OPPORTUNISTIC ROUTING ALGORITHMS IN DELAY TOLERANT NETWORKS

By

Eyuphan Bulut

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: COMPUTER SCIENCE

Approved by the
Examining Committee:

_____
Prof. Boleslaw K. Szymanski, Thesis Adviser

_____
Prof. Christopher Carothers, Member

_____
Assoc. Prof. AlHussein Abouzeid , Member

_____
Assoc. Prof. Biplab Sikdar , Member

Rensselaer Polytechnic Institute
Troy, New York

February, 2011
(For Graduation May 2011)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

This thesis is the end of my journey for obtaining a doctorate degree in Computer Science. I can honestly state that it would not have been possible without the support of many people.

First of all, I would like to indicate that it is my great fortune to have pursued my Ph.D. studies under the guidance of such a great advisor, Professor Boleslaw Szymanski. I would like to thank to him for his continuous support, encouragement and invaluable concern throughout this research. Especially, his pleasant and friendly personality and precise guidance made this graduate study more enjoyable. At every step of my thesis, he guided me with his profound knowledge, insight and wisdom. I would like to express my deep gratitude to him for being such a great advisor.

I also would like to thank to Professor Christophers Carothers, Professor Al-Hussein Abouzeid and Professor Biplab Sikdar for being members of my committee and giving their insightful comments and suggestions through forming my thesis work.

I also would like to thank to my colleagues, Sahin Cem Geyik and Zijian Wang, for being such wonderful people to collaborate with. I appreciate their efforts in helping me for solving several problems I faced through this thesis work. I should also thank many other friends including Hilmi Yildirim, Jerry Xie and Lei Chen for their friendship and sharing life with me through my graduate studies.

Finally, my deep gratitude goes to my parents and my wife. I am very grateful to them for standing by me in everything I have done and giving me whatever they can. They have always provided me continuous support, encouragement and their love. I believe that nothing would be possible without the presence of them and the peaceful family environment they provided me during my life. I dedicate this thesis to them.

ABSTRACT

Delay Tolerant Networks (DTNs), also called as intermittently connected mobile networks, are wireless networks in which a fully connected path from source to destination is unlikely to exist. Therefore, in these networks, message delivery relies on opportunistic routing where nodes use store-carry-and-forward paradigm to route the messages. However, effective forwarding based on a limited knowledge of contact behavior of nodes is challenging.

In this thesis, we discuss several aspects of routing problem in DTNs and present four novel algorithms for different DTN environments: (i) multi-period multi-copy based Spray and Wait routing algorithm where the copies are distributed to the nodes in different periods, (i) multi-period erasure coding based routing algorithm where the optimal erasure coding parameters for different periods are selected to minimize the cost, (iii) efficient single copy based routing algorithm where the correlation between the mobility of nodes are utilized, and (iv) social structure-aware routing algorithm where message exchanges between nodes are performed considering the social relations of nodes. In all of these algorithms, our common objective is to increase the message delivery ratio and decrease the average delivery delay while minimizing the routing cost (number of copies used per message or number of forwardings of a single message between the nodes) under given circumstances. We also present simulation results (based on both real and synthetic DTN traces) regarding the performance comparison of the proposed algorithms with the state-of-the-art routing algorithms in DTNs.

CHAPTER 1
INTRODUCTION

Delay Tolerant Networks (DTN), also referred to as *Intermittently Connected Mobile Networks*, are wireless networks in which at any given time instance, the probability that there is an end-to-end path from a source to destination is low. Since most of the nodes in a DTN are mobile, the connectivity of the network is maintained by nodes only when they come into the transmission ranges of each other. If a node has a message copy but it is not connected to another node, it stores the message until an appropriate communication opportunity arises.

There are many examples of such networks in real life. For example, in north part of the Sweden [1], the communication between villages and the summer camps of the Saami population is provided when the nodes get connected. The same situation is also seen in rural villages of India and some other poor regions [2]. Other fields where this kind of communication scenarios may occur also include satellite communication [3], wildlife tracking [4], military networks [5] and vehicular ad hoc networks [6]. Moreover, such environments can exist even when a stable infrastructure is destroyed by natural disaster or other effects. A more interesting example of DTNs is the applications where sensors are attached to seals [7] and whales [8] to collect large number of sensor readings from the oceans. In these applications, the data collected by sensors on seals and whales is transferred to a sink node using the transitive connectivity between the sensor nodes.

1.1 Routing Problem in DTNs

Although the connectivity of nodes is not constantly maintained, it is still desirable to allow communication between nodes. Therefore, it is necessary to provide a routing protocol which tries to route packets throughout the times the link is available among the nodes. But this can not be done by standard routing algorithms which assume that the network is connected most of the time.

In a standard network, since the nodes are connected most of the time, the

routing protocol forwards the packets in a simple way. The cost of links between nodes are mostly known or easily estimated so that the routing protocol computes the best path to the destination in terms of cost and tries to send the packets over this path. Furthermore, the packet is only sent to a single node because the reliability of paths is assumed relatively high and mostly the packets are successfully delivered. However, in DTN like networks, routing becomes challenging because the nodes are mobile and connectivity is rarely maintained.

The transient network connectivity needs to be of primary concern in the design of routing algorithms for DTNs. Therefore, routing of the packets is based on *store-carry-and-forward* paradigm. That is, when a node receives a message but if there is no path to the destination or even a connection to any other node, the message should be buffered in this current node and the upcoming opportunities to meet other nodes should be waited. Moreover, even a node meets with another node, it should carefully decide on whether to forward its message to that node. It is obvious that to forward a message to multiple nodes increases the delivery probability of a message. However, this may not be the right choice because it can cause a huge messaging overhead in the network which then causes redundant energy and resource consumption. On the other hand, sending a copy of the message to a few number of nodes uses the network resources efficiently but the message delivery probability becomes lower and the delivery delay gets longer. Consequently, it is clearly seen that there is a tradeoff between the message delivery ratio and the energy consumption and delivery delay in the network. Hence, while designing a routing protocol for delay tolerant networks, the important considerations must be (i) the number of copies that are distributed to the network for each message, and (ii) the selection of nodes to which the message is replicated or forwarded.

Consider the sample delay tolerant network illustrated in Figure 1.1. It presents four different snapshots of the network showing connectivity between nodes at four different times. Assume node A has a message destined to node G. Looking at the snapshots, we can easily observe that delivery of the message could be achieved by node B at T_4 if node A forwards the message to node B at time T_1. However, the key point here is how node A will know that node B will meet the destination node

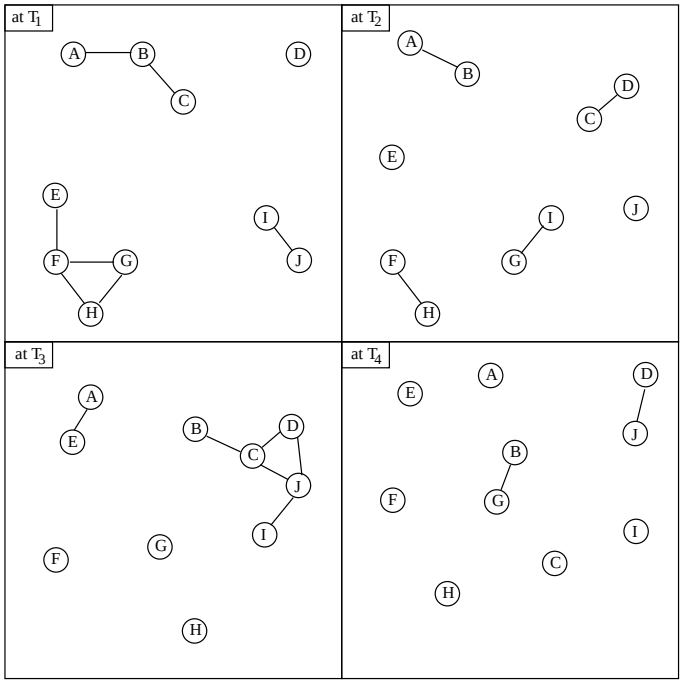


Figure 1.1: Snapshots of a delay tolerant network at four different times.

before it meets the destination. What makes routing challenging in a DTN is to be able to make better decisions at contact times of nodes using only local information available at nodes.

In this thesis, we study the routing problem in delay tolerant networks and provide different solutions. In each, we use different routing techniques and work on different DTN scenarios, however our common main objective is to minimize the routing cost[1] while achieving high delivery rate by the delivery deadline.

1.2 Our Contributions

In this thesis, we propose new opportunistic routing algorithms for *Delay Tolerant Networks*. We analyze the DTN routing problem for different DTN environments (i.e. homogeneous networks, heterogeneous networks, social networks) and by using different approaches (each chapter of the thesis focus on one approach) we provide different routing algorithms to decrease the routing cost of the messages in

[1]We define routing cost as the number of copies used per message in multi-copy based algorithms and as the number of forwardings of a message in single-copy based routing algorithms.

delay tolerant networks. The basic features and novelties of the proposed routing algorithms are the following:

- We propose multi-period spray and wait routing algorithm where we distribute limited and predefined number of message copies at different spraying periods and wait for the delivery of any of them between spraying durations. As the spraying times (periods) start and the message delivery does not happen yet, we spray additional copies of the message to increase the probability of its delivery. As a result of this efficient copying strategy in multiple periods, we get the advantage of early delivery and achieve the same or a higher delivery rate by the deadline while using fewer average number of copies per message than the single period spray and wait algorithm where all copies of the message are distributed at the beginning of the routing.

- We use erasure coding technique to increase the reliability and to further decrease the cost of routing. For a given desired delivery rate and deadline for delivery, we find the optimum parameters to obtain the smallest cost both in single period and two period erasure coding based routing. We also analyze the effects of message distribution algorithms on the cost of routing both in replication based (i.e. spray and wait) and erasure coding based algorithms.

- We analyze real DTN traces and detect the correlations between the movements of different nodes using a new metric called conditional intermeeting time. We then use the correlations between the meetings of a node with other nodes for making the existing single-copy based routing algorithms more cost efficient.

- We analyze the social relations in mobile social networks (which are special kind of delay tolerant networks where the nodes are human-carried devices) and describe how we can improve the performance of DTN routing by extracting and using social network related concepts. First, we analyze the impact of the grouping behavior of nodes in a social network and propose a variant of spray and wait routing protocol for community based social networks where the message copies are distributed considering the community structure of the

network. Second, we propose a new social network metric (inspired from the friendship relations between people) to extract the message contact opportunities between nodes more accurately. We analyze both direct and indirect relations between nodes and form temporal friendship communities at different time frames for nodes to be used in routing decisions.

