respectively. The initial number of tokens created by the source is set to  $\frac{n}{2}$ . The probability vectors are the same as the previous two parts. Here, we compare the delivery ratio in

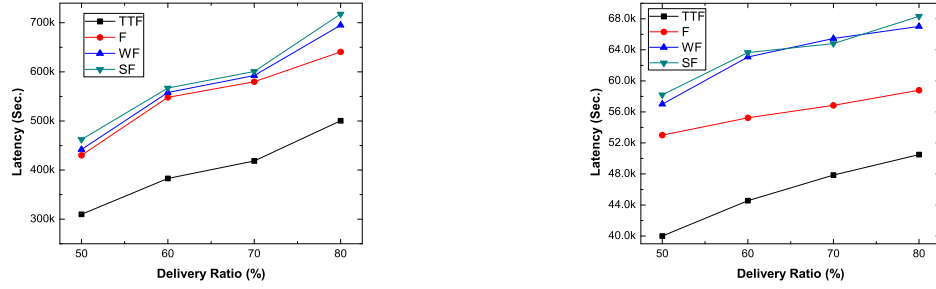


Figure 4.25: Comparing the performance of different schemes in different delivery ratios: (*L*): MIT; (*R*): Infocom.

different settings, and latency when the delivery ratio reaches 50%.

From Tables 4.5 and 4.6, we can see that in both MIT and Infocom2006 traces, the performance varies with the change of the two phases' switch threshold. When the threshold is too small, our scheme will work as weak tie-driven forwarding the majority of the time. When the threshold is too large, our scheme will switch to strong tie-driven forwarding quickly. In both situations, the delivery ratio will decrease, while the latency increases. Therefore, choosing an accurate switch threshold is very important for the performance of our proposed two-phase token-based message forwarding scheme.

### Summary of Simulation

In the simulation, we evaluate our proposed two-phase token-based information dissemination scheme in two real world mobile traces: the MIT reality mining campus trace, and the Infocom2006 conference trace. The simulation results show that our proposed scheme has better performance in both delivery ratio and latency in different network environments, compared to other schemes. When the resources are limited, which means the source node generates a small amount of tokens, our approach performs even better, increasing the delivery ratio and decreasing latency more dramatically than other approaches. The results in various delivery ratio scenarios indicate that our scheme can spread the novel information to certain fractions of users in the network quickly. By changing the value of the two phases' switch threshold, we find that our scheme performs best when the switch threshold is equal to  $\lceil \log_2 C \rceil$ , where  $C$  is the initial number of tokens generated by the source node. The fraction values in the two-phase token-based forwarding protocol guarantee the robustness of our scheme.

### 4.3.4 Conclusion

In this report, we present a social-tie-based information dissemination scheme in MOSNs. We leverage the strength of weak ties and susceptible nodes in novel information dissemination for token

Table 4.5: The performance of our scheme in different two phases' switch thresholds in MIT reality mining trace

Threshold	$\lceil \frac{\log_2 C}{2} \rceil$	$\lceil \log_2 C \rceil$	$\lceil 2 \times \log_2 C \rceil$
<i>Delivery ratio</i>	69%	83%	75%
<i>Latency (Sec.)</i>	363k	310k	345k

Table 4.6: The performance of our scheme in different two phases' switch thresholds in Infocom2006 trace

Threshold	$\lceil \frac{\log_2 C}{2} \rceil$	$\lceil \log_2 C \rceil$	$\lceil 2 \times \log_2 C \rceil$
<i>Delivery ratio</i>	85%	91%	82%
<i>Latency (Sec.)</i>	42.7k	40k	44.9k

split guidance. We design a tie strength calculation mechanism to distinguish the weak and strong ties, which considers both the contact and social feature information. The concept of social network – local bridge, is also used to enhance the information contagion. Then, a two-phase token-based message forwarding algorithm is introduced and evaluated in different network environments in real world mobile traces. The simulation results verify the effectiveness of our proposed approach. This is the first step of exploiting the information influence in MOSNs. Our future work will include more experiments on different social network traces to validate the effectiveness of our approach. We also plan to develop a mobile phone application to exploit the social network properties in large scale mobile social networks.

## 4.4 Conclusion of the Chapter

In this chapter, we study the broadcast problem in DTNs. First, we introduce a greedy solution for content distribution. Then, a ticket-based multiple packet broadcast scheme to disseminate the information efficiently. Finally, we study the broadcast problem from social network point of view by proposing a social-tie-based information dissemination approach.

In the next chapter, we will study the social-aware routing in DTNs. We will propose a hypercube-based social feature multi-path routing scheme.

## Chapter 5

# SOCIAL-AWARE ROUTING IN DTNS

Most routing protocols for delay tolerant networks resort to the sufficient state information, including trajectory and contact information, to ensure routing efficiency. However, state information tends to be dynamic and hard to obtain without a global and/or long-term collection process. In this chapter, we use the internal *social features* of each node in the network to perform the routing process. In Section 5.1, we present a hypercube-based social feature routing scheme. Section 5.2 will formally analyze this hypercube-based social feature multi-path routing approach.

### 5.1 Hypercube-based Social Feature Routing

In this section, we introduce a hypercube-based social feature routing in DTNs. This approach is motivated from several social contact networks, such as the Infocom 2006 trace, where people contact each other more frequently if they have more social features in common. Our approach includes two unique processes: social feature extraction and multi-path routing. In social feature extraction, we use entropy to extract the  $m$  most informative social features to create a feature space (F-space):  $(F_1, F_2, \dots, F_m)$ , where  $F_i$  corresponds to a feature. The routing method then becomes a hypercube-based feature matching process where the routing process is a step-by-step feature difference resolving process. We offer two special multi-path routing schemes: node-disjoint-based routing and delegation-based routing.

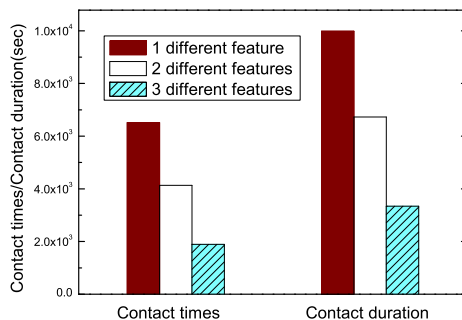


Figure 5.1: Comparison of the contacts in the Infocom 2006 trace.

### 5.1.1 Introduction

In social contact networks, where nodes (individuals) move around and interact at each contact based on their common interests, social features play an important role.

Several social-behavior-based DTN routing schemes have been proposed recently [53, 54, 64, 89, 90]. Most of these approaches consider the trajectory and/or the contact history of mobile nodes. However, most state information is dynamic and hard to obtain without a global and/or long-term collection process. In this report, we use the internal features of a node (an individual) for routing guidance. These features include nationality, affiliation, speaking language, and so on. This approach is motivated from several social contact networks where *people come in contact with each other more frequently if they have more social features in common*.

In Fig. 5.1, we show the difference in contacts when various features differ in the Infocom 2006 conference trace [61]-collected in a period of 337,417 seconds. We can see that the total contact times and contact duration reduce when the social feature difference between two individuals increases. The individuals with only one different feature have about 36.5% more contact times and 32.6% longer contact durations than the individuals with two different features. In [91], Mei et al. found that individuals with similar social features tend to contact more often in DTNs. Hence, we believe that designing a new routing protocol by considering the social features of individuals can improve the performance of DTN routing.

One of the main advantages of using features for routing guidance is its avoidance of state information collection. In addition, *feature-based routing converts a routing problem in a highly mobile and unstructured contact space ( $M$ -space) to a static and structured feature space ( $F$ -space)*. More specifically, each individual (a node in a DTN) is represented by a vector of  $(F_1, F_2, \dots, F_m)$ , where each feature  $F_i$  has  $n_i$  distinct values for  $i = 1, 2, \dots, m$ . In this way, the  $F$ -space contains  $\prod_{i=1}^m n_i$  nodes. Structurally, these nodes form an  *$m$ -dimensional hypercube*, in which two nodes are connected if and only if they differ in one feature. When  $n_i = 2$  for all  $i$ , it is called a binary

hypercube.

Although the initial idea of feature-based routing was proposed earlier in [91], our approach provides a systematic way of multi-path routing in the F-space by taking advantage of the structural property of hypercubes. We start by giving a model for representing the social features of each individual and introduce a method to measure the social similarities between individuals. Generally, each individual has many social features; however, some features are more important than others for routing purposes. Hence, in the *social feature extraction* process, we use Shannon entropy [92] to select  $m$  key social features. After that, individuals can be partitioned into different groups, each of which corresponds to a position in feature space (i.e., a hypercube node).

To perform efficient multi-path routing, node-disjoint routing is used to which a hypercube-based parallel feature matching process is applied. Feature differences are resolved step-by-step until the destination is reached. We also propose a feature matching shortcut algorithm for fast searching, which also ensures node-disjointness. Another way to achieve efficient multi-path routing is to extend delegation forwarding [49]. In delegation forwarding, a copy is made to a newly encountered node if this node is “closer” to the destination than the current node. Here, we use feature closeness as a forwarding metric and apply a feature-distance-based metric for copy redistribution.

In the simulation, we compare node-disjoint-based routing and delegation-based routing with spray-and-wait [53] and spray-and-focus [54], both in synthetic and real traces. To evaluate the impact of node density on the routing performance, we examine three cases in terms of the relative order between  $N$  (number of nodes in DTNs) and  $M = 2^m$  (number of nodes in the F-space): (1)  $M \ll N$  (i.e.,  $M = o(N)$ ), (2)  $M = N$  ( $M = \Theta(N)$ ), and (3)  $M \gg N$  ( $M = O(N)$ ).

### 5.1.2 Preliminaries

#### Social Features

Assume that there are  $N$  individuals in the system. Each individual can be represented by a social feature profile, a representation of her/his social features within a *feature space*, also called the F-space. The social features represent either physical features, such as gender, or logical ones, such as a membership in a social group.

In this report, we convert the mobile and unstructured contact space (M-space) with  $N$  individuals into a static and structured feature space (F-space) with  $M$  nodes. Fig. 5.2 represents a  $4 \times 3 \times 2$  F-space. It consists of 24 groups. In this example, there are three different social features in the F-space, represented by four, three, or two distinct values, respectively. In the F-space in Fig. 5.2, dimension 1 (the left most position) corresponds to *city* with four distinct values: New York (0),

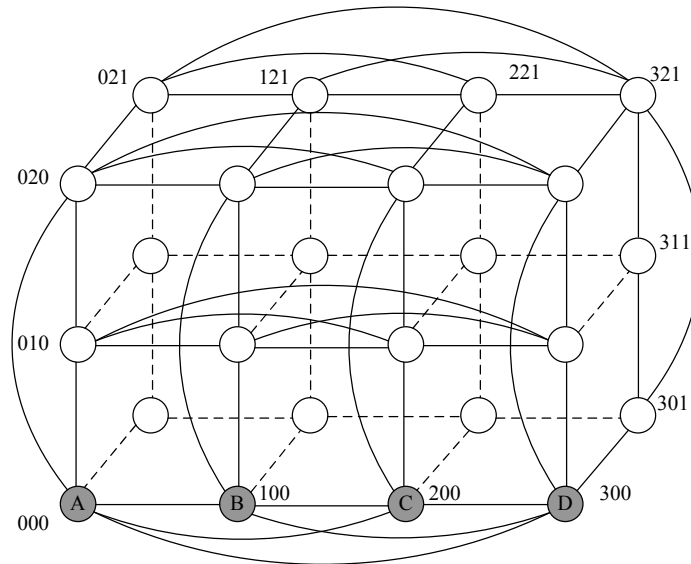


Figure 5.2: A 3-dimensional hypercube.

London (1), Paris (2), and Shanghai (3); dimension 2 (the second left most position) shows *position* with three distinct values: professor (0), researcher (1), and student (2); dimension 3 represents *gender* with two distinct values: male (0) and female (1). In Fig. 5.2, two groups have a connection if they differ in exactly one feature.

### Hypercubes and Hypercube Routing

Given the above definition of the feature space, we can represent the social feature profile for a group of users as a node in a hypercube. More specifically, the F-space  $(F_1, F_2, \dots, F_m)$  is mapped into an  $m$ -dimensional hypercube (or simply  $m$ -D cube), which consists of  $n_1 \times n_2 \times \dots \times n_m$  nodes. Two nodes,  $A = (a_1, a_2, \dots, a_m)$  and  $B = (b_1, b_2, \dots, b_m)$ , in an  $m$ -D cube are connected if and only if they differ in exactly one dimension (say  $i$ , such that  $a_i \neq b_i$ ). To express the virtual similarity between individuals in a cube, we use the feature distance to measure the closeness between two individuals.

The binary hypercube is a special cube in which each feature has a binary value: 0 and 1. In a binary cube, the feature distance between two individuals,  $A$  and  $B$ , is denoted as  $H_{AB}$ , which is the Hamming distance between  $A$  and  $B$ . We assume that source  $S$  has a packet for destination  $D$  with feature distance  $k$  in an  $m$ -D binary cube. There are exactly  $m$  node-disjoint paths from  $S$  to  $D$  based on the hypercube property [93, 94]. These paths are composed of  $k$  shortest paths of length  $k$  and  $m - k$  non-shortest paths of length  $k + 2$ .

In binary cube routing, the relative address of the current node and destination is calculated through XOR on two addresses and is sent, along with the packet, to the next node. The relative

distance is updated at each step until it becomes zero at the destination. We will extend this routing scheme by adding shortcuts for fast feature matching in multi-path routing.

### Delegation Forwarding

In delegation forwarding [49], each node has its estimated distance to the destination which is measured by quality ( $Q$ ). Initially, the quality level ( $L$ ) of each node is equal to its  $Q$ . A packet holder only forwards the packet to a node with a higher quality than its own level. In addition, the packet holder raises its own level to the quality of the higher quality node. This means a node will duplicate and forward a packet only if it encounters another node whose quality value is higher than any node met by the packet so far. It is shown that the expected cost of delegation forwarding in an  $N$ -node network is  $O(\sqrt{N})$ , compared to  $O(N)$  in the naïve scheme of forwarding to any higher quality node [49]. In this section, we use the feature distance as the quality value of the node to a given destination.

#### 5.1.3 Feature Extraction

The individuals are characterized by a high dimensional feature profile. However, usually only a small subset of features is important. We use the feature extraction method from data mining [95, 96] to obtain key features.

There are  $N$  individuals with  $m'$  features, which are denoted as  $F_1, F_2, \dots, F_{m'}$ . The goal of our social feature extraction is to extract the *most informative subset* (MIS) with  $m (< m')$  key features. We use Shannon entropy [92], which quantifies the expected value of the information contained in the feature, to select the key features:

$$E(F_j) = - \sum_{i=1}^{n_i} p(x_i) \log_2 p(x_i), \quad (j = 1, 2, \dots, m') \quad (5.1)$$

where  $E(F_j)$  denotes the entropy of the feature  $F_j$ , and  $p$  denotes the probability mass function of  $F_j$ .  $\{x_1, \dots, x_{n_i}\}$  are the possible values of feature  $F_j$ . The entropy of the feature considers not only the number of possible values, but also the distribution of their frequencies.

Table 5.1 shows the entropy of each social feature that we obtained from the Infocom 2006 trace [61]:  $m = 6$  most informative features out of  $m' = 10$  total features.

#### 5.1.4 Multi-Path Routing

We present a novel *social feature-based multi-path routing* scheme with the objective to reach the destination quickly while maximizing the delivery rate. The constraint is the number of copies of

Table 5.1: Entropy of the social features in the Infocom 2006 trace.

<b>Social Feature</b>	<b>Entropy</b>
<i>Affiliation</i>	4.64
<i>City</i>	4.45
<i>Nationality</i>	4.11
<i>Language</i>	4.11
<i>Country</i>	3.59
<i>Position</i>	1.37

the packet. The main objective is to distribute the copies of the packet in a cost-effective way.

We propose two special multi-path routing schemes: *node-disjoint-based*, where the copies are distributed to multiple node-disjoint paths to resolve the feature difference between the source and destination, and *delegation-based*, where the dissemination of copies is based on the feature distance to the destination.

We use the features of the destination to partition nodes into groups. This approach is called *destination-based partitioning*. At each dimension (i.e., feature), we separate nodes based on whether they have the same features as the one at the destination or not. In this way, a general cube is “compressed” into a binary cube even though each feature may have many different values. We will discuss another approach that uses general cubes directly in Section VI. Our routing scheme focuses on the group level, i.e., a node in a cube. Note that each group has many individuals who have the same partially matched features as the destination. The routing packet is forwarded from groups to groups until it reaches the destination group - the group where the destination is located. The packet can then be forwarded once more to the destination which is in the same group.

### Node-Disjoint-based Routing

Initially, the source has  $m$  copies of the packet to the destination in  $k$  feature distances. As we discussed in Section 5.1.2, there are  $k$  shortest paths of length  $k$  and  $m - k$  non-shortest paths of length  $k + 2$ , which are all node-disjoint.

Suppose that the source and destination differ in  $k$  dimensions  $\{1, 2, \dots, k\}$ , denoted as a set  $C$ .  $C^0 : \langle 1, 2, \dots, k \rangle$  is defined as the *coordinate sequence* (or *sequence*) from a given  $C$ .  $C^0$  determines how a path is constructed based on the resolution order of dimension differences given in  $C^0$ .  $C^i$  is defined as  $i$  circular left shifts of  $C^0$ . In fact,  $C^0$  can be any permutation of  $C$ . Then,  $k$  sequences,  $C^0, C^1, \dots, C^{k-1}$ , will create  $k$  node-disjoint shortest paths from  $C$ :

- *Path 1* generated by  $C^0$ :  $\langle 1, 2, 3, \dots, k \rangle$ ;
- *Path 2* generated by  $C^1$ :  $\langle 2, 3, 4, \dots, k, 1 \rangle$ ;

---

**Algorithm 10** Node-Disjoint-based Routing: source node contacts  $D$  or neighbor  $B$  in dimension  $i$

---

```

1: if  $B$  and  $D$  are the same group then
2:   Forward the packet to  $D$ .
3: else
4:   case  $i \in d$ :  $d = d - \{i\}$  and send  $(C^i, 0)$  to  $B$ .
5:   case  $i \in d'$ :  $d' = d' - \{i\}$  and send  $(C^i || i, 1)$  to  $B$ .
6:   case  $i \notin d \cup d'$ : do nothing.
7: end if

```

---

**Algorithm 11** Node-Disjoint-based Routing: non-source node contacts  $D$  or neighbor  $B$  in  $i$  with  $(seq : C', mode : m)$

---

```

1: if  $B$  and  $D$  are in the same group then
2:   Forward the packet to  $D$ .
3: else
4:   case  $m = 0 \wedge i = first(C')$ : send  $(C' - \{i\}, 0)$  to  $B$ .
5:   case  $m = 1 \wedge i \in C'$ : send  $(C' - \{i\}, 1)$  to  $B$ .
6: end if

```

---

- Path 3 generated by  $C^2$ :  $\langle 3, 4, 5, \dots, k, 1, 2 \rangle$ ;
- Path  $k$  generated by  $C^{k-1}$ :  $\langle k, 1, 2, \dots, k-2, k-1 \rangle$ .

Here, the path generated from source  $S$  by sequence  $C^0$  follows a matching process along dimension 1, dimension 2, and so on. In Fig. 5.3, from node  $G_0$  with sequence  $\langle 1, 2 \rangle$ , the path is  $(G_0, G_4, G_6)$ . In hypercube routing, the coordinate sequence of a path is sent along with the packet. After a successful forwarding along dimension  $i$ , dimension  $i$  will be deleted from the sequence. Clearly, the sequence becomes an empty sequence upon reaching the destination.

In Algorithm 10, for the source node, the source sends  $(seq, mode)$  to a matching neighbor, where  $mode$  is 0 for a shortest path or 1 for a non-shortest path.  $seq$  is the result of a circular left shift of  $C^0$  for  $mode=0$ .  $D$  is the destination. The routing packet is not included in the notation for simplicity. The source also maintains two vectors,  $d$  and  $d'$ .  $d$  is initialized as  $\{1, 2, \dots, k\}$ , which are different features between the source and destination.  $d'$  is  $\{k, k+1, \dots, m\}$ .