1.3 Thesis Structure

In Chapter 2, we start with summarizing the related work in literature. We give different classifications of previously proposed routing algorithms for DTNs and talk about the main properties of some popular algorithms. Chapter 3 presents the details of the proposed multi-period spray and wait routing. It includes both analytical and simulation results in which we demonstrate that the proposed algorithm overcomes original (single-period) spray and wait routing algorithm. In Chapter 4, we present a cost effective erasure coding based routing algorithm. There, we present both the single and multi-period erasure coding based algorithms and their results. Additionally, we also demonstrate a comprehensive comparison of multi-copy and erasure coding based algorithms. We show the effect of message distribution algorithms on the cost of routing algorithms. Chapter 5 looks the routing problem from a different perspective. We focus on the correlation between the movements of nodes in a DTN and utilize the correlation between the meetings of a node with other nodes in making the existing single copy based routing algorithms more efficient. In Chapter 6, we study the routing problem in a special kind of DTNs, mobile social networks, and present two different ways of using social network properties of these networks in designing better routing algorithms. We first discuss the impact of community structure on spray and wait routing algorithm. Then, we propose a friendship based routing algorithm in which we use a new social network metric to detect the direct and indirect relations between nodes more accurately and use these extracted social relations between nodes to make better routing decisions. Finally, the thesis ends with conclusions and discussions in Chapter 7.

CHAPTER 2
RELATED WORK

It has been almost a decade since the initiating talk [9] of Kevin Fall about delay tolerant networks. The primary focus of researchers studying on DTNs has been routing problem. Many studies have been performed on how to handle the sporadic connectivity between the nodes of a DTN and provide a successful and efficient delivery of messages to the destination. Different classifications of these algorithms can be made. In Figure 2.1, we show two different classifications of routing algorithms proposed for delay tolerant networks.

2.1 Classification based on knowledge available at nodes

The first classification of routing algorithms is decided according to the broadness of the knowledge of the network available at nodes. In some studies [82], it is assumed that each node in the network has exact knowledge of (past and future) node trajectories, or node meeting times and durations. Therefore, the messages are routed over predetermined paths (sequence of nodes) deterministically. But these algorithms which assume the existence of oracles giving future information are unrealistic because the intermittent connectivity between the mobile nodes in delay tolerant networks does not allow nodes to have such information. On the other hand, there are also significant number of studies (such as Epidemic [22], Prophet [25],

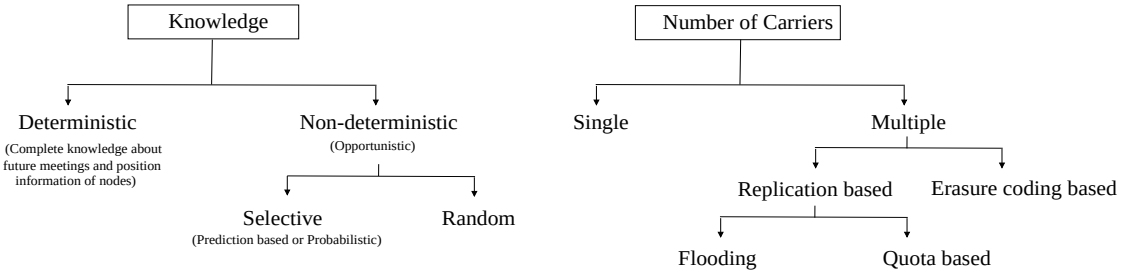


Figure 2.1: Two different classifications of routing algorithms proposed for delay tolerant networks.

Spray and Wait [37]) assuming zero knowledge about the aforementioned features of the nodes. These algorithms either forward the messages randomly or use the meeting history of nodes (which can be obtained locally through encounters with other nodes) and forward the messages over different paths in a nondeterministic manner.

2.2 Classification based on number of carriers of the message

We can also classify the routing algorithms in terms of the number of carriers of the message during routing. In some algorithms (such as Prophet [25], SCAR [91], MaxProp [36] and [95]) there exists only one node carrying the message at all times. In these algorithms, the messages are forwarded to nodes which have higher chance to meet the destination. One other common method used in routing algorithms for delay tolerant networks is using multiple carriers of the message. In the first type of these algorithms, a number of copies of the same message is generated and distributed to multiple nodes so that the delivery probability of the message is increased. Among these algorithms, while some of them distribute limited number of copies ([35], [37]) to other nodes in the network, some others [22] provide flooding like dissemination of the message copies. Different than replication based algorithms, some algorithms [26] use erasure coding technique for efficient routing of messages. They first process and convert a message of k data blocks into a large set of Φ blocks such that the original message can be constructed from a subset of Φ blocks. Then each of these encoded blocks are distributed to the other nodes in the network and the delivery of sufficient number of blocks is expected to reconstruct the original message.

After looking the routing algorithms for delay tolerant networks from a general perspective, next, we will give the details of some of the popular algorithms in this area.

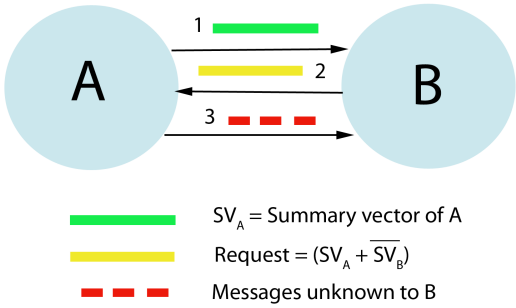


Figure 2.2: When two hosts (A and B) come into transmission range of one another, they first exchange summary vectors, then the necessary packets for transmission is decided and as final step these packets are transmitted to each other [22].

2.3 Popular DTN Routing Algorithms

The pioneering algorithm in the field of multi-copy based (multiple nodes carrying the replications of the same message) routing for delay tolerant networks is Epidemic Routing [22] which is published by Vahdat and Becker. This field has attracted considerable attention with the introduction of this work and other routing algorithms are developed as a counter algorithm to Epidemic Routing. Epidemic Routing protocol basically relies on the epidemic like algorithms [11, 12]. In these algorithms, when the nodes in the network get contact with each other, a pair-wise information exchange between nodes happens so that the message is delivered to the destination eventually. However, if there is no path currently available to the destination, the messages are buffered. In each node meeting, first a summary vector which holds the index of all messages in a node is transferred to the other node. Then, having the other nodes' summary vector, each node learns the message IDs which are not available in its own buffer and requests the transfer of these messages from the other node. To be able to do this, each message is assigned a unique message ID.

Figure 2.2 illustrates this message exchange procedure in Epidemic Routing protocol which starts whenever two hosts come into the range of one another. Here, we will go through the details of the procedure from host A's point of view. The same process is also applied for host B. As a first step, A transmits its summary vector,

SV_A (a compact representation of all the messages being buffered at A) to B. Next, B performs a logical AND operation between the negation of its summary vector, $\neg SV_B$ (represents the messages that it needs) and SV_A. By this way, host B determines the set difference between the messages buffered at A and the messages buffered at B. Then, it transmits a vector requesting these messages from A. As a final step, host A transmits these requested messages to host B. This message exchange procedure is always applied when two hosts come into contact with each other. As a result, if the buffer space at hosts and the time at contact times are sufficient, the messages are eventually delivered to the destination hosts through this pair-wise message exchanges. Here, one can easily notice that Epidemic Routing provides the fastest spread of copies in the network which of course yields the optimum delivery time.

One of the first studies that address the weakness of epidemic routing is Probabilistic ROuting Protocol using History of Encounters and Transitivity (Prophet) [25]. The idea presented in that study basically depends on the following observation. Lindgren et al. believe that the movement of nodes in a typical mobile ad hoc network is not random as it is mostly assumed to be. Moreover, they claim that the nodes move in a predictable fashion based on repeating behavioral patterns such that if a node has visited a location several times before, it is likely that it will visit that location again. Depending on this observation, the authors propose a probabilistic routing model in which the delivery rate of messages is aimed to be improved while keeping buffer usage and communication overhead at a low level.

The Prophet algorithm operates in a similar way as Epidemic Routing does. When two nodes meet, they exchange summary vectors as it is in Epidemic routing, but in this case an additional piece of data called delivery predictability information is also exchanged between the nodes. Here the delivery predictability information is a probabilistic metric established by the authors of the paper and defined by $P_{(a,b)} \in [0, 1]$, at every node A for each known destination B. It indicates how likely it is that the node A will be able to deliver a message to that destination B.

When this exchanging process is done between nodes, then according to these summary vectors, the messages that will be requested from the other node is de-

cided considering the forwarding strategy. The basic difference of Prophet than Epidemic Routing is its forwarding strategy. When two nodes meet, Prophet allows the transfer of a message to the other node only if the delivery predictability of the destination of the message is higher at the other node.

In Prophet, a three step calculation of the delivery predictabilities of nodes is presented.

- Since the idea of the paper inspired by non-random mobility of nodes which also provides different meeting popularities to different nodes, the authors would like to favor the nodes who are most encountered among the others for the delivery of messages. Therefore, to reflect this property in the delivery probability metric of a node, $P_{(a,b)}$ is updated whenever a node is encountered, so that nodes that are often encountered have a high delivery predictability. This calculation is shown below, where $P_{init} \in [0, 1]$ is an initialization constant.

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

- On the other hand, if a pair of nodes does not encounter each other in a while, due to the same reasoning, they are less likely to be good forwarders of messages to each other. Therefore, the authors add an aging mechanisms to the delivery predictability values of the nodes. The equation below shows the simple aging equation of a node in which $\gamma \in [0, 1)$ represents the aging constant, and k is the number of time units that have elapsed since the last time the metric was aged. The time unit used in the formulation can differ, and should be defined based on the application and the expected delays in the targeted network.

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

- It is obvious that if node A frequently encounters node B, and node B frequently encounters node C, then node C probably is a good node to forward

messages destined for node A. To make use of this transitive property, authors include this in the updating process of the delivery predictability. Here $\beta \in [0, 1]$ is the scaling constant that decides how large impact the transitivity should have on the delivery predictability.

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

Once the $P_{(a,b)}$ values of each node is continuously updated according to the above equations, then the rest of the algorithm works as follows. When two nodes meet, a message is transferred to the other node if the $P_{(a,b)}$ value of the destination of the message is higher at the other node.

In [35], Harras et al. study the impact of controlled message flooding schemes over sparse mobile networks on message delay and network resource consumptions. Like Prophet, they use probabilistic modeling for message forwarding and add some additional schemes on top of the probabilistic model. These schemes include time-to-live (TTL), kill time value and passive cure.