More specifically, when the source meets a neighbor with a feature difference in  $i \in d$ , which represents a dimension in a shortest path, the source sends  $C^i$  with  $mode = 0$  (which represents a strict coordinate sequence in  $C^i$ ) and removes  $i$  from  $d$ . If  $i \in d'$ , which represents a dimension in a non-shortest path, the source sends sequence  $C^0 || i$  with  $mode = 1$  (which represents any permutation of  $C$  followed by  $i$ ) and removes  $i$  from  $d'$ . If  $i \notin d \cup d'$ , no action is needed as shown in step 6, where the encountered node comes from a dimension to which a copy has been sent earlier.

In Algorithm 11, for a non-source node, source routing is used when the routing path is determined by the packet header  $seq$ . Step 4 represents short-path routing, where a strict coordinate sequence order is followed through extracting the first dimension in  $C'$ . Step 5 corresponds to non-shortest

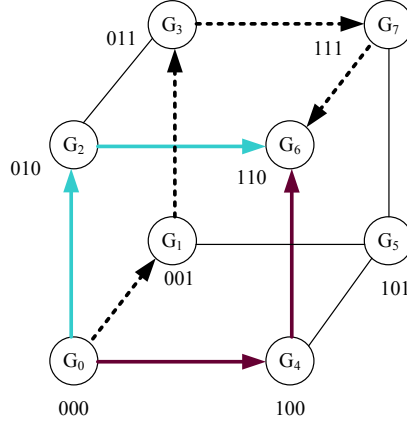


Figure 5.3: An example of node-disjoint-based routing with  $S = G_0$  and  $D = G_6$ . Solid directed paths are the shortest paths, and dashed directed paths represent the non-shortest paths.

path routing, where any permutation of dimension differences can be used. In Fig. 3, the non-shortest path can be either  $(G_0, G_1, G_3, G_7, G_6)$ , as shown in the figure, or  $(G_0, G_1, G_5, G_7, G_6)$ .

We also propose the *feature matching shortcut* for fast searching. In traditional hypercube routing, each forwarding can only correct one dimension at a time. When a packet holder meets another individual who is more than one feature distance away and is closer to the destination, the packet will not be forwarded to that individual. Here, we allow a controlled jump to a group that is more than one feature difference away while still ensuring node-disjointness. Such a controlled jump is called a *shortcut*, which is a *prefix*<sup>1</sup> of the coordination sequence. In Fig. 5.3,  $G_0$  can forward a copy of the packet directly to  $G_6$  as a shortcut for path  $(G_0, G_4, G_6)$ .

### Delegation-based Routing

Delegation-based routing forwards the copies of a packet only to the individual with a smaller feature distance to the destination. The number of copies to be forwarded is proportional to the feature distance to the destination.

In delegation-based routing, shown in Algorithm 12, there are two values to determine packet forwarding: *quality value* and *level value*. We use feature distance as the quality value. The quality value ( $Q_{AD}$ ) of individual  $A$  with destination  $D$  is inversely proportional to the feature distance between  $A$  and  $D$ ; that is,  $Q_{AD} = 1/H_{AD}$ . We simply use  $Q_A$  to represent  $Q_{AD}$ . When  $H_{AD}$  is 0, we set  $Q_A$  to  $+\infty$ . Initially, level value ( $L_A$ ), the highest level that  $A$  has met so far, is the same as  $Q_A$ .

In Algorithm 3, when  $A$ , with  $c$  copies of the packet, meets another individual  $B$  who has no copy but has a higher quality level  $L_B$  (note that  $L_B = Q_B$  in this case) than  $A$ 's level  $L_A$ ,  $A$  will

<sup>1</sup>Subsequence  $\langle 1, 2, \dots, k' \rangle$  is a *prefix* of  $\langle 1, 2, \dots, k \rangle$ , where  $k' \leq k$ .

---

**Algorithm 12** Delegation-Based Routing

---

```
1: /* Individual  $A$  meets  $B$ ,  $A$  has a packet with  $c$  copies and  $B$  has no copy for destination  $D$ . */
2: Initialize  $L_A \leftarrow Q_A$ .
3: if  $L_B > L_A$  then
4:   Forward  $\lceil (1 - L_A/L_B) \cdot c \rceil$  copies of the packet to  $B$ .
5:    $L_A \leftarrow L_B$ 
6: end if
```

---

forward  $\lceil (1 - L_A/L_B) \cdot c \rceil$  copies of the packet to  $B$  and update its level value to  $L_B$ .

### 5.1.5 Node-Disjointness

The multiple paths in hypercube routing are node-disjoint. The benefit of node-disjointness is that it guarantees that the multiple paths will not cross each other, except at destination  $D$ , to increase the efficiency of the routing. In this section, we prove that by including shortcuts, these paths still remain node-disjoint.

**Theorem 5** *In node-disjoint-based routing, the multiple paths with shortcuts are still node-disjoint paths.*

**Proof.** The non-shortcut paths are generated based on results in [93, 94], which are  $k$  node-disjoint paths of length  $k$  and  $m - k$  node-disjoint paths of length  $k + 2$ . All of these paths are generated through coordinate sequences starting from the source. Because each shortcut is a prefix of a coordinate sequence, all resultant paths still remain node-disjoint. ■

As shown in Fig. 5.3, one individual in  $G_0$  has a packet for another individual in  $G_6$ . The shortest paths are  $(G_0, G_2, G_6)$  and  $(G_0, G_4, G_6)$ , which follow the coordinate sequences we discussed in Section 5.1.4. The non-shortest path can be  $(G_0, G_1, G_3, G_7, G_6)$ . These three paths are node-disjoint. The shortcuts from the non-shortest path are  $(G_0, G_6)$ ,  $(G_0, G_7)$ ,  $(G_0, G_3)$ ,  $(G_1, G_6)$ ,  $(G_1, G_7)$ , and  $(G_3, G_6)$ .

### 5.1.6 Contact Frequency

We use the classic probability theory to draw some observations. We assume that the contact probability is time-independent [97]. We use contact numbers in the most recent time window to estimate contact probability (or precisely, frequency)<sup>2</sup>. More specifically, node  $S$  has  $p_1, p_2, \dots, p_m$  contact frequencies to its  $m$  neighbors along  $m$  dimensions that match the destination features in an  $m$ -D cube.  $p_{12\dots k}$  is denoted as the contact frequency between an individual from  $S$  and any individual in group  $D$  that matches destination features, where  $S$  and  $D$  differ in  $k$  features  $1, 2, \dots, k$ .

---

<sup>2</sup>Although contact duration is also important, results in [98] and Figure 1 show that there are high correlation coefficients of duration and frequency in many traces; we simply consider only frequency in this report.

Table 5.2: Comparison of contact frequency with different feature distance in the Infocom 2006 trace.

Path	Frequency	
(0000, 1000)	$p_1 = 0.196$	$P_{11}$
(1000, 1100)	$p_2 = 0.183$	$P_{22}$
(1100, 1110)	$p_3 = 0.192$	$P_{33}$
(1110, 1111)	$p_4 = 0.188$	$P_{44}$
(0000, 1100)	$p_{12} = 0.040$	$P_{12}$
(1000, 1110)	$p_{23} = 0.039$	$P_{23}$
(1100, 1111)	$p_{34} = 0.041$	$P_{34}$
(0000, 1110)	$p_{123} = 0.019$	$P_{13}$
(1000, 1111)	$p_{234} = 0.018$	$P_{24}$
(0000, 1111)	$p_{1234} = 0.01$	$P_{14}$
(0000, 1000, 1100)	$p_1 p_2 \approx 0.036$	$P_{1..2}$
(1000, 1100, 1110)	$p_2 p_3 \approx 0.035$	$P_{2..3}$
(1100, 1110, 1111)	$p_3 p_4 \approx 0.036$	$P_{3..4}$
(0000, 1000, 1100, 1110)	$p_1 p_2 p_3 \approx 0.007$	$P_{1..3}$
(1000, 1100, 1110, 1111)	$p_2 p_3 p_4 \approx 0.007$	$P_{2..4}$
(0000, 1000, 1100, 1110, 1111)	$p_1 p_2 p_3 p_4 \approx 0.0013$	$P_{1..4}$
(0000, 1000, 1100, 1110)	$p_1 p_2 p_3 \approx 0.007$	$P'_{1..3}$
(0000, 1100, 1110)	$p_{12} p_3 \approx 0.008$	
(0000, 1000, 1110)	$p_1 p_{23} \approx 0.008$	
(0000, 1110)	$p_{123} = 0.019$	
(1000, 1100, 1110, 1111)	$p_2 p_3 p_4 \approx 0.007$	$P'_{2..4}$
(1000, 1110, 1111)	$p_{23} p_4 \approx 0.007$	
(1000, 1100, 1111)	$p_2 p_{34} \approx 0.008$	
(1000, 1111)	$p_{234} = 0.018$	
(0000, 1000, 1100, 1110, 1111)	$p_1 p_2 p_3 p_4 \approx 0.0013$	$P'_{1..4}$
(0000, 1000, 1100, 1111)	$p_1 p_2 p_{34} \approx 0.0015$	
(0000, 1000, 1110, 1111)	$p_1 p_{23} p_4 \approx 0.0014$	
(0000, 1100, 1110, 1111)	$p_{12} p_3 p_4 \approx 0.0014$	
(0000, 1000, 1111)	$p_1 p_{234} \approx 0.0035$	
(0000, 1110, 1111)	$p_{123} p_4 \approx 0.0036$	
(0000, 1100, 1111)	$p_{12} p_{34} \approx 0.0016$	
(0000, 1111)	$p_{1234} = 0.01$	

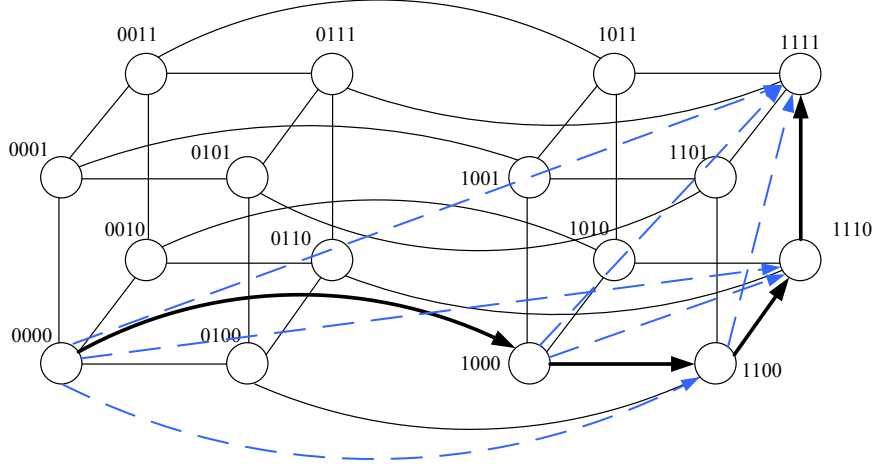


Figure 5.4: An example of the composite path from 0000 to 1111, where dashed directed lines are shortcuts.

Note that this frequency is not symmetric (i.e., the frequency from  $S$  to  $D$  is not the same as from  $D$  to  $S$ ). For simplicity, when we consider a path, its coordinate sequence is a consecutive ascending sequence, such as  $\langle 1, 2, \dots, k \rangle$ .

We conduct an experiment and obtain four social features with the highest entropy values (affiliation, city, nationality, language) from the Infocom 2006 trace to create a 4-D cube, as shown in Fig. 5.4. There are  $2^4 = 16$  groups in the cube. The source 0000 here represents a general source. If the destination has a different feature value than the source in a dimension, the corresponding bit is set to 1. In Fig. 4, we use destination 1111 to illustrate.

From Fig. 5.5, we can consider a *virtual directed triangle* with three nodes  $S$ ,  $B$ , and  $D$ .  $S$  to  $B$  includes dimensions  $i, i + 1, \dots, k$ .  $B$  to  $D$  has dimensions  $k + 1, k + 2, \dots, j$ . Hence,  $S$  to  $D$  spans dimensions  $i, i + 1, \dots, j$ . When  $j = i + 1$ , it corresponds to a *regular directed triangle* with  $A$  and  $B$  (and  $B$  and  $D$ ) differing in exactly one bit position.

We define  $P'_{i..j}$ , called *composite frequency*, as the frequency of a path from source  $S$  to destination  $D$  in the following dimension sequence  $\langle i, i + 1, \dots, j \rangle$ , including all possible shortcuts. The corresponding path is called a *composite path*, as shown in Fig. 4 from 0000 to 1111. This is the summation of the frequencies of all possible paths following the dimension sequence. In Fig. 5.5, we denote the composite frequency from  $S$  to  $D$  as  $P'_{S..D}$ .  $P_{ij}$  represents the frequency of a shortcut from dimension  $i$  to dimension  $j$ , which is equal to  $p_{i(i+1)..j}$ . We call  $P_{ij}$  *shortcut frequency*. The shortcut frequency from  $S$  to  $D$  is denoted as  $P_{SD}$ . The *direct frequency*,  $P_{i..j} = p_i p_{i+1} \dots p_j$ , corresponds to a direct path in our routing process from  $S$  to  $D$ , which is denoted as  $P_{S..D}$ .

**Theorem 6**  $P'_{i..j} = \sum_{k=i}^j P_{ik} P'_{k+1..j}$ , where  $i < j$  and  $i \leq k \leq j$ .  $P'_{i..i} = P_{ii} = p_i$ .

**Proof.** Without the loss of generality, we assume the source as  $S$  and the destination as  $D$ . The

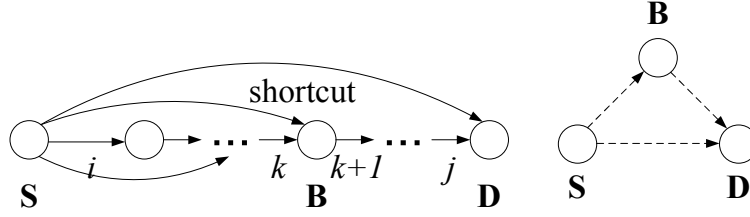


Figure 5.5: An illustration of contact frequency.

corresponding coordinate sequence is  $\langle i, i + 1, \dots, j \rangle$ , as shown in Fig. 5. From Fig. 5, each  $P_{ik}$  ( $i \leq k \leq j$ ) corresponds to a *prefix shortcut* from  $S$  to  $D$ .  $P'_{k+1..j}$  corresponds to the composite frequency of the remaining path to destination  $D$ . A simple summation of these paths enumerates each possible path from  $S$  to  $D$ . ■

Table 5.2 records shortcut, direct, and composite frequencies. Destination 1111 is generic and includes all nodes as possible destinations. The binary cube is constructed based on the destination-based partition that was discussed earlier. Here, we consider path (0000, 1000, 1100, 1110, 1111), which is one of the shortest paths from the source to the destination. From Table 5.2, we have the following two observations that relate to virtual and regular directed triangles:

**Observation 1:**  $P'_{S..D} < P'_{S..B}$ ,  $P'_{S..D} < P'_{B..D}$ , and  $P'_{S..D} > P'_{S..B}P'_{B..D}$ . This means that the composite frequency in the hypotenuse is smaller than each side of the triangle and is larger than the product of the composite frequencies of two sides.

**Observation 2:**  $P_{SD} < P_{SB}$ ,  $P_{SD} < P_{BD}$ ,  $P_{SD} > P_{SB}P_{BD}$ . This means that the shortcut frequency in the hypotenuse is smaller than each side and is larger than the product of the composite frequency of two sides.

Based on Observation 2, we can use induction to prove that  $P_{SD} > P_{S..D}$ , which means that the shortcut frequency is larger than the direct frequency for any path.

From the above observations, we can use shortcuts for fast delivery in terms of a smaller number of forwardings and a shorter delivery time. For given source  $S$  and destination  $D$ , we conjecture that direct frequency  $P_{S..D}$  is a lower bound of shortcut frequency  $P_{SD}$ , and composite frequency  $P'_{S..D}$  is an upper bound of  $P_{SD}$ . In our synthetic trace simulation, we will use these two bounds to generate shortcuts.

### 5.1.7 Extensions

#### General Hypercubes

In the previous sections, we discussed multi-path routing in a binary cube. We can extend this routing scheme to the general cube with multiple distinct values in each dimension without compression. We

can extend the basic scheme by treating all nodes that differ in a particular feature as a *clique*, i.e., a complete subgraph. Fig. 2 shows a clique \*00 in dark color, where \* is a wild card for 0, 1, 2, and 3. The corresponding nodes in the clique are  $A$ ,  $B$ ,  $C$ , and  $D$ , respectively.

Although each pair of nodes in the clique is directly connected, they may not be in contact in the near future (i.e., a low contact frequency). In Fig. 5.2, we assume that  $A$  holds a packet to  $D$  that has the same value as the destination address in that dimension ( $D$  is called a *destination at a dimension*). If node  $A$  meets another node that has a higher contact frequency to node  $D$ , forwarding is allowed. This is the same idea as delegation forwarding, but is used within one dimension. We call this approach *general forward*. The contact frequency is calculated locally based on 2-hop contact history at each node, without resorting to global contact information.

In order to control the hop-count, we can modify the general forward to allow the packet to be forwarded twice at most in each dimension. We call this approach *2-hop general forward*. In this way, we can control the total number of forwardings. In the above example, when  $A$  has a contact with  $B$  that has a higher contact frequency with  $D$ ,  $A$  will forward the packet to  $B$ . Then,  $B$  will hold the packet until it meets  $D$ .

In our simulation, we compare general forward and 2-hop general forward with *general wait* (i.e., routing schemes in binary hypercubes in Section 5.1.4), which will hold the packet until meeting with the destination at a particular dimension.

### Cube-Connected-Cubes (CCCs)

When the initial number of copies of the packet is less than the number of dimensions, we can use the cube-connected-cubes (CCCs) to enhance the performance.

We assume that there are only  $c$  copies of the packet in the source, and the destination is  $k$  feature distances away with  $k > c$ . Here, we assume that  $k$  is divisible by  $c$ . In CCCs,  $k$  dimensions are partitioned into  $c$  groups, each of which includes  $k/c$  dimensions. To offer a good partition for braiding relevant features into the same group, first we pick the highest entropy feature  $F_i$  and select  $k/c$  largest values of *mutual information*  $I(F_i; F_j)$  [92] (details in the next paragraph) as the most relevant features to be braided with  $F_i$ . Here,  $F_j$  is an unselected feature. Then, we repeat the same process with the remaining features to create  $c$  groups of braided features. In CCCs with  $k$  dimensions, an inner  $(k/c)$ -dimensional cube can be considered as a node of an outer  $c$ -dimensional cube. Inside the inner cube, there are  $2^{k/c}$  paths, and the outer cube has  $c$  node-disjoint paths for  $c$  copies. Therefore, CCCs explore more paths compared to the basic scheme.

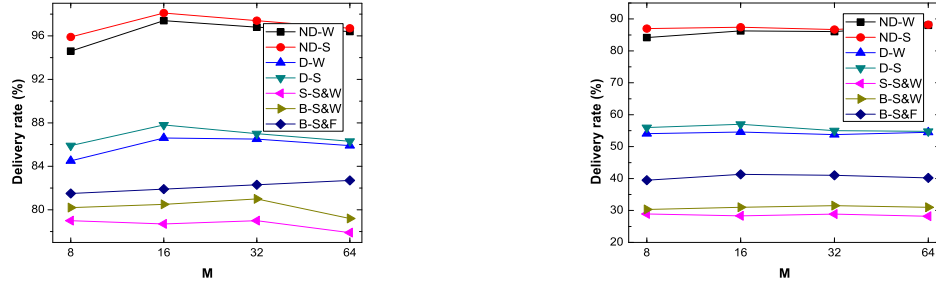


Figure 5.6: Comparing the delivery rate in the real trace: (left:  $L$ ): 20 packets and (right:  $R$ ): 100 packets.