In [31], Li et al. propose an efficient store-and-forward based scheme called Adaptive Multi-Copy Routing (AMR), for packet delivery in Intermittently Connected Mobile Ad Hoc Networks (ICMANs). The novelty of their idea is, instead of using a source-defined replication factor, in this scheme each individual intermediate relay node decides whether to replicate a message or not independently. In other words, the number of copies of the message that will be distributed to the network is not decided at the beginning by the source node, instead it is decided by each individual node according to the current network conditions and the end-to-end delay target, the upper limit of desired delay for the delivery of the message. By this way, the approach becomes replication-factor-free and more cost-efficient than the copying schemes in which the replication factor is defined at the beginning by the source node.

Jones et al. present a practical routing protocol for delay tolerant networks in [83]. The idea used in the proposed algorithm is an adapted version of shortest path routing protocol in traditional networks such that while deciding the path some new

metrics are used. In this approach, one single copy of the message is generated and forwarded towards the destination using the predicted topology information which is obtained with the help of many algorithm maintenance messages distributed to the network. The authors use the same DTN model presented by Jain et al. in [82]. This work can also be considered as the extension of [82] by different authors. The network is assumed to be an undirected graph where the nodes are connected by bidirectional links called contacts. In other words, the edges of the graph represent the opportunity for nodes to exchange their data. The cost of the nodes is estimated by a new metric called *minimum estimated expected delay* (MEED). This is actually similar to the metric *minimum estimated delay* (MED) used in [82]. In MED, the expected waiting time is computed using the information of future contact schedule. However, MEED uses the observed contact history. In other words, the connection and disconnection times of each contact is recorded over a sliding history window. Here, the size of the window is a tuning parameter of the algorithm and can be changed independently at each node. If the window size is large, this makes the metric durable against the perturbations caused by random changes. But, the metric also shows slow reaction to permanent changes. On the other hand, a small window size makes the metric sensitive to random fluctuations but can help in easily adapting the permanent changes.

One of the significant works in this field is MaxProp [36]. Burgess et al. from University of Massachusetts, Amherst propose an effective routing protocol for delay tolerant networks based on prioritizing both the schedule of packets transmitted to other nodes and the schedule of packets that will be deleted from the buffer. In MaxProp, not only a new routing protocol for delay tolerant networks is proposed but also the construction of a real life DTN test-bed is discussed which is then used for the evaluation of proposed scheme. MaxProp has also a convenient design for scenarios in which either the transfer duration or buffer space available in the nodes is limited. The order of packets in which the packets will be deleted in case of full buffers and the order in which the packets are transmitted to other nodes in case of node meetings are addressed by MaxProp by ranking the packets according to some criteria.

MaxProp lets each node keep the track of probability of meeting with other nodes. Denoting the probability that node i will be connected to node j as the next step with f_j^i, all f_j^i's are assigned $1/(|s|-1)$ at the beginning, where s denotes the number of nodes in the network. In each meeting of node i with node j, the value of f_j^i is incremented by 1. But to preserve the total probability value of 1, all f_j^i values are then normalized. By this way, the nodes who see each other less frequently, have lower values of f_j^i. Moreover, MaxProp also enables nodes to exchange their f_j^i values in contact times.

Once a node has the values of delivery likelihood for other nodes, then it calculates the cost of each possible path, c(i, i + 1, . . . , d), to destination up to n (defined by protocol) hops long using the following formula:

$$c(i, i+1, ..., d) = \sum_{x=i}^{d-1} [1 - (f_{x+1}^x)]$$

After the cost of all possible paths are calculated, the path with the lowest cost is selected to be the cost of reaching destination.

The study presented in [47] by Spyropoulos et al. is another significant work in the area of routing protocols for delay tolerant networks. In that study, authors work on proposing algorithm which can perform fewer transmissions than flooding based routing schemes and deliver a message faster than existing single and multi-copy schemes while achieving optimal delays and higher delivery ratios. For this purpose, there are two different routing algorithms proposed:

1. *Spray and Wait:* In this algorithm, authors tries to combine the efficiency of flooding based algorithms and the simplicity of direct transmission. There are two phases:

    - *Spray phase:* In this phase, a limited number of copies (L) of a message is spread over the network by the source and some other nodes which later receives a copy of the message. Here, there are two important questions: how many copies of the message will be spread and how will these copies be spread to other nodes in the network?

- *Wait phase:* After the spreading of all copies of the message is done and the destination is not encountered by a node with a copy of the message in the spraying phase, then each of these nodes carrying a message copy tries to deliver its own copy to destination via direct transmission independently.

2. *Spray and Focus:* This algorithm is designed to eliminate some deficiencies of Spray and Wait algorithm in some network environments. In Spray and Wait algorithm, once all the copies of the message is spread to some nodes and wait phase is started, but if the mobility of each node is restricted to a small local area, then it may not be possible to deliver one of the copies to the destination. Therefore, authors also propose a different version of the Spray and Wait algorithm. There are also two phases in this algorithm:

   - *Spray phase:* This phase is actually same with the spray phase of the first algorithm. For every message originating at the source node, L message copies are spread to all L different nodes.
   - *Focus phase:* Once the spraying phase is done, then nodes start to roam around to find the destination. But different than the wait phase of the first algorithm, in this phase, each copy in a single node is tried to be routed to a closed node via a single-copy utility based scheme [46]. That is, if $U_X(Y)$ denotes the utility of node X for destination Y, then a node having a copy of the message destined to node D, forwards its copy to a new node B in its range, if and only if $U_B(D) > U_A(D) + U_{th}$. Here, U_{th} denotes the utility threshold parameter of the algorithm.

On the other hand, to make the routing in DTNs more reliable, some researchers proposed routing algorithms based on erasure coding technique. One of the first studies utilizing the erasure coding approach is [26]. In that study, Wang et al. present the advantages (robustness to failures etc.) of erasure coding based routing over the replication based routing. In [27], the split of erasure coded blocks over multiple delivery paths (contact nodes) to optimize the probability of successful message delivery is studied. A similar approach focusing on the distribution of

encoded blocks among the nodes is presented in [28] by Liao et al. They propose an estimation based erasure coding routing depending on a realistic assumption that the nodes in the network are not identical. As an extension of this work, in [29], authors also utilizes the information on a node's available resources (buffer space, remaining energy level etc.) in the evaluation of the node's capability to successfully deliver the message. In [30], a hybrid routing algorithm combining the strengths of replication based and erasure coding based routing is proposed. In addition to encoding of each message into large amount of small blocks, these blocks are also replicated to increase the delivery rate.

In [32], Liu et al. propose to use a new long term metric called expected minimum delay (EMD). This is the expected time that an optimal forwarding scheme takes to deliver a message from a source to a destination at a specific time in a network with cyclic and uncertain connectivity. However, in that study each contact time of a node is assumed to be formed from the contact history with an assumption that it will not change later. Consequently, the use of a state diagram which includes a different state for each contact of each node is proposed. However, this creates a huge state diagram when the node meetings have a huge common round duration.

Other than the above studies, a few of the previous works focused on the routing problem in (mobile) social networks, which are special kind of delay tolerant networks in which the nodes are human carried devices. Since the contacts (i.e. message exchange opportunities) between these mobile devices depend on the social relations between the people carrying these devices, researchers tried to utilize social network concepts (community detection, similarity, betweenness etc.) to understand the contact patterns between nodes to develop better routing algorithms. In a social network, there might be communities formed by nodes which meet each other more frequently than the nodes outside the their communities. Clearly such community formation affects the routing decisions that nodes need to make when they meet other nodes. Considering a possible partitioning of nodes into communities in social networks, there are some algorithms proposed to make the routing of messages more efficient in such networks. In [56], Daly et al. use both the betweenness and the similarity metric to increase the performance of routing. When two nodes contact, they

calculate the utility function comprised of these two metrics for each destination, then the node having higher utility value for the message's destination is given the message. In BubbleRap [48], each node is assumed to have two rankings: global and local. While the former denotes the popularity (i.e. connectivity) of the node in the entire society, the latter denotes its popularity within its own community. Messages are forwarded to nodes having higher global ranking until a node in the destination's community is found. Then, the messages are forwarded to nodes having higher local ranking. By this way, first the probability of finding the destination's community is increased. Then, after the message reaches the destination's community, the probability of meeting the destination is increased, so that the shortest delivery delay is attempted. In [49], a publish/subscribe communication in which many-to-many communication paradigm is also addressed as an extension to end-to-end style which is usually assumed in DTNs. Then, using the centrality values of nodes, an effective multi-point communication and efficient routing is enabled. In LocalCom [76], a community-based epidemic forwarding scheme is introduced. First, the community structure of the network is detected using local information of nodes. Then, the message is forwarded to each community through gateways. Additionally, in some other studies, several interesting properties of social networks are considered. In [77], irregular deviations from the habitual activities of nodes are addressed and it is shown that the worst-case performance of routing can be improved by scattering multiple copies of a message in the network such that even deviant (less frequently encountered) nodes will be close to at least one of these copies. In [78], the effect of socially selfish behavior of nodes on routing is studied.

In the design of DTN routing protocols, there are also some important issues that need to be considered to make a fair comparison between the proposed algorithms. In Table 2.1, we show the comparison of some of the discussed algorithms in terms of the assumptions made, number of copies used and the criteria used while forwarding the messages. The table also shows the comparison of simulation model used in each of these studies. When we look at this table, we observe that each of these algorithms has some advantages and disadvantages over the others. Therefore, the right algorithm for the routing of messages in a delay tolerant network should

Table 2.1: Comparison of Algorithms

	Algorithm				**Assumptions**		**SimulationModel**	
	Number of copies	**Decision based on**	**Drawbacks**	**Delivery Ack**	**Buffer Size**	**Bandwidth Capacity**	**Simulator**	**Mobility Model**
Epidemic Routing [22]	Unlimited	Flooding	High resource (bandwidth, buffer) usage	Not mentioned	Limited	Not mentioned	ns-2	RWP
Prophet [25]	Single	Probability obtained from previous meetings	High message overhead	Not mentioned	Limited	Not mentioned	Own	RWP
Controlled Flooding [35]	Unlimited	Node willingness (probability)	Relatively high message overhead	Yes	Not mentioned	Not mentioned	GloMoSim	RWP
Adaptive Routing [31]	Limited	Estimation based on hop count from source and contact duration	Continues spreading after message delivery	Not mentioned	Sufficient	Sufficient	ns-2	RWP
Practical Routing [83]	Single	Best path found according to the previous meetings of nodes	1) Loops may occur 2) High transmission overhead due to routing table maintenance	Not mentioned	Limited	Limited	Own	Real trace
MaxProp [36]	Single	Previous node meetings and updated route estimation	High processing cost in large scale networks	Yes	Unlimited (own) Limited (other)	Limited	Own	Real data
Spray and Wait [47]	Limited (L)	Randomness	Random decision making	Not mentioned	Sufficient	Sufficient	Own	RWP, RD, RW
SimBet [68]	Single	Similarity & Betweenness	Binary and stable social relation representation	Not mentioned	Sufficient	Sufficient	Trace driven	Real trace

be chosen considering the availability of resources (i.e. bandwidth, buffer) in the network and the main goal (i.e. high delivery rate, low delivery delay) that is desired to be achieved.