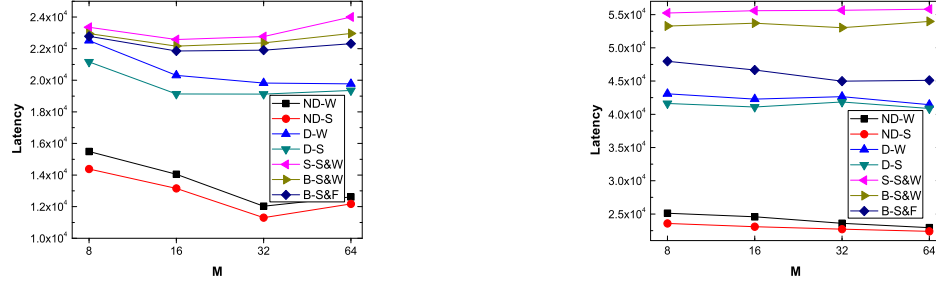


Figure 5.7: Comparing the latency in the real trace: ( $L$ ): 20 packets and ( $R$ ): 100 packets.

The mutual information of two feature variables,  $X$  and  $Y$ , can be defined as:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left( \frac{p(x,y)}{p(x)p(y)} \right) \quad (5.2)$$

where  $p(x,y)$  is the *joint probability distribution function* of  $X$  and  $Y$ , and  $p(x)$  and  $p(y)$  are the marginal probability distribution functions of  $X$  and  $Y$ , respectively.  $p(x,y)$  is equal to the product of  $p(x)/p(y)$  and conditional probability  $p(y|x)/p(x|y)$ :  $p(x,y) = p(y|x)p(x) = p(x|y)p(y)$  [92]. Mutual information quantifies the dependence between the joint distribution of  $X$  and  $Y$ . Hence,  $I(X;Y)$  is larger when features  $X$  and  $Y$  are more similar.

In the simulation, we compare our method (*entropy-based*) with random feature braiding (*random*) and parallel path routing (*parallel*).

### 5.1.8 Simulation

We compare the performance of the proposed multi-path routing scheme with several existing ones, including spray-and-wait and spray-and-focus, in Matlab, using both real and synthetic traces. The simulation is grouped into the following categories.

- Varying node density: we show that multi-path routing is robust under different node density conditions.

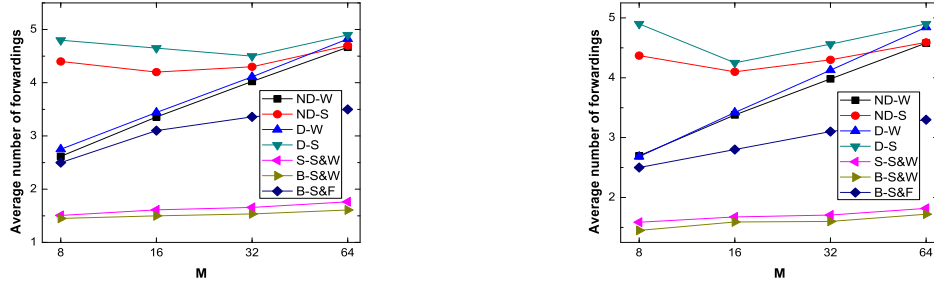


Figure 5.8: Comparing the number of forwardings in the real trace: (L): 20 packets; (R): 100 packets.

- The importance of the non-shortest path in node-disjoint-based routing: we illustrate the impact of the non-shortest path in node-disjoint multiple path routing.
- Comparing the extensions: we compare different methods in two extensions: general hypercubes and cube-connected-cubes (CCCs).

### Simulation Methods and Setting

We implement and compare seven routing schemes in the simulation. The first four are our proposed schemes. In all schedules, we consider  $m$  copies.

1. **Node-disjoint-based with wait-at-destination (ND-W)**: Waiting for the destination after the packet enters the destination group in node-disjoint-based routing.
2. **Node-disjoint-based with spray-at-destination (ND-S)**: Spraying  $N/(2M)$  copies into the destination group after the packet enters the group in node-disjoint-based routing.
3. **Delegation-based with wait (D-W)**: Same final step as ND-W in delegation-based routing.
4. **Delegation-based with spray (D-S)**: Same final step as ND-S in delegation-based routing.
5. **Source spray-and-wait (S-S&W)**: *Spray* phase: the source forwards copies to the first  $m$  distinct nodes it encounters. At the end of the spray, each packet holder has one copy; *Wait* phase: if the destination is not found in the spray phase, the copy carriers wait for the destination.
6. **Binary spray-and-wait (B-S&W)**: *Spray* phase: any node with copies will forward half of the copies to the encountered node with no copy; *Wait* phase: the same as S-S&W.
7. **Binary spray-and-focus (B-S&F)**: *Spray* phase: same as B-S&W; *Focus* phase: if the destination is not found in the spray phase, the copy carriers forward the copy to the encountered node with a smaller feature distance to the destination.

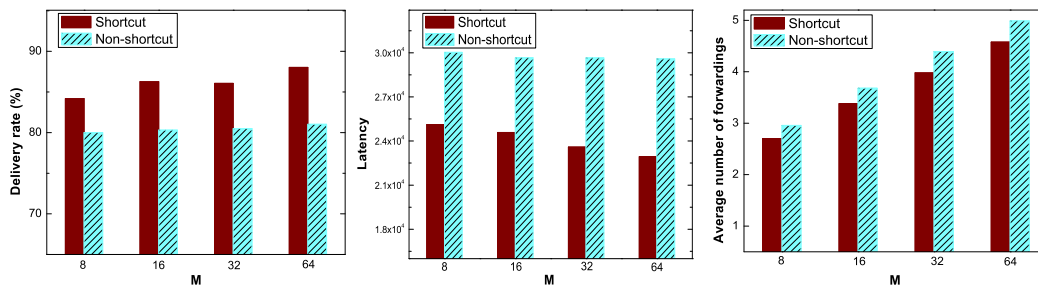


Figure 5.9: Comparing *shortcut* and *non-shortcut* with 100 packets in the real trace: (*L*): delivery rate, (*Center*): latency, and (*R*): number of forwardings.

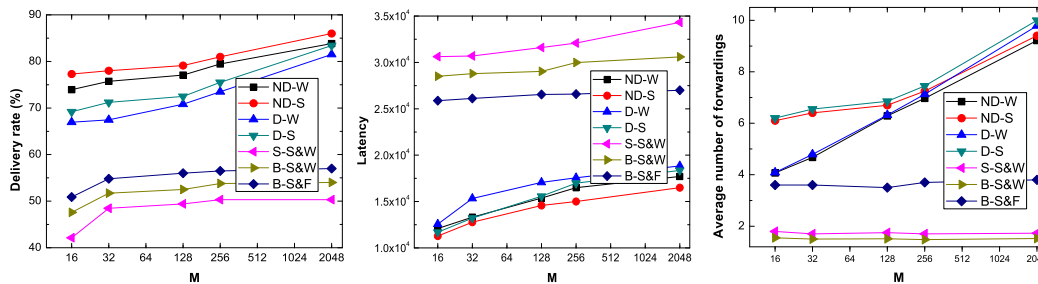


Figure 5.10: Comparing with 100 packets in the synthetic trace: (*L*): delivery rate, (*Center*): latency, and (*R*): number of forwardings.

1) *Real trace*: we use the Infocom 2006 trace [61, 64] in our simulation. This data set consists of two parts: *contacts* between the iMote devices that are carried by participants and *social features* of the participants, which are the statistics of participants' information from a questionnaire form. Firstly, we discard some participants that do not have social features in their profiles. In this way, we reduce the number of participants to 61. There are 74,981 contacts between these participants over a period of 337,418 time slots in seconds. We extract six social features from the original dataset: nationality, language, affiliation, position, city, and country.

2) *Synthetic trace*: we assume the contact frequency between pairwise individuals with only one different feature. That is, a node  $A$  has  $m$  contact frequencies,  $p_1, p_2, \dots, p_m$ , with its  $m$  neighbors in the  $m$ -D F-space. To estimate the contact frequency of node  $B$  that is more than one feature distance away from  $A$ , the shortcut frequency  $P_{AB}$  is randomly selected between its lower bound ( $P_{A..B}$ , direct frequency) and its upper bound ( $P'_{A..B}$ , composite frequency). In our synthetic trace, we create 128 individuals and 50,000 time slots in seconds. Contacts are randomly selected from these time slots based on selected frequencies.

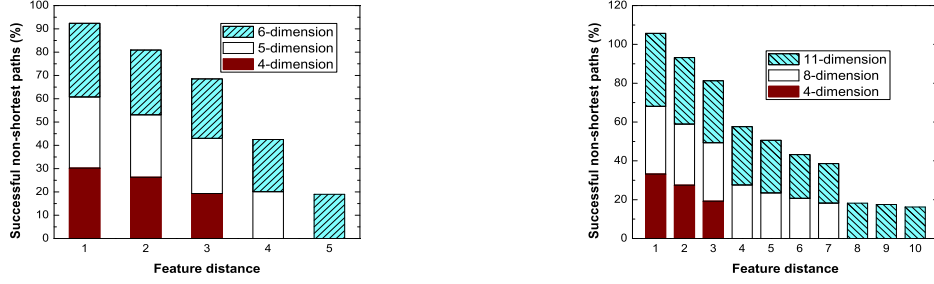


Figure 5.11: Comparing in the percentage of the successful non-shortest path: (*L*): real trace and (*R*): synthetic trace.

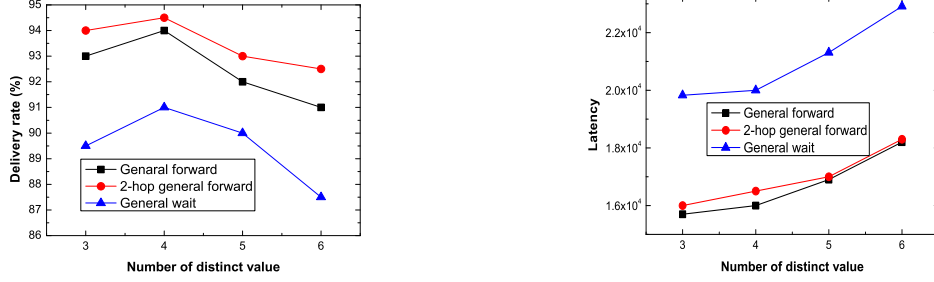


Figure 5.12: Comparing in general hypercubes in the real trace: (*L*): delivery rate and (*R*): latency.

### Varying node density

in this section, we compare the performance of multi-path routing with spray-and-wait and spray-and-focus with varying node densities. In the real trace, as we selected six social features, we set the number of nodes ( $M = 2^m$ ) in the F-space to 8, 16, 32, and 64. In the synthetic trace, we set 16, 32, 128, 256, and 2,048 nodes in the F-space to examine different schemes at a larger scale. We also compare these routing schemes with 20 and 100 packets, which are created at the rate of one packet per 5 time slots.

From Figs. 5.6, 5.7, and 5.8, we can see that node-disjoint-based routing has the highest delivery rate and the lowest latency among all in the real trace. Delegation-based routing performs better than spray-and-wait and spray-and-focus schemes in both delivery rate and latency. Binary spray-and-wait has the smallest number of forwardings before reaching the destination, as it does not forward once there is only one copy left. Node-disjoint-based routing can reduce forwardings by about 5.5% compared to delegation-based routing. out of all multi-path routing schemes, spray-at-destination increases the delivery rate by 2% and reduces the latency by 5% compared to wait-at-destination. Although, the former will increase the number of forwardings when the node density is high. We also find that using shortcuts can increase the delivery rate by about 5%, cut latency by 15%, and reduce the number of forwardings by 8%, as seen in Fig. 5.9.

As the results in the 20 and 100 packets conditions show the same trend, we only report results

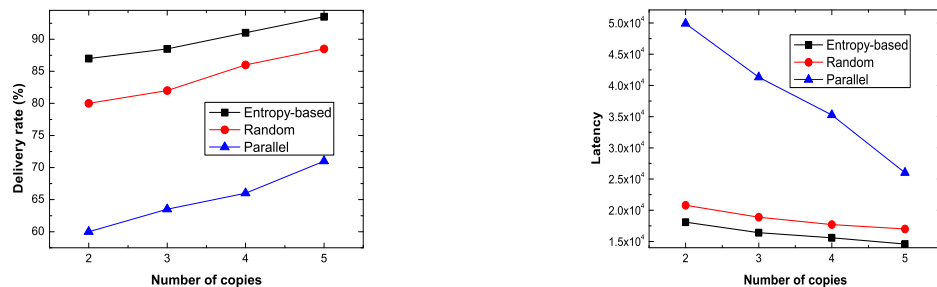


Figure 5.13: Comparing in CCCs in the real trace: (L): delivery rate and (R): latency.

for the 100 packets condition in the synthetic trace. Node-disjoint-based routing has the highest delivery rate and lowest latency in Fig. 5.10. Multi-path routing increases the average number of forwardings compared with spray-and-wait and spray-and-focus. Using shortcuts improves the overall performance in node-disjoint-based routing.

### Non-shortest path

in this section, our simulation demonstrates the importance of the non-shortest paths in node-disjoint-based routing. In the real trace, we set the dimension of the F-space ( $m$ ) to 4, 5, and 6. In the synthetic trace, it is 4, 8, and 11. We compare the performance under different feature distances ( $k$ ). Both the real and synthetic traces show that as  $m - k$  increases, the percentage of the non-shortest paths that reach the destination before any shortest path also increases in Fig. 5.11. This is expected as there are  $m - k$  non-shortest paths and  $k$  shortest paths in a given  $m$ -D cube.

### Extensions

in general hypercubes, we compare the three methods that were discussed in Section 5.1.7: *general forward*, *2-hop general forward*, and *general wait*. In Fig. 5.12, 2-hop general forward is the best as it increases the delivery rate, shortens the hop-count, and reduces the latency compared with general forward. Although general wait can further reduce the hop-count, it will, at the same time, reduce the delivery rate and increase the latency significantly.

In CCCs, we compare the three methods that were discussed in Section 5.1.7: *entropy-based braiding*, *random braiding*, and *parallel path routing* under a given number of copies. We set six social features under both traces. In Fig. 5.13, we can see that entropy-based braiding has the best performance in terms of the delivery rate and the latency. The performance decreases noticeably when using parallel path routing.

## Summary of Simulation

Our simulation concludes that although multi-path routing increases the number of forwardings compared with spray-and-wait and spray-and-focus, it has a significantly higher delivery rate and reduces the latency, especially under node-disjoint-based routing. Node-disjoint-based routing has multiple node-disjoint paths, which help to improve search efficiency. In node-disjoint-based routing, shortcuts also can increase the delivery rate, lower the latency, and reduce the number of forwardings at the same time. The non-shortest path also plays an important role, especially when the number of node-disjoint paths is limited. When the node density is relatively high, there are more individuals in each group. Therefore, using spray-at-destination seems to be a viable solution to reduce the latency compared with wait-at-destination.

Simulation results of two extensions show that the proposed multi-path routing scheme can also achieve a competitive performance in general hypercubes, which is more practical in reality. Multi-path routing can still be effective when the number of copies is limited. This can be done through dimension braiding in CCCs. The competitive performances in all of the extensions verify that multi-path routing can be effective under different conditions in DTN routing.

### 5.1.9 Conclusion

In this section, we proposed a social feature-based multi-path routing scheme in DTNs. Our scheme has two parts: social feature extraction and multi-path routing. We used entropy to extract the most informative social features to build an  $m$ -dimensional hypercube. In multi-path routing, we presented two schemes: node-disjoint-based and delegation-based. In node-disjoint-based routing, the feature difference between the source and the destination is resolved in a step-by-step fashion during the routing process. Shortcuts were used for fast matching. In delegation-based routing, we extended delegation forwarding into the multi-copy model, which uses a feature-distance-based metric as the copy forwarding decision metric. Trace-driven simulation results showed that our proposed multi-path routing scheme performs better than both spray-and-wait and spray-and-focus. We believe that the social features will play an important role in routing in social contact networks. Our future work will include more experiments on different social network traces to validate our observations. We also plan to study more sophisticated routing schemes in general hypercubes.

## 5.2 Analysis of a Hypercube-based Social Feature Multi-Path Routing

As we discussed the hypercube-based social feature multi-path routing in the previous section, in this section we will formally analyze the performance of our proposed routing scheme.

### 5.2.1 Introduction

In hypercube-based social feature matching process, feature differences are resolved step-by-step until the destination is reached, which is a node-disjoint multi-path routing scheme according to the property of hypercube. The shortcut is introduced for fast feature matching, where more than one feature difference can be resolved at one time. The shortcut still can ensure the node-disjointness of these multiple parallel paths.

In this report, we formally analyze the performance of the proposed hypercube-based social feature routing by looking at the delivery rate and latency. Firstly, we study the shortcut case. Recursive formulas are presented in both multi-path and single-path cases. Then, we show the benefit of using multi-path routing. If no shortcut is used, the exact solutions for the expected values of latency and delivery rate are given for different routing schemes, with both single- and multi-path routing schemes. The analysis results show that using the feature matching shortcut can increase the delivery rate and reduce the latency.

In the simulation, we firstly verify our analysis results in synthetic and real traces. Then, we compare hypercube-based social feature multi-path routing with state-of-the-art routing protocols: spray-and-wait [53] and spray-and-focus [54], both in synthetic and real traces.

### 5.2.2 Analysis

In this part, we formally analyze the delivery rates and latencies of the hypercube-based feature matching process. Our formulas of latency are valid for any contact time distributions. We note that the contact time is typically assumed to be exponentially distributed in many studies, but recent empirical results show that, in some cases, it follows a power-law distribution; see e.g., [99]. Since our formulas are valid for any contact distributions, our results may be useful for further studies.

We first obtain a recursive formula for the latency in routing with shortcuts of all distances included. The recursive formula then allows for an efficient way of calculating the delivery rates and the latencies in the whole graph, with shortcuts of all distances included. Then, we will compare the delivery rates and latencies of routing where *shortcuts of all distances*<sup>3</sup> are included, with the case of

---

<sup>3</sup>*Shortcuts of all distances* are the shortcuts in any hops.

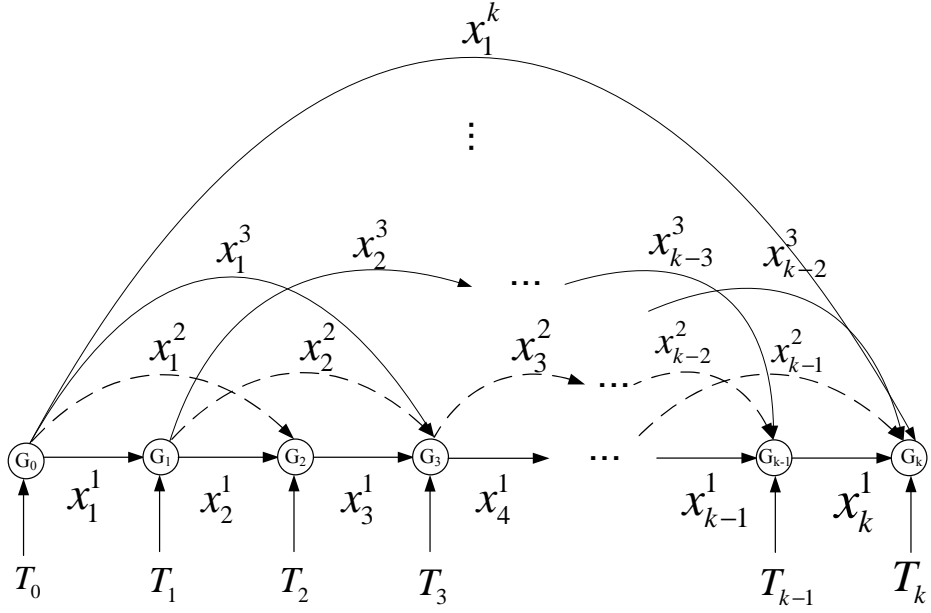


Figure 5.14: An illustration of the shortcut fast feature matching process.

routing with only neighboring groups<sup>4</sup> (routing without shortcuts) under the single path condition. Then, we extend our recursive formula to the multi-path case.

Finally, for the case of routing without any short cuts, we have an analytic expression for the delivery rate and the expected latency for single and multiple paths. These expressions are also valid for any type of contact time distribution. For the case of exponential contact distribution, we obtain an exact formula for the delivery rate and the expected latency. Our results also show that the multi-path routing scheme is more efficient than the single-path routing.

### Single-path Routing with Shortcuts

In this subsection, we first obtain a recursive formula for the latency in routing with shortcuts of all distances included. The recursive formula then allows for an efficient way of calculating the delivery rates and the latencies in the whole graph, with short-cuts of all distances included. Then, we will compare the performances of the shortcut and non-shortcut in the single-path scenario.

From Fig. 5.14, we assume that the source group and destination group is in  $k$  feature distances. Let  $G_0, G_1, G_2, \dots, G_k$  be the nodes of a single path of total feature distance  $k$ , where the feature distance between two adjacent nodes  $G_{j-1}$  and  $G_j$  is equal to 1, for all  $j = 1, \dots, k$ . For  $0 \leq i \leq k$ , let  $x_j^i$  be the minimum arrival time of the packet sent directly from  $G_{j-1}$  to  $G_{j-1+i}$ ,  $j = 1, 2, \dots, k+1-i$ . For the following recursive formula of latency,  $x_j^i$  can be any distribution.

**Theorem 7** *In a single-path routing with shortcuts of all distances included, let  $G_0, G_1, G_2, \dots, G_k$  be*

<sup>4</sup>The individuals that are in the same node within the hypercube are considered as a group. The neighboring groups are groups with exactly 1 feature distance.

the nodes of a single path of total feature distance  $k$ , where the feature distance between two adjacent nodes  $G_{j-1}$  and  $G_j$  is equal to 1, for all  $j = 1, \dots, k$ . For  $0 \leq i \leq k$ , let  $x_j^i$  be the minimum arrival time of a packet sent directly from  $G_{j-1}$  to  $G_{j-1+i}$ ,  $j = 1, 2, \dots, k+1-i$ . Let  $T_i$  be the latency of the packet arriving at group  $G_i$ , whose feature distance to the source group  $G_0$  is  $i$ . Then, the following recursive relations hold:

$$\begin{aligned}
T_0 &= 0 \\
T_1 &= x_1^1 \\
T_2 &= \min((T_1 + x_2^1), (T_0 + x_1^2)) \\
T_3 &= \min((T_2 + x_3^1), (T_1 + x_2^2), (T_0 + x_1^3)) \\
&\dots \\
T_k &= \min((T_{k-1} + x_k^1), (T_{k-2} + x_{k-1}^2), \dots, (T_0 + x_1^k)). \tag{5.3}
\end{aligned}$$

**Proof.** We will prove by induction. From Fig. 5.14, we have:

$$T_0 = 0, T_1 = x_1^1.$$

Assuming that Eq. 5.3 holds for all  $i \leq k$ , we will prove that it holds for  $k+1$ . We note that, when routing for a packet to arrive in  $G_{k+1}$ , it can first arrive in node  $G_i$ , then take a shortcut of distance  $k+1-i$  whose time is  $x_{i+1}^{k+1-i}$ . If  $G_i$  is the last node that the packet visited before arriving at  $G_{k+1}$ , then the minimum delivery time for taking this route from  $G_0$  to  $G_{k+1}$  is  $T_i + x_{i+1}^{k+1-i}$ . Therefore, the overall minimum delivery time from  $G_0$  to  $G_{k+1}$  is:

$$T_{k+1} = \min((T_k + x_{k+1}^1), (T_{k-1} + x_k^2), \dots, (T_0 + x_1^{k+1})).$$

Therefore, Eq. 5.3 holds for  $k+1$ . ■

The above recursive relations allow for an efficient way of calculating the expected latency of all shortcuts included. In what follows, we will give examples where  $x_j^i$ 's are independent and exponentially distributed. We will compare the results between the cases where all shortcuts are included, and the case where only 2-hop shortcuts are included.

First, if only 2-hop shortcuts exist, then  $x_j^i = \infty$  for all  $i \geq 3$  and all  $j$ . By the recursive relations,

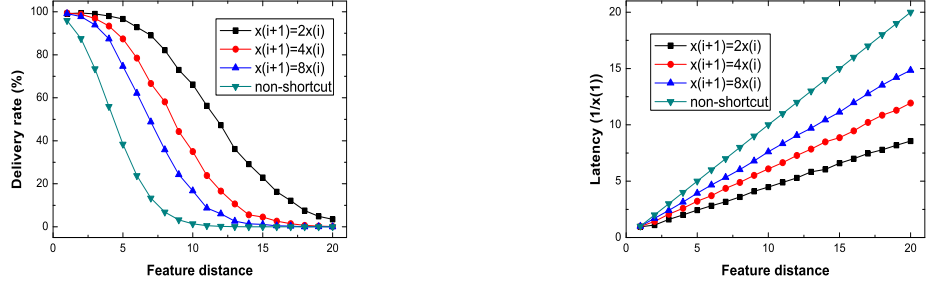


Figure 5.15: Comparison between non-shortcut and 2-hop shortcuts including in the single-path case. ( $x(i)$  is  $x_j^i$  for  $\forall j$ .)

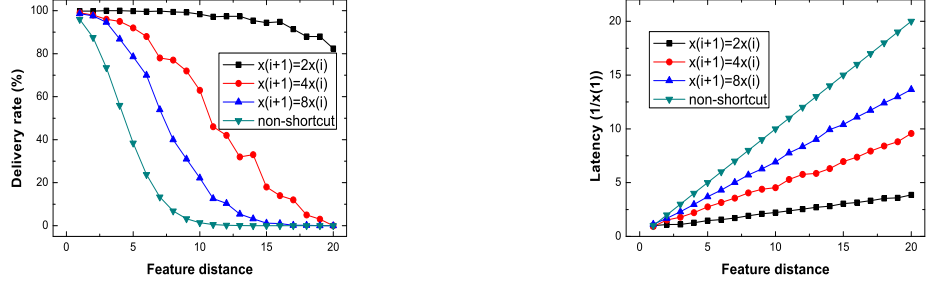


Figure 5.16: Comparison between non-shortcut and all shortcuts including in the single-path case.

we have:

$$\begin{aligned}
 T_2 &= \min((T_1 + x_2^1), (T_0 + x_1^2)) \\
 T_3 &= \min((T_2 + x_3^1), (T_0 + x_2^2)) \\
 &\dots \\
 T_k &= \min((T_{k-1} + x_k^1), (T_{k-2} + x_{k-1}^2)). \tag{5.4}
 \end{aligned}$$

Since the individuals with more features in common will contact each other more, we assume that  $x_j^{i+1} = a * x_j^i$  and that  $a$  is equal to 2, 4, and 8. Based on the above recursive relations, we can calculate the delivery rate and latency, as shown in Fig. 5.15. If all shortcuts are included,  $T_k$  can also be recursively calculated. Fig. 5.16 shows the comparison results. From both Figs. 5.15 and 5.16, we find that using shortcuts can significantly improve the performance. We also compare 2-hop shortcuts including, 2 and 3 hop shortcuts including, and all shortcuts including, which is shown in Fig. 5.17.

### Multi-path Routing with Shortcuts

In what follows, we give a recursive formula for multi-path routing with all shortcuts of all distances included. In this case, we consider the case where the source group  $G_0$  and destination group  $G_k$  are in  $k$  feature distance. Since there are exactly  $k$  node-disjoint shortest paths, for  $1 \leq l \leq k$ , we

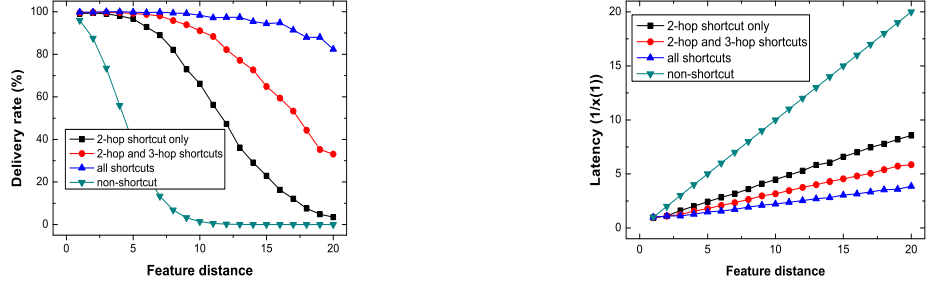


Figure 5.17: Comparison of the performance after including different numbers of hop shortcuts.

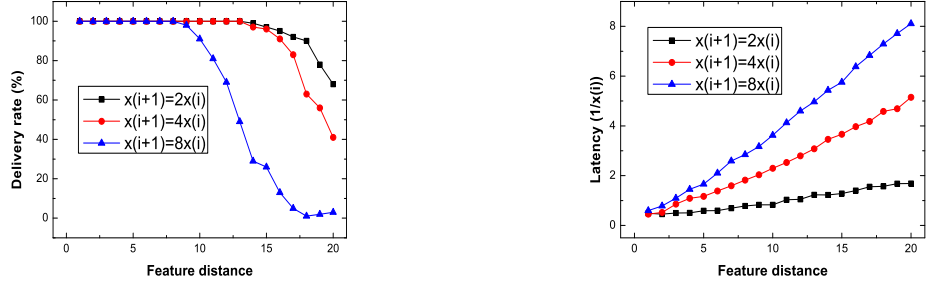


Figure 5.18: Comparison between non-shortcut and all shortcuts including in the multi-path case.

let  $G_{l0}, G_{l1}, G_{l2}, \dots, G_{lk}$  be the nodes of the  $l$ -th path of total feature distance  $k$  that connects  $G_0$  and  $G_k$ , where the feature distance between two adjacent nodes  $G_{l(j-1)}$  and  $G_{lj}$  is equal to 1, for all  $j = 1, \dots, k$ . For  $0 \leq i \leq k$ , let  $x_{lj}^i$  be the minimum arrival time of the packet sent directly from  $G_{l(j-1)}$  to  $G_{l(j-1+i)}$ ,  $j = 1, 2, \dots, k+1-i$ . For our recursive formula of latency,  $x_{lj}^i$  can be any distribution. Using a similar argument as the one in the proof of Theorem 7, we have the following:

**Theorem 8** *In a multi-path routing with shortcuts of all distances included, let  $T_{li}$  be the latency of the packet arriving at group  $G_{li}$  whose feature distance to the source group  $G_{l0}$  is  $i$  in the path  $l$ , and let  $T'_{li}$  be the latency of the packet arriving at group  $G_{li}$  in the path  $l$  without using the longest shortcut  $x_{l1}^k$ . Let  $S_{r,k}$  be the latency of the packet sent from the group  $G_0$  to arrive at group  $G_k$ , using  $r$  shortest paths  $l = 1, 2, \dots, r$ . Then the following recursive relations hold:*

$$\begin{aligned}
 T'_{lk} &= \min \left( (T_{l(k-1)} + x_{l1}^k), (T_{l(k-2)} + x_{l1}^{2(k-1)}), \right. \\
 &\quad \left. \dots, (T_{l1} + x_{l1}^{k-1}) \right), \\
 S_{r,k} &= \min (T'_{1k}, \dots, T'_{rk}, x_{11}^k).
 \end{aligned} \tag{5.5}$$

where the longest shortcuts in all paths are the same; i.e.,  $x_{l1}^k = x_{11}^k$ .

A numerical study for  $S_{r,k}$  is shown in Fig. 5.18. As shown in Table 5.3, using shortcut fast feature matching can improve the performance.

### Routing without Shortcuts: Multi-path vs. Single-path

In this subsection, we consider the case of routing without using shortcuts, where each pair of neighboring groups meet according to an independently identical distribution  $f$ . We give a general formula for the delivery rate and latency of the multiple paths routing scheme. In case  $f$  is an exponential density function with mean  $\frac{1}{\lambda}$ , we have an analytical expression for the delivery rate and the latency. We note that if the feature distance between the source and the destination is  $k$ , then there are exactly  $k$  shortest paths between the source and the destination of length  $k$ . In the following theorem, we find the delivery rate if  $r$  shortest paths are used in the routing scheme.

**Theorem 9** *Suppose the contact times for all pairs with feature distance 1 are independent with the same probability density function  $f$ . For the multi-path routing scheme, if  $r$  shortest paths are used, then the following hold:*

(a) *the delivery rate from the source group to the destination, with  $k$  feature distance in time  $t$ , is given by*

$$1 - \left[ \int_t^\infty \int_0^{x_k} \int_0^{x_{k-1}} \dots \int_0^{x_1} f(x_k - x_{k-1}) f(x_{k-1} - x_{k-2}) \dots f(x_2 - x_1) f(x_1) dx_1 \dots dx_k \right]^r; \quad (5.6)$$

(b) *the expected latency from the source group to the destination, with  $k$  feature distance, is given by*

$$\int_0^\infty \left[ \int_t^\infty \int_0^{x_k} \int_0^{x_{k-1}} \dots \int_0^{x_1} f(x_k - x_{k-1}) f(x_{k-1} - x_{k-2}) \dots f(x_2 - x_1) f(x_1) dx_1 \dots dx_k \right]^r dt. \quad (5.7)$$

**Proof.** (a) Since the multiple paths from the source to the destination are node-disjoint, the multiple paths are independent. By assumption, they also have the same distribution. Therefore, if  $r$  shortest paths are used, then the probability that the destination group has not received the packet by time  $t$  is

$$P(\tau_r > t) = \prod_{j=1}^r P(\tau^j > t), \quad (5.8)$$

where  $\tau_r$  is the delivery time when  $r$  paths are used and  $\tau^j$  is the delivery time when only  $j$ -th path is used.

Since the length of the shortest paths in  $k$  feature distance is  $k$ , and all neighboring groups meet according to the same distribution with density function  $f$ , the delivery time in  $k$  feature distance is a sum of  $k$  independent random variables with common density function  $f$ , and its probability

Table 5.3: Comparison of shortcut and non-shortcut in the multi-path case.

	<b>Feature distance</b>	2	3	4	5
<b>Delivery rate</b>	<i>Non-shortcut</i>	96%	93%	83%	64%
	<i>Shortcut</i>	100%	100%	100%	100%
<b>Latency</b> ( $\frac{1}{\lambda_1}$ )	<i>Non-shortcut</i>	1.25	1.68	2.18	2.72
	<i>Shortcut</i>	0.46	0.50	0.51	0.59

density function is given by the convolution of  $k$  copies of  $f$ . Therefore, the probability that for each one of the multiple paths fail in time  $t$  is

$$P(\tau^j > t) = \int_t^\infty h_k(x) dx, \quad (5.9)$$

where

$$h_k(x) = \int_0^x \int_0^{x_{k-1}} \dots \int_0^{x_2} f(x - x_{k-1}) \dots f(x_2 - x_1) f(x_1) dx_1 \dots dx_{k-1}. \quad (5.10)$$

Therefore, Eq. 5.8 can be represented as follows:

$$P(\tau_r > t) = \prod_{j=1}^r P(\tau^j > t) = \left[ \int_t^\infty h_k(x) dx \right]^r \quad (5.11)$$

Therefore, the delivery rate in time  $t$  is:

$$\begin{aligned} P(\tau_r \leq t) &= 1 - P(\tau_r > t) \\ &= 1 - \left[ \int_t^\infty \int_0^{x_k} \dots \int_0^{x_2} f(x_k - x_{k-1}) \dots f(x_2 - x_1) f(x_1) dx_1 \dots dx_k \right]^r. \end{aligned} \quad (5.12)$$

(b) Since the expected value of a non-negative random variable is equal to the integral of its failing rate at time  $t$  with respect to  $t$ , we have (b). ■

In the case where  $f$  is exponentially distributed with mean  $\frac{1}{\lambda}$ , we have the following analytical expressions for the delivery rate and latency.

**Theorem 10** *For the multi-path routing scheme, if  $r$  shortest paths are used, then the delivery rate from the source group to the destination, with  $k$  feature distance in time  $t$ , is:*

$$1 - \left[ \int_t^\infty \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!} dx \right]^r. \quad (5.13)$$

Table 5.4: The value of  $c_k$ .

$k$	1	2	3	4	5
$c_k$	1	1.25	1.68313	2.18264	2.72188

**Proof.** If  $f$  is an exponential density function with mean  $\frac{1}{\lambda}$ , then  $h_k(x)$  is a gamma distribution

$$g_{k,\lambda}(x) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!}.$$

Therefore, the theorem follows from Theorem 9 (a). ■

In the proposed hypercube-based routing scheme, there are  $k$  multiple node-disjoint shortest paths from the source to the destination, which differ in  $k$  feature dimensions. Hence,  $r$  is equal to  $k$  in the hypercube-based routing. It follows from Theorem 1 that for the case of exponential distribution, the delivery rate is  $1 - \left[ \int_t^\infty \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!} dx \right]^k$ .

**Theorem 11** *For the multi-path routing scheme, if  $r$  shortest paths are used, then the expected latency from the source group to the destination, with  $k$  feature distance, is  $\frac{c_{r,k}}{\lambda}$ , where*

$$c_{r,k} = \frac{(rk)!}{[(k-1)!]^r} \int_1^\infty \cdots \int_1^\infty \frac{u_1^{k-1} \cdots u_r^{k-1}}{(u_1 + \cdots + u_r)^{rk+1}} du_1 \cdots du_r.$$

**Proof.** By change of variables  $w = \frac{\lambda}{t}x$ , we have:

$$\int_t^\infty \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!} dx = \int_\lambda^\infty \frac{t^k w^{k-1} e^{-tw}}{(k-1)!} dw.$$

Then Eq. 5.11 can be represented as:

$$P(\tau_k > t) = \frac{1}{[(k-1)!]^r} \int_\lambda^\infty \cdots \int_\lambda^\infty \frac{t^{rk} w_1^{k-1} \cdots w_r^{k-1}}{e^{-(w_1 + \cdots + w_r)t} dw_1 \cdots dw_r}. \quad (5.14)$$

Since

$$\begin{aligned} & \int_0^\infty t^{rk} e^{-yt} dt \quad (y = w_1 + \cdots + w_r) \\ &= \int_0^\infty \frac{u^{rk}}{y^{rk}} e^{-u} \frac{du}{y} \quad (u = yt, \frac{du}{y} = dt) \\ &= \frac{1}{y^{rk+1}} \int_0^\infty u^{rk} e^{-u} du \\ &= \frac{(rk)!}{(w_1 + \cdots + w_r)^{rk+1}}, \end{aligned}$$

the expected latency can be represented as follows:

$$\begin{aligned}
E(\tau_k) &= \int_0^\infty P(\tau_k > t) dt \\
&= \frac{(rk)!}{[(k-1)!]^r} \int_\lambda^\infty \cdots \int_\lambda^\infty \frac{w_1^{k-1} \cdots w_r^{k-1}}{(w_1 + \cdots + w_r)^{rk+1}} dw_1 \cdots dw_r \\
&= \frac{(rk)!}{[(k-1)!]^r} \int_1^\infty \cdots \int_1^\infty \frac{\lambda^{(k-1)r} u_1^{k-1} \cdots u_r^{k-1}}{\lambda^{k^2+1} (u_1 + \cdots + u_r)^{rk+1}} \lambda^k du_1 \cdots du_r \\
&= \frac{1}{\lambda} \left( \frac{(rk)!}{[(k-1)!]^r} \int_1^\infty \cdots \int_1^\infty \frac{u_1^{k-1} \cdots u_r^{k-1}}{(u_1 + \cdots + u_r)^{rk+1}} du_1 \cdots du_r \right) \\
&= \frac{c_{r,k}}{\lambda}, \tag{5.15}
\end{aligned}$$

where  $c_{r,k} = \frac{(rk)!}{[(k-1)!]^r} \int_1^\infty \cdots \int_1^\infty \frac{u_1^{k-1} \cdots u_r^{k-1}}{(u_1 + \cdots + u_r)^{rk+1}} du_1 \cdots du_r$ . ■

In the proposed hypercube-based routing scheme, there are  $k$  multiple node-disjoint shortest paths from the source to the destination, which differ in  $k$  feature dimensions. Hence,  $r$  is equal to  $k$  in the hypercube-based routing. Let  $c_k = c_{k,k}$ . In the case of exponential contact time, our formulas yield the numerical values of  $c_k$ , listed in Table 5.4.