In this chapter, we tried to summarize the properties of the state-of-art routing algorithms for DTNs. Interested readers can also look at some surveys such as [18] and [60] for more extensive research on the related work in literature.

CHAPTER 3
MULTI-PERIOD SPRAY AND WAIT ROUTING

This chapter describes our multi-period spray and wait based routing algorithm [50, 51, 52]. Below, we first list the assumptions of our model and then describe our routing algorithm in detail. Moreover, we also present the analysis of the proposed algorithm with its variants.

We assume that there are M nodes moving on a $\sqrt{N}$ x $\sqrt{N}$ 2D torus according to the random direction mobility model. Each node has a transmission range R and all nodes are identical. The meeting times of nodes are assumed to be independent and identically distributed (IID) exponential random variables. Furthermore, we also assume that the buffer space in a node is unlimited (this assumption is not crucial since the presented algorithm uses the predefined number of copies with the maximum number comparable to the single period spraying algorithm). We also assume that the communication between nodes is perfectly separable, that is, any communicating pair of nodes do not interfere with any other simultaneous communication. To be consistent with previous research, by L we denote the number of copies that each message distributes to the network.

In Spray and Wait algorithm [47], the delivery of a message can happen both in spray and wait phases. The probability of message delivery at or before time t when there are L copies of the message in the network is $p_d = 1 - e^{-\alpha L t}$, where $\alpha = 1/EM$ is the inverse of the expected intermeeting time between two consecutive encounters of any pair of nodes. During waiting phase, since L is constant, p_d grows with the same L value. However, since the number of copies increases during the spraying phase, p_d function changes each time a new copy is distributed to other nodes.

To simplify the analysis of message delivery probability, we assume that $M >> L$ which is often true in DTNs and which we enforce by limiting permissible values of L. Moreover, for DTNs to be of practical use, the delivery probability p_d must be close to 1, so we assume also that $p_d \geq= 0.9$. We will show below that

from these two assumptions it follows that the formula $p_d = 1 - e^{-\alpha Lt}$ is a good approximation of the delivery probability at time $t = t_d$.

At the i^{th} encounter with another node, the spraying node delivers the message to the destination with probability $1/(M - i)$, so the total probability that the message is delivered during spraying is between L/M and $L/(M - L)$, and since $M >> L$, L/M is a good approximation of this probability. Binary spraying uses $log(L)$ steps, each with average time about EM/M (in k^{th} step, 2^{k-1} nodes sprays a message copy to 2^{k-1} other nodes), so the total spraying delay is $log(L)EM/M$. The approximated formula achieves the same delivery probability at the earlier time EM/M, and from that time on, it matches the behavior of the algorithm perfectly. Hence, the average difference between times at which algorithm and formula achieve the same delivery probability is $d = (log(L) - 1)EM/M$. Thus, the relative error $e(p_d)$ of using the approximate formula for p_d is:

$$\begin{aligned} e(p_d) &= \frac{(1 - e^{-\alpha L(t_d - d)}) - (1 - e^{-\alpha L t_d})}{1 - e^{-\alpha L(t_d - d)}} \\ &= 1 - \frac{(1 + e^{-\alpha L(t_d - d)})(1 - e^{-\alpha L t_d})}{(1 - e^{-2\alpha L(t_d - d)})} \\ &\approx -(1 - p_d)(\log(L) - 1)\frac{L}{M} \end{aligned}$$

Since $p_d \geq 0.9$, for $L <$2048 (so much beyond the range of useful values of L), the relative error of approximation is smaller than L/M which is a small fraction for $M >> L$.

Figure 3.1 shows the cumulative distribution function (cdf) of the delivery probability in a single period spray and wait algorithm for different L values. Clearly, when L increases, the mean value ($1/\lambda = EM/L$) of exponential cdf decreases and the expected delay together with the time needed to reach the desired delivery probability shrinks.

Our main contribution is to introduce and analyze the multi-period spraying algorithm and to show under what condition it is more effective than the single period spraying. In our algorithm, spraying of message copies is defined by the urgency of meeting the desired delivery probability by the given delivery deadline.

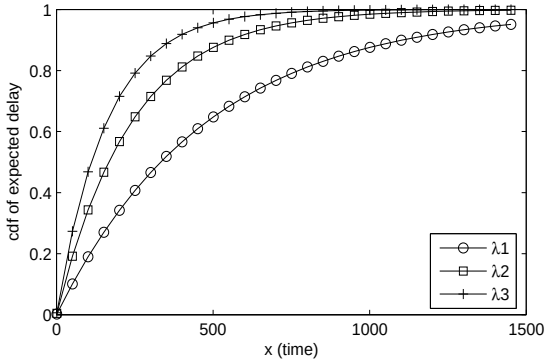


Figure 3.1: The cumulative distribution function of probability of meeting the expected delay in the Spray and Wait algorithm for different values of λ, where $\lambda_1 > \lambda_2 > \lambda_3$.

More precisely, the algorithm starts with spraying fewer message copies than the minimum L needed by the single spraying algorithm, and then waits for a certain period of time to see if the message is delivered. When the delivery does not happen, the algorithm sprays some additional copies of a message and again waits for the delivery. This process repeats until either the message is delivered[2] or the delivery deadline passes. Hence, as the time remaining to the delivery deadline decreases and delivery has not yet happened the number of nodes carrying the message copy increases.

Figure 3.2 summarizes what our algorithm is designed to achieve. In this specific version of the algorithm, we allow two different spraying phases. The first one starts without delay and the second one starts at time x_d. The main objective of the algorithm is to attempt delivery with small number of copies and use the large number of copies only when this attempt is unsuccessful. With proper setting, the average number of copies sprayed in the network until the delivery time can be lower than in the case of spraying all messages without delay, while the delivery rate by the deadline remains the same.

To analyze the performance of our algorithm analytically, we need to derive two formulas; one for the average number of copies used by the algorithm, and the

[2]At the end of Section 3.4, we explain how the delivered messages are acknowledged to other nodes.

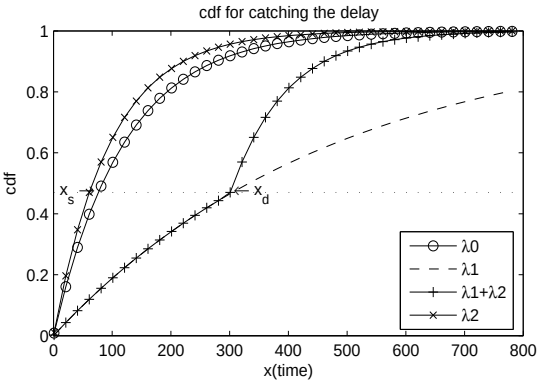


Figure 3.2: The cumulative distribution function of delivery time of a message when spraying different number of copies in two different periods.

second one for the cumulative distribution of the probability of meeting the delivery deadline with the increasing number of copies (and therefore with the increasing λ values).

In our scheme, the term *period* refers to the time duration from the beginning of one spraying phase to the beginning of the next spraying phase. There may be multiple spraying phases and the corresponding periods between them, each of different length. In the next section, we start with the analysis of the two period case to find the optimal period length and the corresponding copy counts for each period. Then, in the following sections we analyze the three and multiple period cases.

3.1 Two Period Case

If there are two periods until the message delivery deadline, the questions that need to be answered are *"when to finish the first period and start the second one"* and *"how many copies should be allowed in each"*. In other words, what should be the value of x_d in Figure 3.2 to minimize the average number of copies used by the algorithm? and how many copies should be sprayed in each period?

Let's assume that the single period spray and wait algorithm uses L copies (including the copy in the source node) of a message to achieve the probability $p_d \approx 1$ of its delivery by the deadline t_d. Let's further assume that the *Two Period*

Spraying algorithm sprays L_1 copies to the network at the beginning of execution and additional $L_2 - L_1$ copies at time x_d, the beginning of the second period. Then, the cumulative distribution function of the probability of delivering the message at or before time x is:

$$cdf(x) = \begin{cases} 1 - e^{-\alpha L_1 x} & \text{if } x \leq x_d \\ 1 - e^{-\alpha L_2 (x - x_s)} & \text{if } x > x_d \end{cases}$$

where, $\alpha = \frac{1}{EM}$ is the inverse of the expected inter-meeting time between two consecutive encounters of any pair of nodes and x_s is the delay with which the spraying with L_2 copies would need to start to match the performance of our algorithm in the second period (See Figure 3.2). The value of the x_s can be found from the equality of respective cdf functions at time x_d:

$$\begin{aligned} 1 - e^{-\alpha L_1 x_d} &= 1 - e^{-\alpha L_2 (x_d - x_s)} \\ x_s &= x_d \frac{L_2 - L_1}{L_2} \end{aligned}$$

The expected delivery rate when L copies are used in the single period spray and wait algorithm is by definition $p_d = 1 - e^{-\alpha L t_d} \approx 1$. Our objective is to match this delivery rate by decreasing the average number of copies below L. Hence, by the delivery deadline, t_d, the following inequality must be satisfied:

$$\begin{aligned} 1 - e^{-\alpha L_2 (t_d - x_s)} &\geq 1 - e^{-\alpha L t_d} \\ L_2 \left(t_d - x_d + x_d \frac{L_1}{L_2} \right) &\geq L t_d \end{aligned}$$

We can use this inequality to bound x_d as $x_d \leq t_d \frac{L_2 - L}{L_2 - L_1}$. As x_d gets larger, the average copy count gets lower when L_1 and L_2 values are same. Since our algorithm aims at decreasing the average copy count while maintaining the delivery rate of the single period spraying algorithm, then the optimal x_d must be the largest possible and therefore:

$$x_d = t_d \frac{L_2 - L}{L_2 - L_1}$$

We want to minimize the average number of copies, $c_2(L_1, L_2)$ defined as:

$$\begin{aligned} c_2(L_1, L_2) &= L_1(1 - e^{-\alpha L_1 x_d}) + L_2 e^{-\alpha L_1 x_d} \\ &= L_1 + (L_2 - L_1)e^{-\alpha L_1 x_d} \end{aligned}$$