It follows from Theorem 9 that the delivery rate increases as  $r$  increases and the expected latency decreases as  $r$  increases. In the proposed hypercube-based social feature routing scheme,  $r$  is equal to  $k$ , which results in a higher delivery rate and smaller expected latency compared to the single-path scheme, in which  $r$  is equal to 1. Thus we have

*Remark 1: The proposed hypercube-based social feature routing scheme increases the delivery rate and decreases the expected latency compared to the single-path scheme.*

### 5.2.3 Discussion

In the previous subsection, we obtain analytical results for latencies in various routing schemes. Recursive formulas for latency are given for routing instances with all shortcuts included, in both single-path and multi-path schemes. These formulas are efficient for calculating the probability and expected values of latencies. However, it would be interesting for future studies to obtain exact solutions for the latencies with all shortcuts included. Moreover, Eq. 5.4 shows that  $T_k, k = 0, 1, 2, \dots$  is a Markov chain with 2-step memory. This may be useful in determining the asymptotic behavior of the expected values of  $T_k$ . Our future work will also include impact analysis of using the shortcut along a non-shortest path.

### 5.2.4 Simulation

First, we verify the analysis results in Matlab using both real and synthetic traces in the following categories:

1. **Comparison between multi-path and single path schemes.** Both schemes do not use the shortcut.
2. **Comparison between non-shortcut and shortcut schemes.** Both schemes in this category do not consider the process after the packet arrives at the destination group.

Then, we compare the performance of the proposed hypercube-based social feature routing scheme with several state-of-the-art ones, including spray-and-wait [53] and spray-and-focus [54].

### Simulation Datasets

In the simulation, we make comparisons of the performance of the routing schemes, both in real and synthetic traces.

1) *Real trace*: we use two real traces, the *Infocom 2006* trace [61, 64] and the *MIT reality mining* data [86], in our simulation.

2) *Synthetic trace*: we assume that the two groups in  $i$  feature distances meet according to a uniform exponential distribution with mean time  $\frac{1}{\lambda_i}$ . Since people contact each other more frequently if they have more social features in common, we assume that  $\lambda_i = 2 * \lambda_{i+1}$ . The contact rate between individuals in the same group ( $\lambda_0$ ) is  $2 * \lambda_1$ . The contact rate of pairwise individuals corresponds to the properties of their belonging groups. We create 64 individuals and 50,000 time slots in seconds. Contacts are randomly selected from these time slots based on their contact rate.

The estimation of  $\lambda_1$  is according to the real traces. For example, in the case of a 3-dimensional hypercube, as shown in Fig. 5.2, there are 12 edges. If the contact rate of each edge  $e_i$  is  $\lambda_{e_i}$ , the estimated  $\lambda_1$  in the synthetic trace is:

$$\frac{1}{\lambda_1} = \frac{1}{12} \sum_{i=1}^{12} \frac{1}{\lambda_{e_i}}. \quad (5.16)$$

In the simulation, we will compare two performance metrics: delivery rate and latency.

### Simulation Methods

We implement and compare several routing schemes in the simulation. In all schemes, we consider the following:

1) **Hypercube-based social feature routing with wait-at-destination (HSFR-W)**: Waiting for the destination after the packet enters the destination group in hypercube-based social feature routing.

2) **Hypercube-based social feature routing with focus-at-destination (HSFR-F)**: Forwarding the packet to higher active level nodes after the packet enters the destination group in

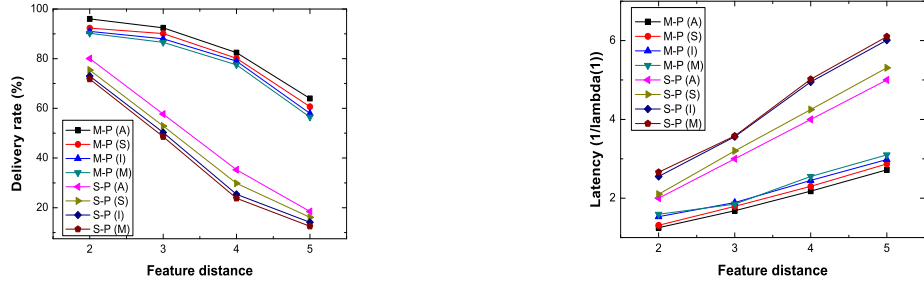


Figure 5.19: Comparison of the performance between *multi-path* and *single path*: (L): delivery rate and (R): latency. (M-P: multi-path, S-P: single-path; A: analysis, S: synthetic, I: Infocom, and M: MIT.)

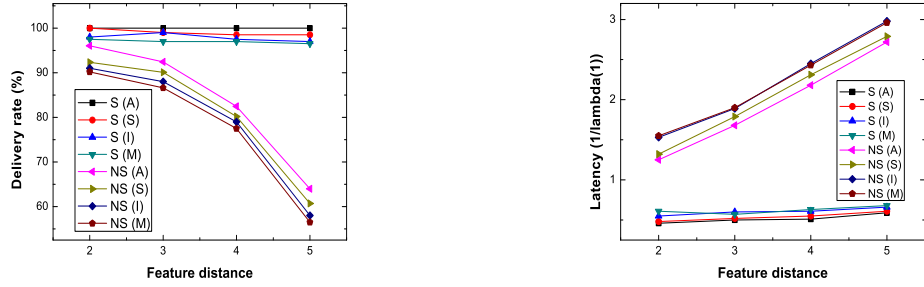


Figure 5.20: Comparison of the performance between *shortcut* and *non-shortcut*: (L): delivery rate and (R): latency. (S: shortcut, NS: non-shortcut.)

hypercube-based social feature routing.

3) **Spray-and-wait (S-W)** [53]: *Spray* phase: any node with copies will forward half of the copies to the encountered node with no copy; *Wait* phase: if the destination is not found in the spray phase, the copy carriers wait for the destination.

4) **Spray-and-focus (S-F)** [54]: *Spray* phase: same as S-W; *Focus* phase: if the destination is not found in the spray phase, the copy carriers forward the copy to the encountered node with a smaller feature distance to the destination.

5) **SimBet** [34]: The packet forwarding is based on the SimBet utility, which is a combination of locally determined social *similarity* to the destination node and pre-estimated *betweenness* centrality metrics.

The first two schemes use hypercube routing, in which the multiple paths are node-disjoint. In the other three schemes, the multiple paths may cross each other, which may reduce the efficiency of the routing process.

### Comparison of multi-path and single path schemes

In this part, we will show the benefits of the multi-path scheme. From Fig. 5.19, we find that both analysis and simulation results show that multi-path routing has a higher delivery rate and a

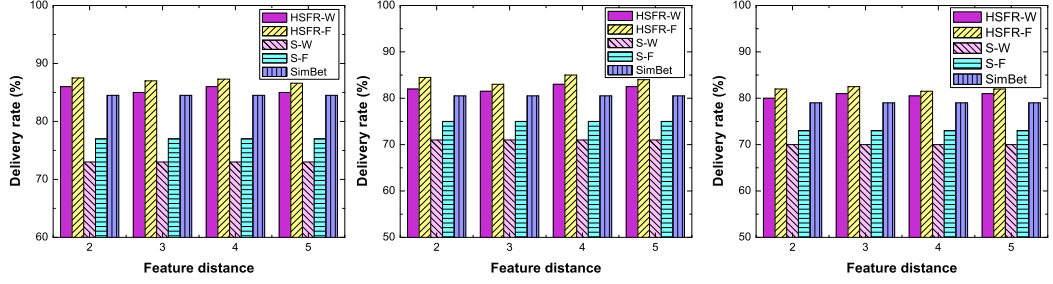


Figure 5.21: Comparing the delivery rate: (*L*): synthetic, (*Center*): Infocom, and (*R*): MIT.

smaller latency compared to the single path scheme. Multi-path routing increases delivery rate by about 115%, and decreases latency by about 45%. The simulation results are consistent with the analysis results. The difference between the analysis and synthetic simulation results is about 3% in the delivery rate and 5% in the latency. The results from the real traces reduce the delivery rate by about 6%, and increase the latency by about 15% compared to the analysis results. There are two reasons that the real trace simulation results are a little bit different from the analysis results. Firstly, the estimated contact rates ( $\lambda$ ) are not very accurate, because we assume that the pairwise contact rates are all the same in the analysis part; however, in the real trace, they are different. Secondly, in the simulation part, there are multiple nodes in each group, but we assume that there is only one node in one group in our analysis.

### Comparison of shortcut and non-shortcut schemes

Our simulation demonstrates the importance of the shortcut fast feature matching in hypercube-based routing. Fig. 5.20 shows the performance of the shortcut and non-shortcut schemes in the proposed hypercube routing. Both analysis and simulation results show that using the shortcut fast feature matching process can significantly increase the delivery rate and reduce the latency. In the synthetic trace, using the shortcut can increase the delivery rate by about 15%, and reduce the latency by about 60%. When the feature distance increases, the improvement also increases. The results in the real traces show the same phenomenon. The simulation results are consistent with the analysis results, especially in the synthetic trace. From the synthetic trace results, we see that it decreases the delivery rate by about 2%, and increases the latency by about 4%. The results from two real traces show that it decreases the delivery rate by about 5%, and increases the latency by about 15% compared to the analysis results.

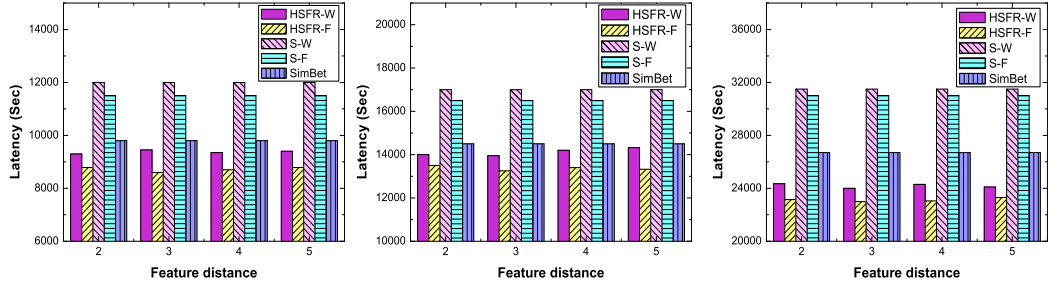


Figure 5.22: Comparing the latency: (*L*): synthetic, (*Center*): Infocom, and (*R*): MIT.

### Comparison with other state-of-the-art routing protocols

We compare the performance of hypercube-based routing with spray-and-wait, spray-and-focus, and seek-active in both synthetic and real traces. From Figs. 5.21 and 5.22, we find that hypercube-based routing performs better than other schemes. In the synthetic trace, HSFR-W increases the delivery rate by about 17% and reduces the latency by about 22%, compared to spray-and-wait. HSFR-F increases the delivery rate by about 14% and reduces the latency by about 24%, compared to spray-and-focus. In real traces, the proposed hypercube-based social feature increases the delivery rate by about 14% and 13%, respectively, in the Infocom and MIT reality mining traces, compared to spray-and-wait and spray-and-focus. The latency can be reduced by about 18% and 24%, respectively. In both synthetic and real traces, hypercube-based routing has a higher delivery rate and smaller latency compared to the SimBet scheme. After the packet enters the destination group, using *focus* can improve the performance, as compared to using *wait*. From Figs. 5.21 and 5.22, we can see that HSFR-F increases the delivery rate by about 2% in the synthetic trace, 3% in the Infocom trace, and 2.5% in the MIT reality mining trace compared to HSFR-W. HSFR-F reduces the latency by about 7% in the synthetic trace, 5% in the Infocom trace, and 4% in the MIT reality mining trace compared to HSFR-W, in Figs. 5.21 and 5.22.

### Summary of Simulation

We first verify the analysis results in the simulation. Then, comparisons with other state-of-the-art DTN routing protocols are presented, both in synthetic and real traces.

By comparing the simulation results to the analysis results, we find the consistency between these two. Because of the estimation deviation for the contact rate, the simulation results reduce the delivery rate, and increase the latency by a little. Another reason the latency is a little bit different in the simulation results is because, in simulation, there are multiple nodes in one group, which is different from the analysis assumption.

Our simulation concludes that, compared with spray-and-wait, spray-and-focus, and SimBet

schemes, the hypercube-based social feature routing scheme has a significantly higher delivery rate and reduced latency. The hypercube-based routing is a multi-path routing scheme, in which there are multiple node-disjoint paths seeking the destination that helps to improve search efficiency. The shortcut fast feature matching process can increase the delivery rate and reduce the latency. When the packet enters the destination group, forwarding the packet to the active relay nodes can improve the performance, as compared to waiting for the destination node.

### 5.2.5 Conclusion

In this section, we provide an analytical model for a hypercube-based social feature routing scheme in DTNs, which converts the routing problem from a high mobility space (M-space) into a static feature space (F-space). Hypercube-based routing is a feature matching process where the feature difference between the source and the destination is resolved step-by-step. The shortcut algorithm is used for fast feature matching, where the feature distance can be reduced more than one at one time. The shortcut also can guarantee the node-disjointness of these multiple paths, which can improve the efficiency of the routing process. In the analysis section, we formally analyze the delivery rate and latency of the hypercube-based routing. Exact solutions are presented in multi-path and single-path cases when there is no shortcut. If shortcuts are included, recursive formulas are introduced for both multi-path and single-path cases. We prove that multi-path scheme has better performance than single-path scheme. Shortcut can increase the delivery rate and reduce the latency. Trace-driven simulation results show that our proposed hypercube-based routing scheme performs better than the spray-and-wait, spray-and-focus, and SimBet schemes. The simulation results also verify the accuracy of our analysis results. We believe that the social features will play an important role in DTN routing.

## 5.3 Conclusion of the Chapter

In this chapter, we introduce a novel social feature-based hypercube routing, which is a multi-path scheme. Then, we provide an analytical model for this proposed scheme by studying the delivery rate and latency of the routing process.

In the next chapter, we will study the resource management problem in DTNs. We will present a joint replication-migration approach for congestion control.

## Chapter 6

# RESOURCE MANAGEMENT IN DTNS

The limited buffer space of the mobile devices leads to the congestion problem when implementing the multiple copy routing in DTNs. When the relay nodes lack sufficient buffer, the routing will be delayed until it encounters some available nodes. Our goal here is to reduce the buffer cost, while seizing enough relay nodes and better utilizing their available buffer, in order to enhance the routing speed and success rate of message delivery. In this chapter, we propose a congestion control scheme in DTNs called *joint replication-migration*. We will show the details in Section 6.1.

### 6.1 A Joint Replication-Migration-based Routing

Delay tolerant networks (DTNs) use mobility-assisted routing, where nodes carry, store, and forward data to each other in order to overcome the intermittent connectivity and limited network capacity. In this section, we propose a routing protocol that includes two mechanisms: *message replication* and *message migration*; the former one is to speed up the message propagation, while the latter one is to make the congestion message active at the maximum. Each mechanism has two steps: *message selection* and *node selection*. In message replication, we choose the smallest hop-count message to replicate. We propose a metric called *2-hop activity level* to measure the relay node's transmission capacity, which is used in node selection for message replication. In the message migration, we select the largest hop-count message to migrate to an alternative node with enough buffer space. The node selection is based on the node's community-based mobility pattern, in that the nodes with similar community properties will contact each other more frequently. A novel community property estimation algorithm is presented in this thesis. We validate our protocol via extensive simulation

experiments. We use a combination of synthetic and real mobility traces.

### 6.1.1 Introduction

Spyropoulos et al. claimed that multiple-copy routing performs more efficiency than single-copy routing in DTNs [100]. However, the limited buffer space of the mobile devices leads to the congestion problem when implementing the multiple copy routing. When the relay nodes lack sufficient buffer, the routing will be delayed until it encounters some available nodes. In this report, we propose a joint replication-migration-based routing scheme that includes a *message replication* phase: replicating the message to other nodes and taking their chances to encounter the destination quickly; and a *message migration* phase: migrating the message to an alternative node with more buffer space so that the buffer used in message delivery in the entire network can be balanced, and the congestion problem in routing for a quicker path can be avoided. Our work is to determine the timing of each phase, measured by the proposed metric. The substantial improvement of our new routing with these different phases is demonstrated by experimental work with real trace data.

In this section, we use a metric, *2-hop activity level*, to control message replication during a contact. The notion of the activity level is based on the observation that an active node has a higher chance of contacting more nodes later to improve the routing performance. We use 2-hop neighborhood information (or simply 2-hop information) to predict the node's activity level, which combines local information and the encountered nodes' information. Our message replication scheme is a multiple-copy model based on encounter history that considers the 2-hop information, so that destinations can be reached quickly with a high delivery rate. There are two steps controlling the message replication: *message selection* (selecting the message with the smallest hop-count as the highest priority to replicate) and *node selection* (selecting the best relay node to carry, store, and forward the message).

We use message hop-count to prioritize the messages in the buffer. The *hop-count threshold* ( $T_h$ ) is used to control the message replication speed. In other words, if the hop-count of the selected message is smaller than the threshold, the message will be replicated directly, in order to spray the message quickly. Otherwise, we will use the node activity level to select a good relay node to replicate, in order to control the number of copies. When the threshold is low, the replication speed is slow. Hence, it has a higher latency. When the threshold is high, it performs significant replications, which may lead to buffer congestion when the buffer space is limited. We evaluate the performance by using message replication scheme in a synthetic trace of 20 nodes. Each of these nodes has a 10-message buffer. As can be seen in Fig. 6.1, the delivery rate and latency functions are both quadratic. According to the increasing of the hop-count threshold, the delivery rate will increase

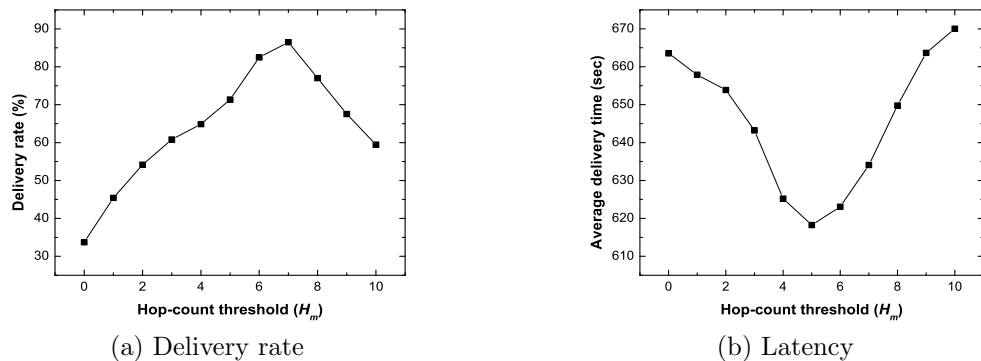


Figure 6.1: Comparison delivery rate and latency in different hop-count thresholds using message replication.

first, and then decrease. The optimal hop-count threshold for delivery rate is 7, in Fig. 6.1(a). For the average delivery time, we find that it decreases until the hop-count threshold reaches 5. After that, the latency increases, as shown in Fig. 6.1(b). We find that the optimal hop-count thresholds ( $T_h$ ) are different for maximizing the delivery rate or minimizing the latency.

Although a large amount of effort has been invested in the design of buffer management policies for DTNs [45–47], most of them deal with choosing the appropriate message to discard when the available buffer space is below a threshold ( $T_b$ ). In this report, we propose a novel congestion control scheme, called *message migration*, which enables the congestion message left over by replication at the maximum. This scheme initiates an early migration to an alternative node when the available buffer space is below  $T_b$ . We also consider the community properties of the nodes by learning from the nodes’ mobility patterns. The nodes in the same community contact each other more frequently. Our message migration policy has two parts: *message selection* (selecting the lowest priority message to migrate) and *node selection* (selecting an alternative node to store the message by considering the buffer space condition and the community property of the alternative node). We propose a novel community property estimation algorithm in this thesis.

### 6.1.2 Replication-Migration-based Routing

In this section, we present the details of the protocol: message selection and message migration. When nodes  $a$  and  $b$  come in contact with each other, if  $b$  is the destination of the message in the buffer of  $a$ ,  $a$  will forward the message to node  $b$ . In the remaining part of this section, we mainly consider the scenario where  $b$  is not a destination.

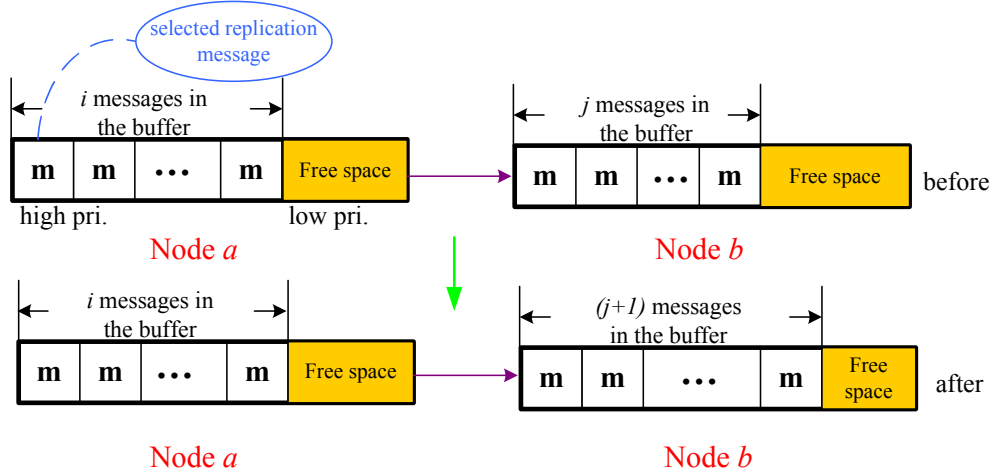


Figure 6.2: An illustration of message replication.

### Neighbor Discovery

Nodes must discover another node in their communication range before transferring a message. Then, two encountered nodes will exchange their metadata. In our protocol, the node will send the following information to the encountering node: (1) information about neighboring nodes that have been encountered before this contact; (2) the message information stored in the buffer; (3) its buffer usage status; and (4) the community table.

### Message Replication

In our message selection stage, we use *hop-count* ( $H_m$ )<sup>1</sup> as the priority metric to select the smallest hop-count message  $m$  as the candidate for the replication message in the contact. We adopt the message ID to break the tie if any.

To reduce the message delivery time, we propose a hop-count threshold ( $T_h$ ) to control the message replication speed. If the hop-count of the selected message is smaller than  $T_h$ , we believe that it has not traveled far in the network. Hence, we will replicate this message to the relay node without any delay. In this way, a message will be replicated more often when it is relatively new in the networks, increasing the opportunities to reach the destination. Otherwise, if the hop-count is larger than  $T_h$ , we will go to the replication node selection stage to decide whether to replicate the message to the encountered node.

The node selection is based on the 2-hop activity level. Next, we present the definition of the 2-hop activity level. The 2-hop activity level of a node is the combination of its own encounter history, and its neighbors' encounter histories before this contact. Node  $a$ 's activity level ( $A_a$ ) can be calculated as follows:

<sup>1</sup>Hop-count is the number of forwardings that the message  $m$  has made.

---

**Algorithm 13** Message Replication (at node  $a$ )

---

Nodes  $a$  and  $b$  first exchange the metadata.  
 $a$  selects the smallest hop-count message ( $m$ ) to replicate.  
**if**  $H_m < T_h$  **then**  
     $a$  replicates  $m$  to  $b$ .  
**else**  
    **if**  $A_b > A_a$  **then**  
         $a$  replicates  $m$  to  $b$ .  
    **else**  
         $a$  replicates  $m$  to  $b$  with probability  $A_b/A_a$ .  
    **end if**  
**end if**

---

$$A_a = cE_a + (1 - c) \sum_{k=1}^i w_k E_k, \quad (6.1)$$

where  $E_a$  is node  $a$ 's total number of contacts with other nodes in the network over a certain time period<sup>2</sup>.  $w_k$  is neighbor  $k$ 's contribution to  $a$ 's activity level, which is the ratio of  $k$  that appeared in node  $a$ 's encounter list.  $c$  is a constant value to represent the weight of the neighbors' contributions. For example, node  $a$  has encountered two other nodes,  $x$  and  $y$ , 2 and 3 times, respectively. By the definition,  $E_a = 5$ , and we assume that  $E_x = 8$  and  $E_y = 6$ . If we set  $c$  to 0.8, and by using Eq. (6.1), we can get the activity level of  $a$ :  $A_a = 0.8 \times 5 + 0.2 \times (\frac{2}{5} \times 8 + \frac{3}{5} \times 6) = 4.68$ .

When nodes  $a$  and  $b$  are in each other's communication range, if the hop-count of the selected message  $m$  ( $H_m$ ) is smaller than  $T_h$ ,  $a$  replicates  $m$  to  $b$ . Otherwise, they will compare their activity levels. If node  $b$ 's activity level  $A_b$  is larger than node  $a$ 's activity level  $A_a$ ,  $a$  replicates  $m$  to  $b$ . Otherwise,  $a$  replicates  $m$  to  $b$  with probability  $A_b/A_a$ . If  $b$ 's buffer is full, the existing method usually discards the largest hop-count message in the buffer of  $b$ , including the replicated message from  $a$ . Fig. 6.2 illustrates the process of message replication. The message replication process is shown in Algorithm 13 when nodes  $a$  and  $b$  are in contact.

### Message Migration

Although there are many works that have designed message dropping policies, we believe that using message dropping policies will reduce the delivery rate in the following situation: if the buffer size is relatively small, the buffer of an active node will be overloaded quickly. In this situation, the messages in the buffer may all have a relatively small hop-count. Hence, dropping any message in the buffer will decrease the delivery rate.

Therefore, we propose a message migration scheme to control the buffer congestion. When the available buffer space is below a threshold ( $T_b/S_a > 1$ ), the node will migrate a message to an

---

<sup>2</sup>Although contact duration is also important, results in [98] show that there are high correlation coefficients of contact duration and contact times in many traces; we simply consider only contact times here.

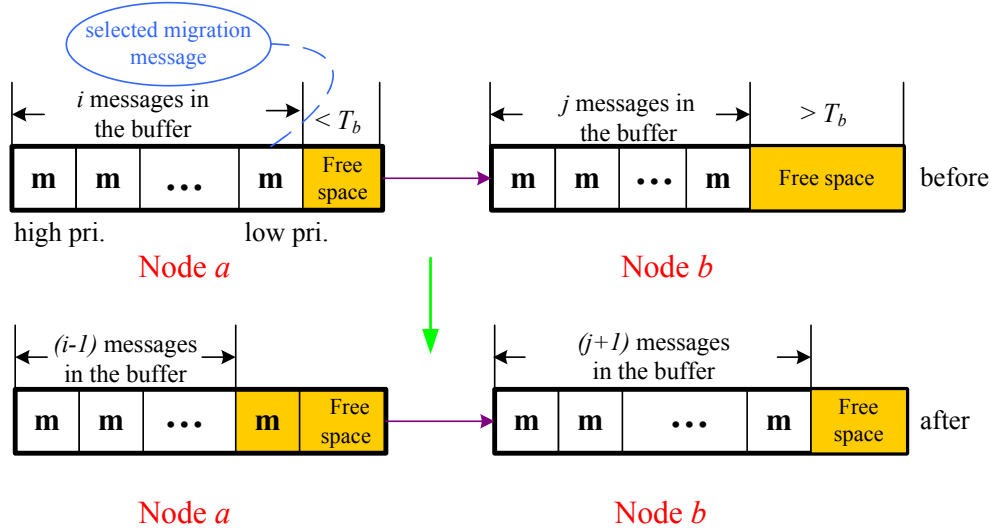


Figure 6.3: An illustration of message migration.

alternative node that has enough buffer space early on. In the message migration process, the sending node will forward the largest hop-count message to the alternative node, and will not retain a copy in its buffer.

We select the largest hop-count message to migrate. This is because a message with a larger hop-count has a smaller opportunity to be replicated in the congestion node.

In order to make sure that the message will not be discarded quickly after the migration, the selected migration message will be migrated to an alternative node with enough buffer space ( $S_b - 1 \geq T_b$ ).

We use Fig. 6.3 to illustrate our message migration scheme. There are two requirements for selecting an appropriate migration node: *buffer space condition* and *community property*.

When the available buffer space of node  $a$  ( $S_a$ ) is below the buffer space threshold  $T_b$  of the message dropping schemes,  $m$  will be discarded to release the buffer space for new messages. However, it will reduce the delivery probability to the destination, and will increase the delivery time because of the message loss. By using the message migration scheme, node  $a$  migrates  $m$  to node  $b$  with enough buffer space ( $S_b - 1 \geq T_b$ ) before its own buffer becomes full. After migration,  $a$  will have more buffer space to receive new messages while  $b$  is carrying this message, and it will continue to replicate or migrate to its encountered nodes.

In this report, we use the community properties of the nodes to enhance the migration node selection. The node's community property is learned from the node's historical mobility pattern. We assume that node  $a$  is the congestion node to migrate messages, and node  $b$  is a candidate node with enough buffer space, as discussed above. The destination of the message is node  $d$ . If nodes  $b$  and  $d$  belong to the same community, the migration probability is  $p$ . Otherwise, the migration probability

---

**Algorithm 14** Message Migration (at node  $a$  with respect to the message destination  $d$ )

---

Nodes  $a$  and  $b$  first exchange the metadata.

**if**  $S_a < T_b$  and  $S_b - 1 \geq T_b$  **then**

**if** Nodes  $b$  and  $d$  belong to the same community **then**

        Node  $a$  selects the largest hop-count message to migrate to node  $b$  with probability  $p$ .

**else**

        The migration probability is  $1 - p$ .

**end if**

**end if**

---

Table 6.1: Encounter history.

	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$
<b>Encounter History</b>	3	2	4	1	6

is  $1 - p$ . Here,  $p$  is larger than 0.5.

In recent research, it has been shown that a human’s mobility model follows the community-based mobility pattern [72, 101]. In this section, we suppose that every node has  $C$  communities. For simplicity, each node has two major communities 1 and 2, and one minor community, 3. For instance, a professor can travel to other places for conferences occasionally, but he/she mainly stays at home or works at school every day. The professor’s family members and other faculty members at school form the major communities, and the people the professor meets at the conference form the minor community.

In our proposed migration node selection process, based on our observation, the node with the same community as the message destination will be a good alternative node to carry, store, and forward the migration message. Hence, we give higher priority to these nodes.

We propose a novel *node community property estimation algorithm* in this report. Each node has a *community table*, which includes the probability of itself and the encountered nodes which appeared in each community. Initially, the probability of node  $a$  in each community is equal, and is set to  $\frac{1}{C}$ . After node  $a$  has a contact with another node in community  $i$ , the probability value in community  $i$ ,  $p_i^a$ , is incremented by 1; then, all values of  $p$  are re-normalized. This is called *incremental averaging*. Each time two nodes meet, they exchange the community table with each other.

For example, a DTN has 3 communities; initially, the probability values of node  $a$  are  $p_1^a = p_2^a = p_3^a = \frac{1}{3}$ . After encountering another node in community 2,  $p_2^a$  is incremented to  $\frac{4}{3}$ . Then, by re-normalizing all probability values so that they add up to 1 again, we get  $p_1^a = p_3^a = \frac{1}{6}$  and  $p_2^a = \frac{2}{3}$ .

The message migration process is shown in Algorithm 14, when nodes  $a$  and  $b$  are in contact. When either node  $a$  or  $b$  is overloaded, we do the message migration first.

We use Fig. 6.4 as an example. Fig. 6.4(a) shows the contact table, which includes the contacted nodes in each time step. We can conclude an encounter history table, which indicates the number

<b>Node contact table</b>		<b>Message generation</b>		
Node ID	Node ID	Message ID	Destination ID	
1	$N_1$	$N_5$	$m_1$	$N_2$
2	$N_2$	$N_3$	$m_2$	$N_1$
3	$N_4$	$N_5$	$m_3$	$N_3$
4	$N_1$	$N_3$	$m_4$	$N_2$
5	$N_2$	$N_5$		
6	$N_3$	$N_5$		
7	$N_2$	$N_5$		
8	$N_3$	$N_5$		

(a) Contact Table

<b>Replication- Migration</b>		<b>Replication Only</b>	
Node ID: Message ID(s)		Node ID: Message ID(s)	
$N_1: m_1$	$N_5: m_1$	$N_1: m_1$	$N_5: m_1$
$N_2: m_2$	$N_3: m_2$	$N_2: m_2$	$N_3: m_2$
$N_4: m_3$	$N_5: m_3, m_1$	$N_4: m_3$	$N_5: m_3, m_1$
$N_1: m_1, m_4$	$N_3: m_1, m_4$	$N_1: m_1, m_4$	$N_3: m_1, m_4$
$N_2: m_2, m_3$	$N_5: m_1$	$N_2: m_2$	$N_5: m_1$
...	...	...	...

(b) Comparison

Figure 6.4: An example of the replication-migration-based scheme and replication scheme comparison: there are 5 nodes in the network. All nodes have the same buffer size, which can contain 2 messages.

Table 6.2: Simulation parameters.

	<b>Synthetic</b>	<b>Real</b>
Number of nodes	20	9
Buffer size	20MB-100MB	20MB-100MB
Average transfer size	10	10
Size of the message	1MB	1MB
Number of contacts	2,563	2,766
Hop-count threshold	0 - 10	0 - 6

of contacts the node encountered with other nodes, as shown in Table 6.1. Four messages generate from time 1 to time 4, tagged with the destination ID. Fig. 6.4(b) shows the message storing status after each contact. We find that by only using the replication scheme, message  $m_3$  will be stored in node  $N_4$ , which is an inactive node; hence, message  $m_3$  cannot be delivered to the destination. Therefore, the replication-migration-based scheme can increase the delivery rate, as compared to the replication-only scheme.

### 6.1.3 Implementation and Simulation

In order to evaluate our designed routing protocol, we have performed a number of simulations. In particular, we have taken a synthetic community-based mobility trace to test. We then test our protocol in real mobility traces collected by the Intel Research Laboratory in Cambridge [50].

#### Simulation Methods and Setting

We implement our routing protocol both in synthetic and real mobility traces using MATLAB.

- **Synthetic trace** In the synthetic mobility models, we set up a 20-node environment. The mobility pattern of the nodes is followed by the random waypoint model. There are 2,563 time slots in seconds, and in each time slot, there is a contact between two nodes.
- **Real trace** We evaluate our schemes in the Intel trace [50], which includes Bluetooth sightings by groups of users carrying small devices (iMotes) for six days in the Intel Research Cambridge Corporate Laboratory. There is a stationary node, and 8 nodes that correspond to mobile iMotes. There are 2,766 contacts between these nodes over a period of 359,311 time slots in seconds.

In our simulation, we analyzed four specific scenarios:

**Scenario 1:** comparisons of delivery rate, latency, number of forwardings, and number of lost messages between the joint replication-migration scheme and the replication scheme in different buffer space thresholds.

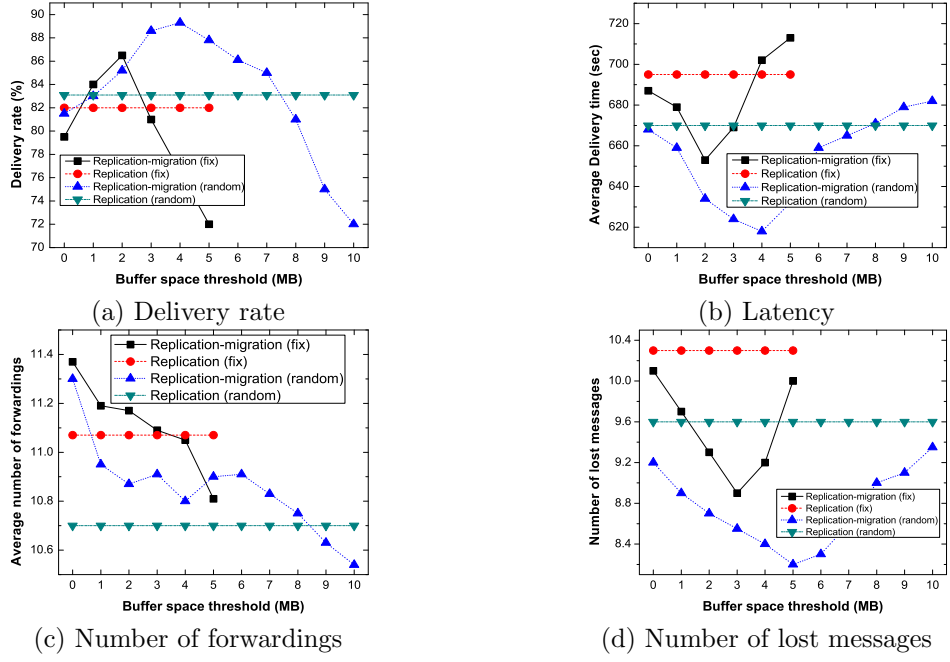


Figure 6.5: Comparison between migration and dropping in different buffer space thresholds in the synthetic trace.

**Scenario 2:** comparisons of delivery rate, latency, number of forwardings, and number of lost messages between the joint replication-migration scheme and the replication scheme in different hop-count thresholds.

**Scenario 3:** number of forwardings and latency comparisons between using 2-hop neighbor information and just using 1-hop information (its own activity level).

**Scenario 4:** comparisons of delivery rate, latency, number of forwardings, and number of lost messages between the methods by using community properties, and without using community properties, in different buffer space thresholds.

In our simulation, the message size is 1MB. The message generation rate is 1 contact per message. There are a total of 20 messages. In Scenario 1, we evaluate under two conditions: *fixed buffer size* (the buffer size of all nodes is 10MB) and *random buffer size in a range* (each node will assign a random buffer size in a range (0 to 20MB)).  $T_b$  varies from 0MB to 5MB in the fixed buffer size condition, and 0MB to 10MB in the random buffer size condition. In Scenario 2, we assume that the buffer size is infinite. In Scenario 3, the hop-count threshold varies from 0 to 10 in the synthetic trace, and 0 to 6 in the real trace. In Scenario 4, we use random buffer size in a range which is the same as we discussed above in Scenario 1. In our simulation, we find that by using a different  $c$  in Eq. 6.1, the results have the same trend. Hence, we set  $c$  to 0.5 in this section.

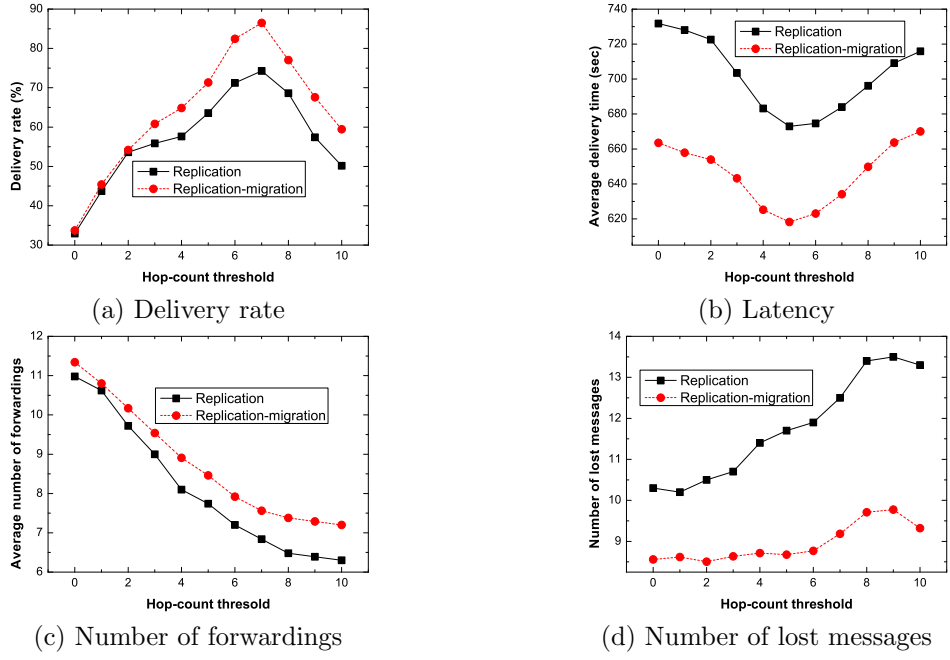


Figure 6.6: Comparison of delivery rate, latency, number of forwardings, and number of lost messages in different hop-count thresholds in the synthetic trace.