Note that if the message is not delivered in the first period, then the cost becomes L_2 copies. Substituting x_d in the above, we get:

$$c_2(L_1, L_2) = L_1 + (L_2 - L_1)e^{-\alpha L_1 t_d \frac{L_2 - L}{L_2 - L_1}}$$

Taking derivative of $\frac{dc_2}{dL_2}$, we obtain:

$$\frac{dc_2}{dL_2} = \left(1 - \alpha L_1 t_d + \alpha L_1 t_d \frac{L_2 - L}{L_2 - L_1}\right) e^{-\alpha L_1 t_d \frac{L_2 - L}{L_2 - L_1}}$$

We are only interested in the sign of this derivative, so we can ignore always positive factor $e^{-\alpha L_1 t_d \frac{L_2 - L}{L_2 - L_1}}$. For the same reason we can also multiply the result by always positive factor $L_2 - L_1$. As we consider only values $L_2 > L_1$, then we obtain:

$$sgn\left(\frac{dc_2}{dL_2}\right) = sgn\left(L_2 - L_1 - \alpha L_1 t_d (L - L_1)\right)$$

We conclude that sign of derivative changes only once, at:

$$L_2^* = L_1 + \alpha L_1 t_d (L - L_1) > L_1$$

and it changes from negative to positive so the cost function has the unique minimum at this point. Hence, $L_2^* - L_1 = \alpha L_1 t_d (L - L_1)$ and therefore:

$$c_2^*(L_1) = L_1[1 + \alpha t_d (L - L_1)e^{-\alpha L_1 t_d + 1}]$$

Again, by taking the derivative of c_2^* in regard of L_1, and comparing it to zero, we can obtain the optimum value of L_1. Let $z = \alpha t_d L_1 - 1$ and $A = \alpha t_d L$, then:

$$c_2^*(z)\alpha t_d = 1 + z + (A - 1 + (A - 2)z - z^2)e^{-z}$$

Setting $f(z) = c_2^*(z)\alpha t_d$ we get

$$f(z) = 1 + z + (A - 1 + (A - 2)z - z^2)e^{-z}$$

Taking derivative of this function in regard of z, we obtain:

$$f'(z) = 1 + e^{-z}(-1 - Az + z^2)$$

Then, by multiplying by an always positive factor e^z we obtain:

$$g(z) = f'(z)e^z = e^z + z^2 - Az - 1$$

We notice that because the number and types of extreme points of a function are defined by the number of zeros and the derivative signs near these zeros, functions $g(z)$ and $c_2^*(L_1)$ have the same number and types of extreme points related by equation $z_e = \alpha t_d L_{1e} - 1$.

The first zero of function $g(z)$ is at $z = 0$, and it corresponds to maximum when $A > 1$, because near zero, $g(z) \approx 1 + z - Az - 1 = -z(A - 1)$, so it is positive for $z < 0$ and negative for $z > 0$. If $A < 1$, then of course the point $z = 0$ is at the minimum, and it corresponds to $L_2 = L$ which means that the one-time spraying is optimal in such a case. For $A = 1$, the point $z = 0$ is neither, but then the derivative is non-negative for $z > 0$. So, again one-time spraying is optimal. However, the condition that $A > 1$ is satisfied in realistic applications because $p_d = 1 - e^{-A} > \frac{e-1}{e} \approx 63\%$. Usually, the reasonable values for p_d are in high 90's percents which means that the introduced algorithm will perform better when reasonable delivery rate by the deadline is required.

More formally, for $0 < 4z < \min(A - 1, 24/7)$ we have:

$$\begin{aligned} e^z &= \sum_{i=0}^{\infty} \frac{z^i}{i!} < 1 + z + \frac{z^2}{2} + \frac{z^3}{6} \sum_{i=0}^{\infty} \left(\frac{z}{4}\right)^i \\ &= 1 + z + \frac{z^2}{2} + \frac{2z^3}{12 - 3z} < 1 + z + z^2, \end{aligned}$$

and $2z^2 - Az < -z$, so $g(z) < 1 + z - z - 1 = 0$. Similarly, for $-1 < z < 0$ we have

$e^z > 1 + z$, so $g(z) > 1 + z + z^2 - Az - 1 = z(z - (A - 1)) > 0$. This also means that there is at least one more zero point, as $g(z)$ is a continuous function, negative in close positive neighborhood of 0 and positive in $+\infty$. Moreover, considering the equality of two functions:

$$e^z + z^2 - 1 = Az$$

Algorithm 1 FindOptimalsInTwoPeriods(L, α,t_d)

```
1: opt_cost = L; opt_cts = [L, L]
2: for each 0 < L_1 < L do
3:    L_2floor = max(L + 1, L_1 + ⌊αL_1 t_d(L − L_1)⌋)
4:    for L_2 = L_2floor, L_2floor + 1 do
5:       if c_2(L_1, L_2) <opt_cost then
6:          opt_cost = c_2(L_1, L_2); opt_cts = [L_1, L_2]
7:       end if
8:    end for
9: end for
10: return opt_cts
```

We notice that $e^z + z^2 + 1$ is a convex function which have at most two intersection points with any straight line, including the line Az. Hence, $e^z + z^2 + 1$ and Az intersect at $z = 0$ and, for $A > 1$, exactly once more, at the point corresponding to the minimum, which is the point of interest to us here. Indeed, let $z_{opt} > 0$ denote the nearest to 0 intersection point of these two functions. From the convex property of the first function it follows that to the right of z_{opt}, the first function is always above the straight line Az. Hence, these two functions cannot intersect for $z > z_{opt}$. Furthermore, it must be that $z_{opt} < A$ as for $z \geq A$ we have $e^z + z^2 - 1 > 1 + A + Az - 1 > Az$, so the cost function is already growing.

We conclude that there is a unique optimum point at $z_{opt} > 0$, if and only if $A > 1$, or in other words, if and only if the required delivery rate by the deadline is greater than $1 - \frac{1}{e}$, or succinctly, $p_d > 1 - \frac{1}{e} \approx 63\%$, a very reasonable condition for practical solutions. This point can be found in $\lceil \log A \rceil$ steps by bisecting the interval $(0, A)$ until we get the range of the solution within two consecutive integers. Then, we can use the floor and ceiling of the approximation to find integer solution that

we are interested in. Complexity of this algorithm is low, $O(\log A)$, and because $A = -\ln(1-p_d)$, A is the natural logarithm of the inverse of the miss (non-delivery) rate of messages delivery by the end of the deadline. Hence the complexity is the polylogarithmic function of the inverse of delivery miss rate.

We can also find the optimal values of L_1 and L_2 by a simpler method, which generalizes nicely to cases with more periods, so we will present it here. From the equation defining $c_2(L_1, L_2)$, it is clear that the average number of copies sprayed by our algorithm is larger than L_1, so for our algorithm to be able to decrease the average number copies below L, L_1 must be smaller than L. As a result, the following boundaries for L_1 must hold:

$$0 < L_1 < L$$

Since the possible values for all L_1 variables are integers, we can use enumeration method as explained in Algorithm 1 and obtain the optimal values relatively quickly, in $O(L)$ steps. With constant values of EM and t_d, this is a logarithmic function of the inverse of delivery miss rate, so growing faster than complexity of finding a solution via the derivative of the cost function.

3.2 Three Period Case

In this section, we assume that there are three spray and wait periods until the delivery deadline. This time, we need to find two different boundary points which separate these three periods. Let x_{d_1} and x_{d_2} denote these boundary points. While the former stands at the boundary between the first and the second periods, the latter marks the boundary between the second and the third periods. The cumulative distribution function of the probability of delivering the message by the time x is:

$$cdf(x) = \begin{cases} 1 - e^{-\alpha L_1 x} & [0, x_{d_1}] \\ 1 - e^{-\alpha L_2 (x - x_{s_2})} & (x_{d_1}, x_{d_2}] \\ 1 - e^{-\alpha L_3 (x - x_{s_3})} & (x_{d_2}, x] \end{cases}$$

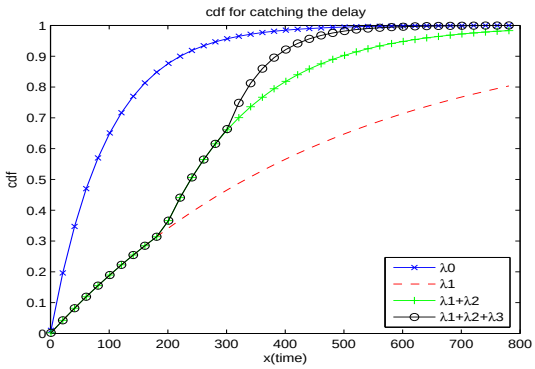


Figure 3.3: The cumulative distribution function with spraying different number of copies in three different periods.

where x_{s_2} and x_{s_3} are the delays with which the second (L_2) and the third (L_3) spraying would have to start to equal the cdf of our algorithm over the second and third spraying periods, respectively. As before, using the equality of the functions at times x_{d_1} and x_{d_2}, we can obtain the values of x_{s_2} and x_{s_3}:

$$\begin{aligned} 1 - e^{-\alpha L_1 x_{d_1}} &= 1 - e^{-\alpha L_2 (x_{d_1} - x_{s_2})} \\ x_{s_2} &= x_{d_1} \frac{L_2 - L_1}{L_2} \end{aligned}$$

and analogously:

$$\begin{aligned} 1 - e^{-\alpha L_2 (x_{d_2} - x_{s_2})} &= 1 - e^{-\alpha L_3 (x_{d_2} - x_{s_3})} \\ x_{s_3} &= x_{d_2} \frac{L_3 - L_2}{L_3} + x_{d_1} \frac{L_2 - L_1}{L_3}. \end{aligned}$$

Figure 3.3 illustrates our approach with three periods. Similar to the two period case, we want to achieve the same or higher delivery rate p_d at the given deadline t_d while minimizing the average number of copies used. That is, we need to satisfy the following inequality:

$$\begin{aligned} 1 - e^{-\alpha L t_d} &\leq 1 - e^{-\alpha L_3 (t_d - x_{s_3})} \\ L t_d &\leq L_3 (t_d - x_{s_3}) \\ x_{d_2}(L_3 - L_2) + x_{d_1}(L_2 - L_1) &\leq t_d (L_3 - L) \end{aligned}$$

Using this inequality, we can eliminate x_{d_2} because as x_{d_2} gets larger, the average copy count gets smaller when all other parameters L_1, L_2, L_3, x_{d_1} are kept constant. Therefore, replacing the above inequality with an equation, we obtain:

$$x_{d_2} = \frac{t_d(L_3 - L) - x_{d_1}(L_2 - L_1)}{L_3 - L_2}$$

Algorithm 2 FindOptimalsInThreePeriods(L)

```
1: opt_cost = L; opt_cts = (L,L,L)
2: for each 0 < L_1 < L do
3:    L_{2_Bound}(L_1) = ⌈L_1 + (L − L_1)e^{αL_1 t_d}⌉
4:    for each L_1 < L_2 ≤ L_{2_Bound}(L_1) do
5:       if L_2 ≥ L and c_2^*(L_1, L_2) < opt_cost then
6:          opt_cost = c_2^*(L_1, L_2)
7:          opt_cts = (L_1,L_2,L_2)
8:       end if
9:       if L_1 < (L_2 + 1)e^{−αL_2 t_d(L_2+1−L)} then
10:         L_{3_opt} = max(L,L_2)+1
11:         if c_3^*(L_1, L_2, L_{3_opt}) < opt_cost then
12:            opt_cost = c_3^*(L_1, L_2, L_{3_opt})
13:            opt_cts = (L_1,L_2,L_{3_opt})
14:         end if
15:      else
16:         R_1 = L_1 + (L_2 − L_1)ln(L_1) + αL_2 t_d(L − L_1)
17:         R_2 = R − (L_2 − L_1)ln(R)
18:         L_{3_opt} = Find optimum L_3 by bisecting in [R_2, R_1]
19:         if c_3^*(L_1, L_2, L_{3_opt}) < opt_cost then
20:            opt_cost = c_3^*(L_1, L_2, L_{3_opt})
21:            opt_cts = (L_1,L_2,L_{3_opt})
22:         end if
23:      end if
24:   end for
25: end for
26: return opt_cts
```

Furthermore, the average copy count used in this three period spraying can be defined as:

$$\begin{aligned} c_3(L_1, L_2, L_3, x_{d_1}) &= L_1 + (L_2 - L_1)e^{-\alpha L_1 x_{d_1}} \\ &\quad + (L_3 - L_2)e^{-\alpha L_2(x_{d_2} - x_{s_2})} \end{aligned}$$

When we substitute x_{s_2} and x_{d_2} in $c_3(L_1, L_2, L_3, x_{d_1})$ and take the derivative $\frac{dc_3}{dx_{d_1}}$, we obtain:

$$\frac{dc_3}{dx_{d_1}} = -\alpha(L_2 - L_1)L_3 e^{-\alpha L_1 x_{d_1}} \left(\frac{L_1}{L_3} - e^{-\frac{\alpha L_2(t_d(L_3-L)-x_{d_1}(L_3-L_1))}{(L_3-L_2)}} \right)$$

After ignoring the always positive factors, the sign of the derivative becomes:

$$sgn\left(\frac{dc_3}{dx_{d_1}}\right) = -sgn\left(L_1 - L_3 e^{-\frac{\alpha L_2(t_d(L_3-L)-x_{d_1}(L_3-L_1))}{(L_3-L_2)}}\right)$$

We see that if $L_1 < L_3 e^{-\frac{\alpha L_2 t_d(L_3-L)}{(L_3-L_2)}}$, the sign is always positive (note that $x_{d_1} \geq 0$ by definition), that is the cost function is always growing with increasing L_3. Therefore, minimum cost is obtained at $x_{d_1} = 0$. Otherwise, we notice that the sign of the derivative changes from negative to positive only once at:

$$x_{d_1} = \frac{\alpha t_d L_2(L_3 - L) + \ln(L_1/L_3)(L_3 - L_2)}{\alpha L_2(L_3 - L_1)} \tag{3.1}$$

For both cases, we obtained the optimum values of x_{d_1}. Then, we can easily obtain formula $c_3^*(L_1, L_2, L_3)$ by substituting x_{d_1} with these optimum values in corresponding conditions. Since $L_1 < L < L_3$ and $L_1 \leq L_2 \leq L_3$ and all these values are integers, by enumeration explained in Algorithm 2, we can simply find the copy counts (L_1, L_2, L_3) that gives the minimum copy count for a given L.

However, to use enumeration, we need to establish bounds on both L_2 and L_3. Using inequality $x_{d_1}<t_d$ that must be satisfied for the second period to start before the deadline for message delivery, we can calculate the upper bound for L_2, denoted as $L_{2_{Bound}}(L_1)$, as follows:

$$\begin{aligned} L &> c_3^* > L_1 + (L_2 - L_1)e^{-\alpha L_1 t_d} \\ L_2 &< L_1 + (L - L_1)e^{\alpha L_1 t_d} = L_{2_{Bound}}(L_1) \end{aligned}$$

Now, we have the ranges for L_1 and L_2. So, for a given (L_1,L_2), we will try to obtain the optimum L_3 value that makes the cost function minimum. When we

take the derivative $\frac{dc_3}{dL_3}$, we obtain:

$$\begin{aligned} \frac{dc_3}{dL_3} &= \left(1 + \frac{\alpha L_2(t_d(L_2 - L) - x_{d_1}(L_2 - L_1))}{(L_3 - L_2)}\right) e^m \\ \text{where } m &= \frac{-\alpha L_2 t_d(L_3 - L)}{(L_3 - L_2)} + x_{d_1}\frac{\alpha L_3(L_2 - L_1)}{(L_3 - L_2)} \end{aligned}$$

Since, we are interested in the sign of the derivative, we can ignore the always positive factor. Then, we have:

$$sgn\left(\frac{dc_3}{dL_3}\right) = sgn\left(1 + \frac{\alpha L_2(t_d(L_2 - L) - x_{d_1}(L_2 - L_1))}{(L_3 - L_2)}\right)$$

As we consider the values $L_3 > L_2$, we conclude that the sign of the derivative changes from negative to positive only once at:

$$L_3^* = L_2[1 + \alpha(x_{d_1}(L_2 - L_1) - t_d(L_2 - L))]$$

When we substitute x_{d_1} in this equation with the optimum x_{d_1} in Eq.3.1, we have the following equation:

$$L_3 + (L_2 - L_1)\ln(L_3) = L_1 + (L_2 - L_1)\ln(L_1) + \alpha L_2 t_d(L - L_1)$$

Since to find the value of $L_{3_{opt}}$ in the above function is not so easy, we can instead find a range for $L_{3_{opt}}$ then by bisecting within the range we can reach the optimum integer value of L_3. It is obvious that $L_{3_{opt}}$ satisfies the following:

$$L_{3_{opt}} < L_1 + (L_2 - L_1)\ln(L_1) + \alpha L_2 t_d(L - L_1) = R_1$$

On the other hand, $L_{3_{opt}}$ is also bigger than $R_1 - (L_2 - L_1)\ln(R_1)$ because

$$L_{3_{opt}} = R_1 - (L_2 - L_1)\ln(L_{3_{opt}}) > R_1 - (L_2 - L_1)\ln(R_1) = R_2$$

Therefore, $L_{3_{opt}}$ lies within the range $[R_2, R_1]$. Since as L_3 increases, the value of $L_3 + (L_2 - L_1)\ln(L_3)$ increases, we can find this integer optimum value of L_3 by

bisecting in this range. One can easily see that the complexity of this bisecting search is $O(log_2(L_2))$.

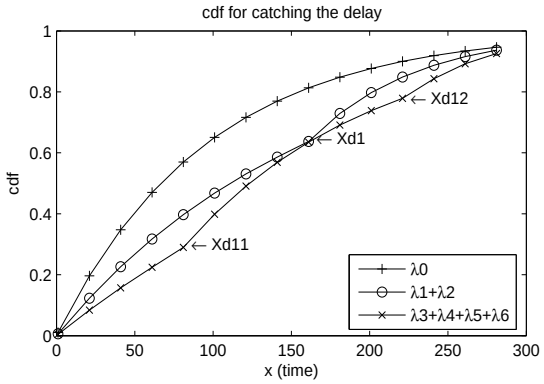


Figure 3.4: Recursive partitioning algorithm to define more periods of spraying and further decrease the total cost of spraying.

In Algorithm 2, for each (L_1,L_2) pair, we first find the optimum L_3 minimizing the cost function then we compare it with the current optimum cost. Here note that, if $L_1 < L_3 e^{-\frac{\alpha L_2 t_d (L_3 - L)}{(L_3 - L_2)}}$, then $c_3^*(L_1, L_2, L_3)$ is obtained by using the optimum $x_{d_1} = 0$. Otherwise $c_3^*(L_1, L_2, L_3)$ is computed using the optimum x_{d_1} value given in Eq.3.1.

To assess complexity of Algorithm 2, we observe that $L_{2_{Bound}}(L_1)$ can be approximated as follows:

$$L_2 < L_1 + (L - L_1)e^{\alpha L_1 t_d} < L_1 + \max_{x \in (1, L-1)} (xe^{-\alpha t_d x})e^{\alpha L t_d} \leq L + \frac{1}{e\alpha t_d (1 - p_d)},$$

because function $f(x) = xe^{-\alpha t_d x}$ has derivative $(1 - \alpha t_d x)e^{-\alpha t_d x}$ and therefore maximum at $x = \frac{1}{\alpha t_d}$. Hence, the complexity is:

$$\sum_{L_1=1}^{L-1} \sum_{L_2=L_1+1}^{L_1 + \frac{1}{e\alpha t_d (1-p_d)}} O(log_2(L_2))$$

In conclusion, the complexity of enumeration in this case is $O\left(-\frac{L log_2(e\alpha t_d(1-pd))}{e\alpha t_d(1-pd)}\right)$ so it is inversely proportional to non-delivery probability times logarithm of the inverse of non-delivery probability.

3.3 Increasing the Number of Periods by Recursive Partitioning

In this section, we show that by applying recursive partitioning of each period, more spraying periods can be created in such a way that the total cost of spraying can be decreased even more. An example is given in Figure 3.4. From *Two Period Case* section, we know how to achieve the optimum partitioning of the entire time interval from the start to the delivery deadline into two periods. However, it is also possible to partition each of these two periods individually to decrease the cost of spraying even further. Although this may not be the optimal partitioning in the resulting number of periods, it still decreases the spraying cost.