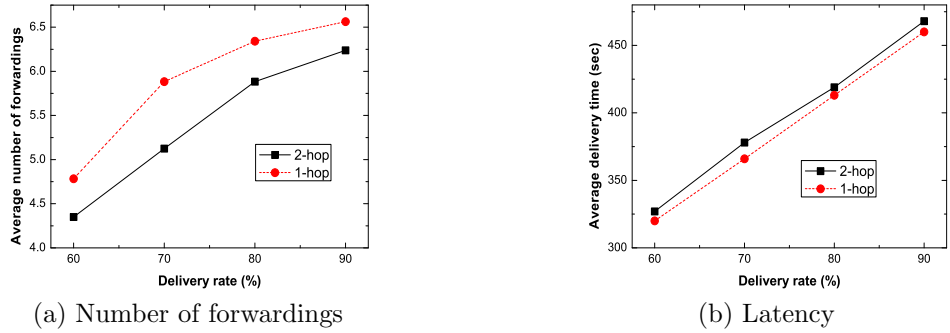


Figure 6.7: Comparison between 2-hop and 1-hop in the synthetic trace.

## Results in Synthetic Trace

*Scenario 1:* in both the fixed and random buffer size conditions, we find that in an appropriate buffer space threshold, the joint replication-migration scheme can increase the delivery rate and decrease the latency, as compared to the scheme using the message replication policy only in Fig. 6.5. In Fig. 6.5(a), we find that in the fixed buffer size condition, the joint replication-migration scheme increases by the most, as compared to the replication scheme when the buffer space threshold is 2 in delivery rate. In the random buffer size condition, the joint replication-migration scheme performs better than that applied in the fixed buffer size condition, which can have a 7.5% increment in the delivery rate when the buffer space threshold is 4. The joint replication-migration scheme can dramatically reduce the number of lost messages in all buffer space thresholds in both conditions.

*Scenario 2:* from Fig. 6.6(a), we find that using the joint replication-migration policy results in

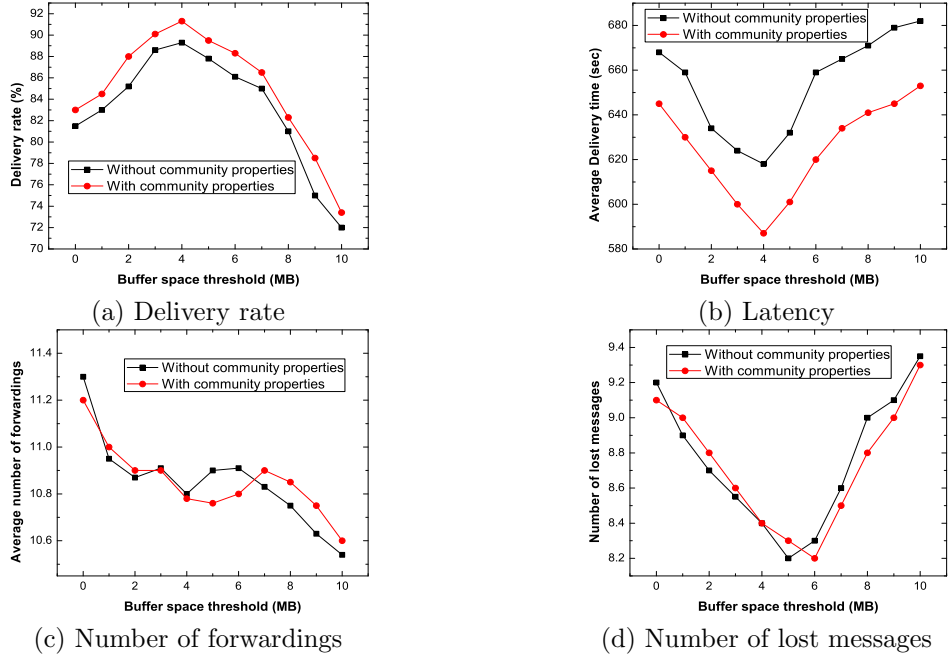


Figure 6.8: Comparison between the methods by using community properties, and without using community properties, in different buffer space thresholds in the synthetic trace.

a better delivery rate, compared to using the replication policy only. When the hop-count threshold is 5, the improvement is the best at about 20%. The joint replication-migration scheme reduces the latency overall by about 5%, as shown in Fig. 6.6(b). When the threshold is 0, the decreased ratio is about 6.5%, which is the highest one. The lowest one occurs when the threshold is equal to 9. The number of forwardings reduces when the hop-count threshold increases. Using the joint replication-migration policy increases the number of forwardings by about 9% compared to the replication scheme in Fig. 6.6(c). By using the joint replication-migration policy, there are fewer messages lost in Fig. 6.6(d).

*Scenario 3:* from Figs. 6.7(a) and 6.7(b), we can see that using 2-hop information can reduce the number of forwardings by about 9.5% while the increment of latency can be controlled within 2%. Therefore, using 2-hop information outperforms using 1-hop information in the synthetic trace. From the simulation results, we find that the delivery rate is similar when using 2-hop and 1-hop neighborhood information to calculate the activity level.

*Scenario 4:* from Fig. 6.8, we find that by using the community property to enhance the message migration scheme, the results in a better performance. The delivery rate can be increased by about 2% in Fig. 6.8(a), and the latency can be reduced by about 5% in Fig. 6.8(b). At the same time, it will not increase the number of forwardings and the number of lost messages as compared to the no community information scheme in Figs. 6.8(c) and 6.8(d).

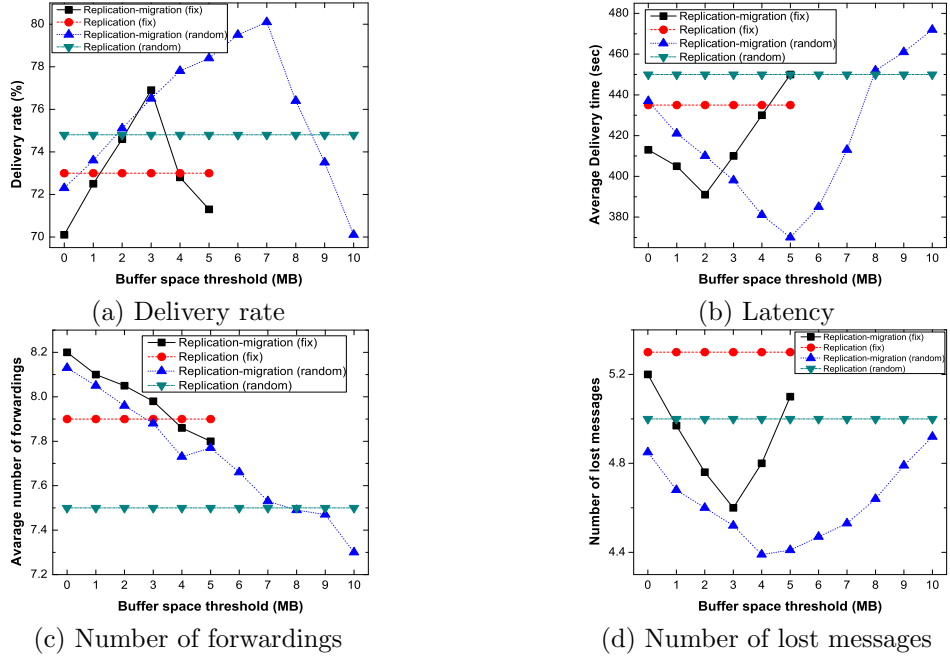


Figure 6.9: Comparison between migration and dropping in different buffer space thresholds in the real trace.

## Results in Real Trace

*Scenario 1:* in the real trace, we find the same results as in the synthetic trace; when choosing an appropriate buffer space threshold, the joint replication-migration scheme can significantly increase the delivery rate and decrease the average delivery time, in both the fixed buffer size condition and the random one in Fig. 6.9. In Fig. 6.9(d), the joint replication-migration scheme can reduce the number of lost messages by about 13.6% in the fixed buffer size condition, and by 14.6% in the random buffer size condition.

*Scenario 2:* as shown in Fig. 6.10(a), we find that when the hop-count threshold is 3, the joint replication-migration scheme has the best improvement over the replication scheme – about 25%. The joint replication-migration scheme decreases the latency overall by about 4% in Fig. 6.10(b). When the threshold is 0, the decreased ratio is about 6%, which is the highest one. The lowest one happens when the threshold is equal to 6. The number of forwardings reduces when the hop-count threshold increases in the real trace, and using the joint replication-migration policy results in having a slightly larger number of forwardings, as compared to using replication only in Fig. 6.10(c). From Fig. 6.10(d), we have the same conclusion as in the synthetic trace. When the hop-count threshold is larger, using the joint replication-migration policy results in a much better performance.

*Scenario 3:* in Figs. 6.11(a) and 6.11(b), we compare the number of forwardings and latency between using 2-hop and 1-hop neighborhood information. We find that by using 2-hop information, the average number of forwardings can be reduced by about 7%, as is seen in Fig. 6.11(a). Note that

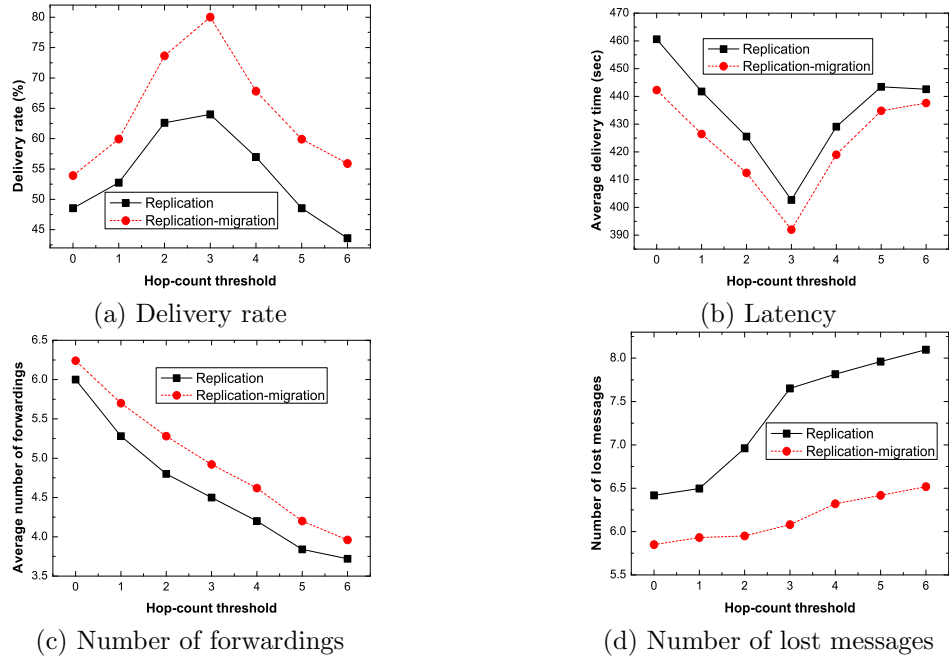


Figure 6.10: Comparison of delivery rate, latency, number of forwardings, and number of lost messages in different hop-count thresholds in the real trace.

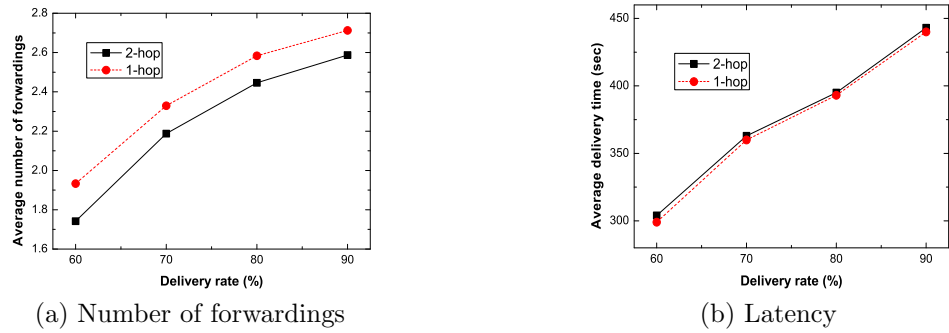


Figure 6.11: Comparison between 2-hop and 1-hop in the real trace.

these two methods have a similar latency.

*Scenario 4:* by comparing the methods between using community properties and without community properties, we have the same conclusions as the results in synthetic trace, as shown in Fig. 6.12. The delivery rate will increase, and latency will decrease, by using community properties to enhance the message migration scheme.

### Summary of Simulation

From the simulation results, we can see that the joint replication-migration scheme outperforms only the replication scheme when buffer congestion happens in various conditions. With an appropriate buffer space threshold, the joint replication-migration scheme can significantly increase the delivery rate and decrease the average delivery time and the number of lost messages in all conditions. In the

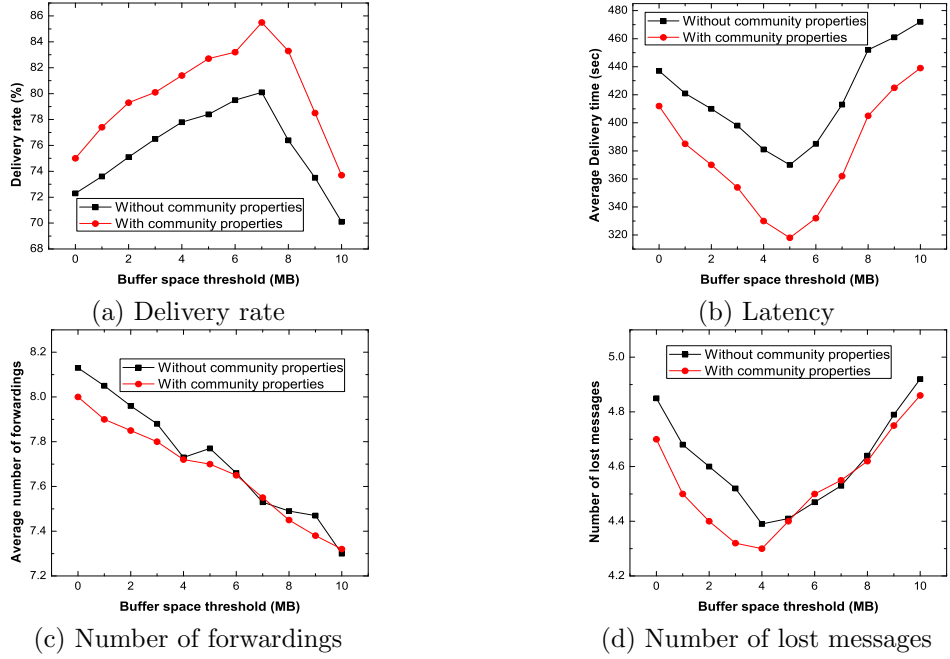


Figure 6.12: Comparison between the methods by using community properties, and without using community properties, in different buffer space thresholds in the real trace.

random buffer size condition, if the inactive nodes have a larger amount of buffer space to contain the migration messages from the active nodes, the buffer of the inactive nodes will be overloaded by the migration messages quickly. By comparing different hop-count thresholds, we find that when the threshold is too small, the message replication is based on the node selection, and the delivery rate decreases. When the threshold is too large, the message will replicate at every contact; thus, there will be massive copies of messages in the network. Because of the limitations in buffer space and bandwidth, the performance degrades because of the unnecessary copies of messages. Choosing different hop-count thresholds will also degrade the performance of the routing process. Using 2-hop information to calculate the activity level results in a better performance compared to just using 1-hop information.

#### 6.1.4 Conclusion

In this section, we proposed an efficient routing protocol for DTNs. The message hop-count was used to prioritize the messages stored in the buffer. We introduced the hop-count threshold to control the message replication speed. An inappropriate hop-count will affect the performance of the routing scheme. We used 2-hop information to predict nodes' activity levels, which is a metric that selects a good relay node for replication. To overcome the limited buffer size and bandwidth, we proposed the message migration policy to control buffer congestion. We considered the available buffer space threshold, and performed evaluations under the fixed and random buffer size conditions.

By considering the nodes' mobility patterns, we use the community properties of the nodes to guide the message migration. The joint replication-migration-based DTN routing scheme performs well in both synthetic and real traces. Our approach can increase the delivery rate, and decrease the average delivery time and the amount of lost messages in various conditions, although it slightly increases the average number of copies. There are many parameters that affect the hop-count threshold selection, such as buffer size, bandwidth, message generation speed, etc. In the future, we plan to address these issues.

## **6.2 Conclusion of the Chapter**

In this chapter, we present a joint replication-migration resource management scheme. The message hop-count was used to prioritize the messages stored in the buffer. 2-hop historical information is used to predict the popularity of the node.

In the next chapter, we will conclude this thesis and point out our future research plans.

## Chapter 7

# CONCLUSION

This thesis presents in-depth studies of the designs of novel DTN routing schemes. Due to the intermittent connection and uncertainty mobility pattern, designing efficient routing schemes becomes a challenge problem. In this thesis, we utilize the mobile nodes' mobility pattern, contact history information, and social feature information for routing protocols design. We summarize our contributions and point out several promising future research topics.

### 7.1 Summary of Contribution

Delay tolerant networks (DTNs) are wireless networks where most of the time there does not exist an end-to-end path between some or all of the nodes in the network. The nature of node contact is non-deterministic. These networks have a variety of applications including crisis environments, such as emergency response and military battlefields, vehicular communication, deep-space communication, and mobile social networks. This thesis aims at exploring the design and deployment of efficient routing protocols for DTN multicasting, broadcasting, as well as resource management. Background and existing arts are reviewed. Our designs are verified by large-scale simulations in both synthetic and real traces.

The thesis starts with a comprehensive literature review in Chapter 2. In Chapter 3, we propose three DTN multicast approaches: delegation forwarding multicast, non-replication multicast, and cloud-based multicast with feedback. Chapter 4 presents the data dissemination approaches we designed in DTNs. In Chapter 5, we studies social-aware DTN routing scheme. We design and analyze the hypercube-based social feature multi-path routing scheme. Chapter 6 presents a joint-replication-migration-based mechanism to solve the resource management problem in DTNs.

## 7.2 Future Research

With the growing amount of social data, ubiquitous systems, and mobile social media applications transcending everyday life, the analysis of social networks is receiving increased attention. While there is a large body of research concerning online social networks, important aspects of offline social networking still remains largely unexplored. My future research plan is to develop *ubiquitous mobile social systems* for research purpose. This ubiquitous mobile social networks can use different kinds of technology, such as RFID, Bluetooth, smart phone, to record the participants contact information, geographic information, phone call or message exchange information, and so on. We can collect the data in campus, conference, shopping mall, and research labs. Then, the collected data can be used in the following research area:

1. *Routing protocols*: exploiting various aspects of social behavior of mobile users to improve the performance in these mobile opportunistic social networks. Especially, developing efficiency multicasting and broadcasting schemes for smart phone users, which can also be validated in the developed ubiquitous mobile social systems.
2. *Social network analysis*: exploiting the participants social behavior, mobility, and community properties. For this research direction, we can collaborate with the professors from data mining, sociology, as well as business.
3. *Mobile cloud computer*: designing the system architecture and developing mobile phone applications for the mobile cloud computer system, and testing in the developed ubiquitous mobile social systems.

### 7.2.1 Routing Protocols

My plan is to design efficient routing protocols in mobile opportunistic social networks, and evaluated in the developed ubiquitous mobile social systems. In this research field, I plan to study in three directions:

1. *Contact-history-based routing*: using the collected contact history information, such as contact frequency, contact duration, contact geographic position, and so on, to predict the future contacts and enhance the routing process.
2. *Social-behavior-based routing*: using the knowledge of the social information of the participants for routing guidance. The social information includes social features, community properties, strong and weak ties, centrality, influential, etc.

3. *Mobility-based routing*: There are two important features of mobile networks. One is the mobility of the nodes resulting from the intrinsic nature of humans that compels them to travel with their devices from one location to another. Another one is that direct communication between any two devices is only possible when they are within transmission range of each other. These two features make such networks highly dynamic in terms of their connectivity and strongly dependent on human mobility patterns. Therefore, another direction of my future research is exploiting the human mobility and relationships models in mobile social networks.