If we want to have three periods until the message delivery deadline, we can either partition the first period (with parameter $\lambda 1$) or the second period (with $\lambda 2$) and select the one which achieves the lower cost. In other words, we need to select either ($\lambda 3$, $\lambda 4$, $\lambda 2$) or ($\lambda 1$, $\lambda 5$, $\lambda 6$) as the exponential factors in the corresponding three exponential functions. Furthermore, after obtaining the three period spraying, we can run the same algorithm to find a lower cost spraying with four periods. However, we need to partition each period carefully considering the boundaries of possible L_i values.

Algorithm 3 IncreasePartitions(k,$x_d[\,]$,$L[\,]$)

```
1: min_cost = current copy cost with k periods
2: for each 1≤i≤k do
3:    [x'_d,L'] = PartitionIntoTwo(i,x_d[ ],L[ ])
4:    c = Cost(k + 1,x'_d,L')
5:    if c<min_cost then
6:       p = [x'_d,L']
7:       min_cost = c
8:    end if
9: end for
10: return p
```

Assume that we currently have k periods of spraying. Let L_i denote the copy count after spraying in i^{th} period and x_{d_i} denote the end time of that period. Then, the cumulative distribution function of the probability of delivering the message by

Algorithm 4 PartitionIntoTwo(i,$x_d[\,]$,$L[\,]$)

1: $f_1 = cdf(x_{i-1})$
2: $f_2 = cdf(x_i)$
3: min_cost = $L_i(f_2\text{-}f_1)$ //current cost of period
4: **for** each $L_{i-1} < L_i^- < L_i$ **do**
5: **for** each $L_i^- < L_i^+ < L_{i+1}$ **do**
6: Compute x_{split} using Eq.3
7: Compute x_{s^-} using Eq.2
8: internal_cost = $L_i^-(f_2\text{-}f_1)+L_i^+(f_3\text{-}f_2)$
9: **if** internal_cost<min_cost **then**
10: min_cost = internal_cost
11: $x_{opt} = x_{split}$
12: $[L_{opt}^-, L_{opt}^+] = [L_i^-, L_i^+]$
13: **end if**
14: **end for**
15: **end for**
16: $x'_d[\,] = [x_{d_1},...,x_{d_{i-1}},x_{opt},x_{d_i},...,x_k]$
17: $L'[\,] = [L_1,...,L_{i-1},L_{opt}^-,L_{opt}^+,L_{i+1},...,L_k]$
18: return $[x'_d,L']$

the time x becomes:

$$cdf(x) = \begin{cases} 1 - e^{-\alpha L_1(x-x_{s_1})} & [0, x_{d_1}] \\ 1 - e^{-\alpha L_2(x-x_{s_2})} & (x_{d_1}, x_{d_2}] \\ ... & \\ 1 - e^{-\alpha L_k(x-x_{s_k})} & (x_{d_{k-1}}, x] \end{cases}$$

where x_{s_i} is the delay with which spraying with L_i copies would have to start to equal the cdf of our algorithm over the i^{th} spraying period, so of course $x_{s_1} = 0$ and for $i > 1$, we have:

$$x_{s_i} = \sum_{j=1}^{i-1} x_{d_j} \frac{L_{j+1} - L_j}{L_i} \tag{3.2}$$

This expression is easy to derive from the following simple iterative definition of x_{s_i} for $i > 1$ resulting from the equality of the respective exponential functions

at point $x_{d_{i-1}}$:

$$x_{s_i} = \frac{x_{s_{i-1}} L_{i-1} + x_{d_{i-1}}(L_i - L_{i-1})}{L_i}$$

We want to increase the number of periods to $k+1$ while decreasing the total cost for spraying with the same delivery rate at the delivery deadline. Algorithms 8 and 9 summarize the steps to achieve this goal.

Basically, we partition each period into two periods, one by one, to find the new cost for the current partitioning. Then, from these possible partitions, we select the one that achieves the lowest cost. For each period i, we need to find new number of copies L_i^-, L_i^+ to assign to each of the two newly created periods into which the original period is split. The delivery rate at the end of the both periods needs to stay unchanged but the average cost should be smaller than the original average cost of period i.

For each period being split, except the last one, there are the following bounds on those two numbers:

$$L_{i-1} < L_i^- < L_i^+ < L_{i+1}$$

We can also find an upper bound for the last period, which we will denote for convenience as L_{k+1}. Let x_{split} denote the boundary point in which the second inner period starts (i.e, the start of period for spraying additional L_i^+-L_i^- copies). The value of x_{split} can be found from the equality of the probability of message delivery by the ends of the original and the split periods:

$$\begin{aligned} 1 - e^{-\alpha L_i (x_{d_i} - x_{s_i})} &= 1 - e^{-\alpha L_i^+ (x_{d_i} - x_{s^+})} \\ L_i(x_{d_i} - x_{s_i}) &= L_i^+(x_{d_i} - x_{s^+}) \end{aligned}$$

Substituting x_{s_i} and x_{s^+} by the formula in Eq. 3.2, which clearly s^- and s^+ must also obey, we obtain:

$$x_{split} = \frac{x_{d_i}(L_i^+ - L_i) + x_{d_{i-1}}(L_i - L_i^-)}{L_i^+ - L_i^-} \tag{3.3}$$

For the last period k, we need to find an upper bound for the values of L_k^+ with given L_k^-. The cost of this last period in terms of average number of copies used is slightly different than the cost of other periods. Let p_k denote the probability of message delivery before the period k starts. Similarly, let p_{split} denote probability of message delivery before the second added period starts. Of course, $p_k \leq p_{split} \leq p_d$, where p_d denotes the probability of delivery of the message by the deadline t_d. The cost of the original period k can be simply written as:

$$Cost_k = (1 - p_k)L_k$$

whereas the cost of the split period k is:

$$\begin{aligned} Cost_{ksplit} &= (1 - p_k)L_k^- + (1 - p_{split})(L_k^+ - L_k^-) \\ &\geq (1 - p_k)L_k^- + (1 - p_d)(L_k^+ - L_k^-) \end{aligned}$$

Since we want $Cost_k > Cost_{ksplit}$, then the following inequality must hold:

$$(1 - p_k)L_k^- + (1 - p_d)(L_k^+ - L_k^-) < (1 - p_k)L_k$$

which yields the following upper bound for feasible values of L_k^+:

$$L_k^+ < L_k^- + (L_k - L_k^-)\frac{1 - p_k}{1 - p_d} = L_{k+1}$$

Algorithm 8 shows how the optimal partitioning of a single period $0 < i < k+1$ is found. For convenience, we denote $L_0 = 0$. For each pair of number (L_i^-, L_i^+) such that $L_{i-1} \leq L_i^- < L_i^+ \leq L_{i+1}$, the cost of spraying is found and optimal pair which gives the minimum cost is selected. Clearly, the complexity of this algorithm is $O(L^2)$.

3.4 Acknowledgment of Delivery

The descriptions of the most of the published routing protocols for delay tolerant networks do not contain details of how the nodes in the network learn about

the delivery of a message to the destination to avoid spraying after the message delivery. Yet, this is a crucial issue in our algorithm because it directly affects the cost of copying of messages. If a message is delivered to destination, but a specific node is not notified about the delivery, this node will continue spraying the message, increasing the average cost of copying.

We study two types of acknowledgments for notifying the nodes about the delivery of the messages.

TYPE I: When destination receives a message, it first creates an acknowledgment for that message and sends it to other nodes within its range, which is assumed to be same for all the nodes in this case. Then, using epidemic routing, this acknowledgment is spread to all other nodes whenever there is a contact between a node carrying the acknowledgment and a node without it. Note that, since the acknowledgment packets are much smaller than data messages, the cost of this acknowledgment epidemic routing is small compared to the cost of routing the data packets. More costly is the delay with which all nodes in the network learn about the delivery of the message. During this delay, there may be useless spraying of the already delivered message increasing the total cost of copying.

TYPE II: In this type of acknowledgment, we assume that the destination uses one time broadcast over the more powerful radio than the other nodes (the assumption often satisfied in practice) so the broadcast reaches all the nodes in the network. Like in the previous case, the acknowledgment message is short, so its broadcast is inexpensive. However, to make the scheme more efficient, we use the following epidemiology inspired idea.

We considered an environment in which at different times, individuals infected by different pathogens. Each pathogen has an incubation period during which the infected individual is not contagious. After the incubation period, the sick individual is contiguous and able to infect others. We assume that there are effective vaccines for all pathogens and we want to vaccinate the entire population with the proper mix of vaccines in the most efficient way. The best way to achieve this goal is to wait until the closest end of an incubation period of any infected individual and to apply the vaccines for all observed infections to the entire population at that time.

Such delayed vaccination campaign allows emergence of new infections, possibly with new types of pathogens, before letting sick individuals infect others. This approach minimizes the number of necessary vaccination campaigns, each with all vaccines necessary to stop already started epidemics.

Inspired by this idea, we use the following efficient acknowledgment scheme. As the destination receives messages, it waits until the closest period change time (x_d) of any of the received message approaches. At that time, the destination broadcasts an acknowledgment of all so far received messages. Hence, the destination broadcasts acknowledgments relatively infrequently, proportionally to a substantial fraction of the t_d, which is assumed large. Even though acknowledgments of some messages are delayed, spraying of any received messages after the delivery time are suppressed.

It is clear that the second case results in better performance of delayed spraying than the first one. However, it may require higher energy consumption. In simulations, we compare the performances of both types of acknowledgment by showing how they affect the results of our algorithm.

3.5 Simulation Model and Results

To evaluate our multi-period algorithm, we have developed a discrete event-driven simulator in Java. We performed extensive simulations with different parameters that may affect the performance of the proposed algorithm.

First of all, we compare the results of simulations with the analytical results that we have obtained in previous sections. Moreover, we also look at the effects of two different mobility models on the results.

We deployed M=100 mobile nodes (including the sink) onto a torus of size 300 m by 300 m. All nodes (except the sink that has high range of acknowledgment broadcast in TYPE II case) are assumed to be identical and their transmission range is set at $R = 10$ m (note that these parameters generate a sparse delay tolerant network which is the most common case in practice). The movements of nodes are decided according to two different mobility models [44]:

1. Random Walk Model:

   The speed of a node is randomly selected from the range [4, 13]m/s and its

direction is also randomly chosen. Then, each node goes in the selected random direction with the selected speed until the epoch lasts. Each epoch's duration is again randomly selected from the range [8, 15]s.

2. Random Waypoint Model:
   First, a new destination inside the network area is chosen randomly. Then the node moves towards that destination with a randomly selected speed from the range [4, 13]m/s.