### 7.2.2 Social Network Analysis

Computational social science is an emerging field that leverages the capacity to collect and analyze data at a scale that may reveal patterns of individual and group behaviors, which can benefit not only computer science, but also economics, sociology, psychology, finance, and management. In addition to traditional survey-based data, the real-life data can be gathered through smart phones in a cost-effective way in mobile social networks. Analyzing the social behavior of individuals and communities can synergistically improve the performance of emerging opportunistic mobile networking systems with intermittent network connectivity.

### 7.2.3 Mobile Cloud Networks

Mobile users carry ever increasingly powerful mobile devices that feature substantial processing power and memory along with a wide range of sensors. Moreover, these devices offer several wireless networking interfaces to access network infrastructure, to communicate directly with one another, and to interact with a multitude of external sensors. These capabilities can be used stand-alone but, more importantly, they may also be harnessed to support resource sharing: powerful smart devices can dynamically instantiate ephemeral mobile clouds within which they communicate spontaneously and efficiently to tackle a problem from cooperative sensing to content sharing to distributed data mining. Indeed, it is appealing to exploit the richness of diverse social, contextual, content, software, and hardware resources in these devices to create distributed applications that extend beyond the capabilities of any single device. My future plan is to design system to support distributed task execution in opportunistic mobile network, including leveraging human social behavior for efficient opportunistic interaction between a variety of sensors, personal communication devices and resources embedded in the local environment.

Computation offloading and energy-accuracy trade-off are becoming challenge problems in mobile cloud networks. Designing the system architecture and developing mobile phone applications for the mobile cloud computer system are two key aspects in this research field.

# Bibliography

- [1] K. Fall. A delay-tolerant network architecture for challenged Internets. In *Proc. of ACM SIGCOMM*, pages 27–34, 2003.
- [2] DTNRG. Delay tolerant networking research group. <http://www.dtnrg.org/>.
- [3] IPN. Interplanetary internet project, internet society special interest group. <http://www.ipnsig.org>.
- [4] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebrant. *SIGARCH Comput. Archit. News*, 30(5):96–107, 2002.
- [5] A. Pentland, R. Fletcher, and A. Hasson. Daknet: rethinking connectivity in developing nations. *Computer*, 37(1):78–83, 2004.
- [6] J.P. Singh, N. Bambos, B. Srinivasan, and D. Clavin. Huggle: A networking architecture designed around mobile users. In *Proc of the Third Annual IFIP Conference on Wireless On-demand Network Systems and Services (WONS)*, 2006.
- [7] J. Scott, P. Hui, J. Crowcroft, and C. Diot. Wireless LAN performance under varied stress conditions in vehicular traffic scenarios. In *Proc of IEEE 56th Vehicular Technology Conference (VTC)*, volume 2, pages 743 – 747, 2002.
- [8] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *Proc. of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking (WDTN)*, pages 244–251, 2005.
- [9] J. Santiago, A. Casaca, and P. Pereira. Multicast in delay tolerant networks using probabilities and mobility information. *Ad Hoc & Sensor Wireless Networks*, 7(1-2):51–68, 2009.
- [10] Q. Ye, L. Cheng, M. Chuah, and B. Davison. Performance comparison of different multicast routing strategies in disruption tolerant networks. *Comput. Commun.*, 32(16):1731–1741, October 2009.
- [11] W. Zhao, M. Ammar, and . Zegura. Multicasting in delay tolerant networks: semantic models and routing algorithms. In *Proc. of the ACM SIGCOMM workshop on Delay-tolerant networking (WDTN)*, pages 268–275, 2005.
- [12] W. Gao, Q. Li, B. Zhao, and G. Cao. Multicasting in delay tolerant networks: a social network perspective. In *Proc. of the tenth ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 299–308, 2009.
- [13] S. Symington, R. C. Durst, and K. Scott. Custodial multicast in delay tolerant networks. In *Proc. of IEEE Consumer Communications and Networking Conference (CCNC)*, pages 207–211, 2007.
- [14] U. Lee, S. Oh, K. Lee, and M. Gerla. Relaycast: Scalable multicast routing in delay tolerant networks. In *Proc. of IEEE ICNP*, pages 218 –227, 2008.

- [15] Z. Narmawala and S. Srivastava. Midtone: Multicast in delay tolerant networks. In *Proc. of the International Conference on Communications and Networking in China (ChinaCOM)*, pages 1–8, 2009.
- [16] M. Mongiovi, A.K. Singh, X. Yan, B. Zong, and K. Psounis. Efficient multicasting for delay tolerant networks using graph indexing. In *Proc. of the IEEE INFOCOM*, pages 1386–1394, 2012.
- [17] P. Yang and M. Chuah. Efficient interdomain multicast delivery in disruption tolerant networks. In *Proc. of the International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pages 81–88, 2008.
- [18] M. Abdulla and R. Simon. Controlled epidemic routing for multicasting in delay tolerant networks. In *Proc. fo IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems (MASCOTS)*, pages 1–10, 2008.
- [19] M. Chuah and P. Yang. Context aware multicast routing scheme for disruption tolerant networks. *Int. J. Ad Hoc Ubiquitous Comput.*, 4(5):269–281, July 2009.
- [20] Y. Li, G. Su, D.O. Wu, D. Jin, L. Su, and L. Zeng. The impact of node selfishness on multicasting in delay tolerant networks. *IEEE Transactions on Vehicular Technology*, 60(5):2224–2238, jun 2011.
- [21] A. Palma, P.R. Pereira, A. Casaca, and null. Multicast routing protocol for vehicular delay-tolerant networks. In *Proc. of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 753–760, oct. 2012.
- [22] S. Lo and N. Luo. Quota-based multicast routing in delay-tolerant networks. In *Proc. of International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 544–548, 2012.
- [23] K. Srinivasan and P. Ramanathan. Reliable anonymous multicasting in disruption tolerant networks. In *Proc. of the IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–5, 2008.
- [24] Q. Ye, L. Cheng, M. Chuah, and B.D. Davison. Os-multicast: On-demand situation-aware multicasting in disruption tolerant networks. In *Proc. of the IEEE Vehicular Technology Conference (VTC)*, volume 1, pages 96–100, 2006.
- [25] Y. Xi and M. Chuah. An encounter-based multicast scheme for disruption tolerant networks. *Computer Communications*, 32(16):1742–1756, 2009.
- [26] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. *Technical Report, Dept. of Computer Science, Duke University*, 2000.
- [27] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, July 2003.
- [28] A. Goundan, E. Coe, and C. Raghavendra. Efficient broadcasting in delay tolerant networks. In *Proc. of IEEE GLOBECOM*, 2008.
- [29] G. Karlsson, V. Lenders, and M. May. Delay-tolerant broadcasting. In *Proc. of the ACM SIGCOMM workshop on Challenged networks (CHANTS)*, pages 197–204, 2006.
- [30] F. Peruani, A. Maiti, S. Sadhu, H. Chate, R.R. Choudhury, and N. Ganguly. Modeling broadcasting using omnidirectional and directional antenna in delay tolerant networks as an epidemic dynamics. *IEEE Journal on Selected Areas in Communications (JSAC)*, 28(4):524–531, may 2010.
- [31] T. Ning, Z. Yang, X. Xie, and H. Wu. Incentive-aware data dissemination in delay-tolerant mobile networks. In *Proc. of 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 539–547, 2011.

- [32] W. Gao and G. Cao. User-centric data dissemination in disruption tolerant networks. In *Proc. IEEE INFOCOM*, pages 3119–3127, 2011.
- [33] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proc. of ACM MobiHoc*, pages 241–250, 2008.
- [34] E. M. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *Proc. of the 8th ACM MobiHoc*, pages 32–40, 2007.
- [35] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, 2007.
- [36] X. Zhang, J. Kurose, B. Levine, D. Towsley, and H. Zhang. Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing. In *Proc. of the 13th ACM MobiCom*, pages 195–206, 2007.
- [37] P. Costa, C. Mascolo, M. Musolesi, and G.P. Picco. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 26(5):748–760, 2008.
- [38] M. Motani, V. Srinivasan, and P. S. Nuggehalli. Peoplenet: engineering a wireless virtual social network. In *Proc. of the 11th ACM Mobicom*, pages 243–257, 2005.
- [39] V. Erramilli, A. Chaintreau, M. Crovella, and C. Diot. Diversity of forwarding paths in pocket switched networks. In *Proc. of the 7th ACM SIGCOMM conference on Internet measurement (IMC)*, pages 161–174, 2007.
- [40] I. Rhee, M. Shin, S. Hong, K. Lee, S. Kim, and S. Chong. On the levy-walk nature of human mobility. *IEEE/ACM Transactions on Networking*, 19(3):630–643, 2011.
- [41] M. Abdulla and R. Simon. The impact of the mobility model on delay tolerant networking performance analysis. In *Proc. of IEEE 40th Annual Simulation Symposium (ANSS)*, pages 177–184, 2007.
- [42] A.J. Mashhadi, S. Ben Mokhtar, and L. Capra. Habit: Leveraging human mobility and social network for efficient content dissemination in delay tolerant networks. In *Proc. IEEE WoWMoM*, 2009.
- [43] L. McNamara, C. Mascolo, and L. Capra. Media sharing based on colocation prediction in urban transport. In *Proc. of the 14th ACM Mobicom*, pages 58–69, 2008.
- [44] A. Mei, G. Morabito, P. Santi, and J. Stefa. Social-aware stateless forwarding in pocket switched networks. In *Proc. of IEEE INFOCOM*, pages 251–255, 2011.
- [45] X. Zhang, G. Neglia, J. Kurose, and D. Towsley. Performance modeling of epidemic routing. *Comput. Netw.*, 51(10):2867–2891, July 2007.
- [46] A. Lindgren and K.S. Phanse. Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks. In *Proc. of COMSWARE*, 2006.
- [47] A. Krifa, C. Baraka, and T. Spyropoulos. Optimal buffer management policies for delay tolerant networks. In *Proc. of IEEE SECON*, pages 260–268, 2008.
- [48] M. Seligman, K. Fall, and P. Mundur. Alternative custodians for congestion control in delay tolerant networks. In *Proc. of ACM CHANTS*, pages 229–236, 2006.
- [49] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot. Delegation forwarding. In *Proc. of ACM MobiHoc*, pages 251–260, 2008.
- [50] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAWDAD trace cambridge/haggle/imote/cambridge (v. 2006-01-31), January 2006.

- [51] X. Chen, J. Shen, T. Groves, and J. Wu. Probability delegation forwarding in delay tolerant networks. In *Proc. of IEEE ICCCN*, pages 1–6, 2009.
- [52] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. Endre Tarjan. Time bounds for selection. *J. Comput. Syst. Sci.*, 7(4):448–461, 1973.
- [53] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proc. of ACM WDTN*, pages 252–259, 2005.
- [54] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *Proc. of IEEE PERCOMW*, 2007.
- [55] I. Rhee, M. Shin, S. Hong, K. Lee, and S. Chong. On the levy-walk nature of human mobility. In *Proc. of IEEE Infocom (2008)*.
- [56] NCSU networking research lab: Human mobility models. Downloaded from <http://research.csc.ncsu.edu/netsrv/?q=content/human-mobility-models-download-tlw-slau>, 2009.
- [57] Nielson. More US Consumers Choosing Smartphones as Apple Closes the Gap on Android. <http://blog.nielson.com/nielsenwire/consumer/>, 2012. [Online; accessed 28-February-2012].
- [58] M. McPherson, L. Smith-Lovin, and J. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27:415–444, 2001.
- [59] S. Ross. *Stochastic Processes*. Wiley, 2 edition, January 1995.
- [60] Y. Wang, X. Li, and J. Wu. Multicasting in delay tolerant networks: delegation forwarding. In *Proc. of IEEE GLOBECOM*, 2010.
- [61] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAW-DAD trace cambridge/haggle/imote/infocom2006 (v. 2009-05-29). Downloaded from <http://crawdad.cs.dartmouth.edu/cambridge/haggle/imote/infocom2006>, May 2009.
- [62] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE/ACM Trans. Netw.*, 14:2508–2530, June 2006.
- [63] N. Banerjee, M. D. Corner, D. Towsley, and B. N. Levine. Relays, base stations, and meshes: enhancing mobile networks with infrastructure. In *Proc. of the 14th ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 81–91, 2008.
- [64] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6:606–620, 2007.
- [65] P. Denantes, F. Benezit, P. Thiran, and M. Vetterli. Which distributed averaging algorithm should I choose for my sensor network. In *Proc. of the 27th IEEE Conf. Computer Communications and Networks (INFOCOM)*, 2008.
- [66] U. Feige and V. S. Mirrokni. Maximizing non-monotone submodular functions. In *In Proceedings of 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, page 2007, 2007.
- [67] P. M. Pardalos and S. A. Vavasis. Quadratic programming with one negative eigenvalue is NP-hard. *Journal of Global Optimization*, 1(1):15–22, 1991.
- [68] G. Karlsson, V. Lenders, and M. May. Delay-tolerant broadcasting. In *Proc. of the ACM SIGCOMM workshop on Challenged networks (CHANTS)*, pages 197–204, 2006.
- [69] G. M. Viswanathan, V. Afanasyev, S. V. Buldyrev, E. J. Murphy, P. A. Prince, and H. E. Stanley. Levy flights search patterns of wandering albatrosses. *Nature*, 381:413–415, 1996.

- [70] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453:779–782, 2008.
- [71] Ryan S. and Emin G. Antfarm: efficient content distribution with managed swarms. In *Proc. of USENIX NSDI*, 2009.
- [72] K. Steinhaeuser and N. V. Chawla. Community detection in a large real-world social network. *Social Computing, Behavioral Modeling, and Prediction*, Springer, 2008.
- [73] University of Cambridge computer laboratory: Social network founded mobility models for ad hoc network research. Downloaded from <http://www.cl.cam.ac.uk/research/srg/netos/mobilitymodels/>.
- [74] I. Rhee, M. Shin, S. Hong, K. Lee, S. Kim, and S. Chong. CRAW-DAD data set ncsu/mobilitymodels (v. 2009-07-23). Downloaded from <http://crawdad.cs.dartmouth.edu/ncsu/mobilitymodels>.
- [75] M.S. Granovetter. The Strength of Weak Ties. *The American Journal of Sociology*, 78:1360–1380, 1973.
- [76] S. Aral and D. Walker. Identifying social influence in networks using randomized experiments. *IEEE Intelligent Systems*, 26(5):91–96, September 2011.
- [77] L. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 25:211–230, 2003.
- [78] E. Sun, I. Rosenn, C. Marlow, and T. M. Lento. Gesundheit! modeling contagion through facebook news feed. In *Proc. of the 3rd International Conference on Weblogs and Social Media (ICWSM)*, 2009.
- [79] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. The role of social networks in information diffusion. In *Proc. of the ACM 21st international conference on World Wide Web(WWW)*, pages 519–528, 2012.
- [80] T. W. Valente. *Network models of the diffusion of innovations*. Hampton Press, Cresskill, N.J., 1995.
- [81] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proc. of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.
- [82] D. J. Watts and P. S. Dodds. Influentials, networks, and public opinion formation. *Journal of Consumer Research*, 34:441–458, 2007.
- [83] D. Centola and M. W. Macy. Complex contagions and the weakness of long ties. *American Journal of Sociology*, 113:702–734, 2007.
- [84] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.
- [85] S. Aral and D. Walker. Identifying influential and susceptible members of social networks. *Science*, 337:337–341, 2012.
- [86] A. Pentland N. Eagle and D. Lazer. Inferring social network structure using mobile phone data. In *Proc. of the National Academy of Sciences*, volume 106(36), pages 15,274–15,278, 2009.
- [87] Y. Kalish and G. Robins. Psychological predispositions and network structure: The relationship between individual predispositions, structural holes and network closure. *Social Networks*, 28(1):56 – 84, 2006.
- [88] J. Staiano, B. Lepri, N. Aharony, F. Pianesi, N. Sebe, and A. Pentland. Friends don’t lie: inferring personality traits from social network structure. In *Proc. of the 2012 ACM Conference on Ubiquitous Computing (UbiComp)*, pages 321–330, 2012.

- [89] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *Proc. of IEEE INFOCOM*, 2006.
- [90] G. S. Thakur, A. Helmy, and W. Hsu. Similarity analysis and modeling in mobile societies: the missing link. In *Proc. of ACM CHANTS*, 2010.
- [91] A. Mei, G. Morabito, P. Santi, and J. Stefa. Social-aware stateless forwarding in pocket switched networks. In *Proc. of IEEE Infocom*, 2011.
- [92] C. Shannon, N. Petigara, and S. Seshasai. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.
- [93] J. Wu. *Distributed System Design*. CRC Press, 1998.
- [94] Y. Saad and M.H. Schultz. Topological properties of hypercubes. *IEEE Transactions on Computers*, 37(7):867–872, 1988.
- [95] A. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19:153–158, 1997.
- [96] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.
- [97] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.
- [98] T. Hossmann, T. Spyropoulos, and F. Legendre. Putting contacts into context: Mobility modeling beyond inter-contact times. In *Proc. of ACM MobiHoc*, 2011.
- [99] H. Cai and D. Eun. Crossing over the bounded domain: from exponential to power-law intermeeting time in mobile ad hoc networks. *IEEE/ACM Trans. Netw.*, 17(5):1578–1591, October 2009.
- [100] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Efficient routing in intermittently connected mobile networks: the multiple-copy case. *IEEE/ACM Transactions on Networking*, 16(1):77–90, 2008.
- [101] M. Musolesi and C. Mascolo. A community based mobility model for ad hoc network research. In *Proc. of ACM REALMAN*, pages 31–38, 2006.

# PUBLICATION LIST

## *Journal Papers*

1. **Yunsheng Wang**, Wei-Shih Yang, and Jie Wu. “Analysis of a Hypercube-based Social Feature Multi-Path Routing in Delay Tolerant Networks”, *accepted to appear in Transactions on Parallel and Distributed System (TPDS)*, 2012.
2. Jie Wu and **Yunsheng Wang**. “Hypercube-based Multi-path Social Feature Routing in Delay Tolerant Networks”, *accepted to appear in Transactions on Computers (TC)*, 2012.
3. **Yunsheng Wang**, and Jie Wu. “Ticket-based Multiple Packet Broadcasting in Delay Tolerant Networks”, *accepted to appear in Journal of Ad Hoc & Sensor Wireless Networks*, 2012.
4. **Yunsheng Wang**, and Jie Wu. “A Dynamic Multicast Tree based Routing Scheme without Replication in Delay Tolerant Networks”, *Journal of Parallel and Distributed Computing (JPDC)*, Volume 72, Issue 3, Pages 424-436, 2012.
5. **Yunsheng Wang**, Xiaoguang Li, and Jie Wu. “Delegation Forwarding in Delay Tolerant Networks Multicasting”, *Journal of Communications*, Vol 6, No 5 (2011), 384-392, Aug. 2011.

## *Conference Papers*

6. **Yunsheng Wang** and Jie Wu. “Information Influence in Mobile Opportunistic Social Networks”, *In the 14th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Madrid, Spain, June 2013.
7. **Yunsheng Wang**, Jie Wu, Zhen Jiang, and Feng Li. “A Joint Replication-Migration-based Routing in Delay Tolerant Networks”, *In IEEE International Conference on Communications (ICC)*, Ottawa, Canada, June, 2012.
8. Jie Wu and **Yunsheng Wang**. “Social Feature-based Multi-path Routing in Delay Tolerant Networks”. *In The 31st Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2012.
9. **Yunsheng Wang**, Yuhong Guo, and Jie Wu. “Making Many People Happy: Greedy Solutions for Content Distribution”, *In The 40th Annual IEEE International Conference on Parallel Processing (ICPP)*, Taipei, Taiwan, September, 2011.
10. **Yunsheng Wang**, Xiaoguang Li, and Jie Wu. “Multicasting in Delay Tolerant Networks: Delegation Forwarding”, *In IEEE Global Communications Conference (GLOBECOM)*, Miami, December, 2010.
11. Jie Wu and **Yunsheng Wang**. “A Non-Replication Multicasting Scheme in Delay Tolerant Networks”. *In The 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, San Francisco, November, 2010.
12. Chi-Kin Chau, Muhammad Husni Wahab, Fei Qin, **Yunsheng Wang**, and Yang Yang. “Battery Recovery Aware Sensor Networks”, *In IEEE 7th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, Seoul, June, 2009.