When nodes move according to the above models with given parameters, the value of EM in the former and latter becomes 480s and 350s respectively (we both computed these values from the given parameters and validated the results by simulations).

Assuming[3] that the desired p_d by the given deadline t_d is 0.99, first we have found the optimum copy counts for both two period (2p) and three period (3p) cases using Algorithm 1 and Algorithm 2. Table 1 shows the values of these optimum L_i's for different t_d values as well as the minimum L value that achieves the desired p_d in single period (1p) spray and wait algorithm. Clearly, as the deadline decreases, L_{min} (minimum L achieving p_d by t_d) in 1p increases because more copies are needed to meet the desired p_d by the deadline. Such an increase is also observed for L_i values used in both 2p and 3p algorithms. It is also important to remark that the optimum L_i values are different for random walk and random waypoint models because the EM values generated in these two different settings are different. Although we mentioned previously that our algorithms are designed for the environments in which the deadline is not so tight with respect to EM value, in the simulations we also test our algorithms with tight deadlines (such as 200s and 250s)[4] to see how they perform in these cases. Moreover, for the optimum L_i values of three periods, we also ran Algorithm 8 and Algorithm 9 over the result that we obtained with Algorithm 1 and observed that the results closely match the optimum L_i values that we obtained

[3]We have selected a high desired delivery probability because it is the most likely case in real applications. However, we also look at the effects of different p_d values in later simulations.

[4]These values can surely be considered as tight deadlines because note that direct delivery (L=1) in single spraying can achieve p_d=0.99 at 2210s and 1610s in the given random walk and random waypoint models, respectively.

using Algorithm 2.

	Random Walk			Random Waypoint		
t_d	L_{min} in 1p	2p Opt L_i's	3p Opt L_i's	L_{min} in 1p	2p Opt L_i's	3p Opt L_i's
200	12	7,22	6,12,27	9	5,16	4,8,20
250	9	5,15	5,9,19	7	4,13	3,6,15
300	8	5,14	4,8,18	6	3,11	3,6,14
400	6	4,11	3,6,14	5	3,10	2,4,12
500	5	3,9	2,4,11	4	2,8	2,4,10
600	4	2,7	2,4,9	3	2,5	1,2,6
700	4	2,8	2,4,10	3	2,6	1,2,7
800	3	2,5	1,2,6	2	2,7	1,3,10
900	3	2,6	1,2,7	2	1,4	1,2,5

Table 3.1: Optimum L_i copy counts that minimize the average number of copies while preserving the desired probability of delivery.

We started by computing x_{d1}, x_{d2} and the optimum L_i values from theory. Then we performed simulations to find the average copy count used per message when these computed values are used. We have generated messages from randomly selected nodes to the sink node whose initial location was also chosen randomly. Furthermore, we used binary spraying while distributing the allowed copy counts in each period. All results are the average of 2000 runs.

In Figure 3.5 and Figure 3.6, we show the average copy counts obtained when the optimum L_i values are used in 2p and 3p versions of our algorithm and when the predefined random walk model is used. Our analysis defines the cost function as the average copy counts used per message at the exact delivery time and computes the optimum L_i values which minimize this cost function. Hence, to compare theory with simulations, we obtained the average copy counts in simulations using Type II acknowledgments. However, we also include the average copy counts obtained in simulations when Type I acknowledgment is used. From the results in both figures, we observe that analysis results are very close to Type II results but as the deadline gets tight, they become an upper bound for Type II results. This is because for the smaller values of t_d, the number of copies sprayed to the network increases (optimum L_i values in 2p and 3p are large due to large L_{min} in 1p) so that spraying period

takes longer. Besides this also increases the difference betweeen the average copy counts needed when Type I and Type II acknowledgments are used because as L_i values gets larger, more nodes carrying message copies need to be acknowledged about the delivery when Type I acknowledgment is used.

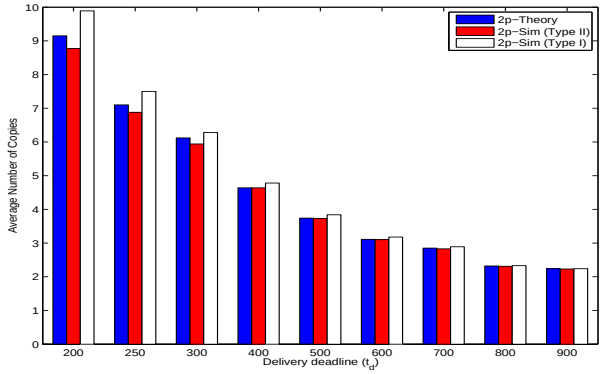


Figure 3.5: The comparison of the average number of copies obtained via analysis and simulation for the two-period case when random walk model is used.

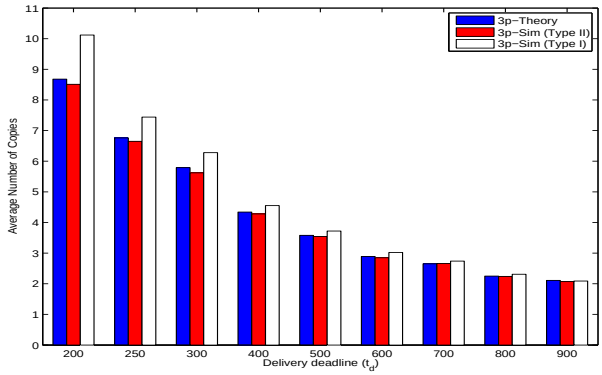


Figure 3.6: The comparison of the average number of copies obtained via analysis and simulation for the three-period case when random walk model is used.

We also compared the results when random waypoint mobility model is used. Figure 3.7 and Figure 3.8 show the comparison of average copy counts obtained in simulations with those computed analytically. The conclusions are similar to those made above for the random walk model, even though L_{min} values are different from

those used in the random walk model since the settings in this model generates an EM of 350s. This shows that our analysis holds for different mobility models. It only relies on the EM, the average intermeeting time between nodes for the applied mobility model.

To compare the performance of the proposed algorithms with the single period (1p) spraying algorithm (which is a special case of our algorithm), we first compare the average number of copies used in both algorithms when different types of acknowledgment mechanisms are used.

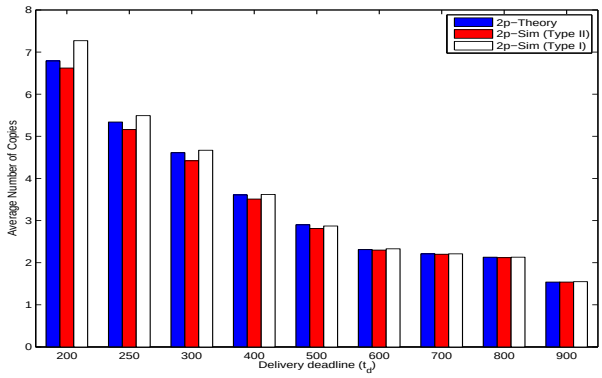


Figure 3.7: The comparison of the average number of copies obtained via analysis and simulation for the two period case when random waypoint model is used.

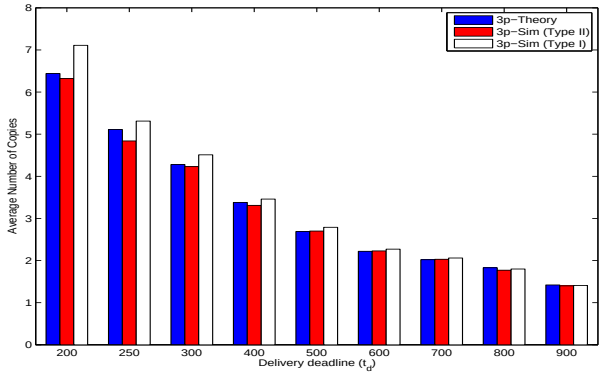


Figure 3.8: The comparison of the average number of copies obtained via analysis and simulation for the three period case when random waypoint model is used.

		Type I			Type II		
t_d	L_{min}	1p	2p	3p	1p	2p	3p
200	12	11.61	9.89	10.12	10.92	8.77	8.51
250	9	8.79	7.50	7.44	8.52	6.88	6.65
300	8	7.80	6.28	6.28	7.58	5.94	5.62
400	6	5.87	4.78	4.55	5.78	4.64	4.28
500	5	4.91	3.84	3.72	4.86	3.73	3.54
600	4	3.96	3.18	3.02	3.93	3.10	2.85
700	4	3.96	2.89	2.74	3.93	2.83	2.66
800	3	2.97	2.33	2.31	2.95	2.31	2.24
900	3	2.97	2.24	2.09	2.96	2.23	2.07

Table 3.2: Average number of copies used in single (1p), two-period (2p) and three-period (3p) spraying algorithms with different acknowledgment types and deadlines.

In Table 3.2, we present the average copy counts used in three compared algorithms when random walk model is used (we did not include the results when random waypoint model is used because they are similar to the results presented here). From the table, we observe that in both acknowledgment types, 3p algorithm uses fewer copies on average than either 2p or 1p spraying algorithm does. However, when Type I acknowledgment is used, the saving in the number of copies obtained by 3p algorithm decreases. Moreover, in some cases (t_d = 200s), its performance becomes worse than 2p algorithm. This is because when the deadline gets tight, the number of copies that are sprayed to the network increases so that the number of nodes carrying the message copies increases and the duration of epidemic like acknowledgment is longer. Consequently, more redundant copies are sprayed by the nodes having message copies before they are informed about the delivery. Moreover, we also notice that using the proposed algorithms even with Type I acknowledgment results in lower average copies used than when using the single period spraying algorithm with Type II acknowledgment. It should also be noted that in single period spraying algorithm with L copy count, the average number of message copies sprayed to the network is less than L. This is simply because even in single period spraying which does all spraying at the beginning, there is non-zero chance that the message will be delivered before all copies are made.

To further compare the performance of the proposed algorithms with the single period spraying algorithm, we have measured some additional metrics. Figure 3.9 and Figure 3.10 show the comparison of average message delivery delay and the average time of spraying completion[5] (time by which the last copy is sprayed) in these algorithms, respectively. Inspecting these two graphs, we observe that the proposed 2p and 3p algorithms incur higher average delay than 1p algorithm but they achieve the same delivery probability[6] before the deadline compared to the 1p spraying algorithm. Moreover, since the proposed algorithms postpone the spraying of all copies to later times, they finish spraying later than the single period spray and wait algorithm does. This results in lower memory usage averaged over execution time of our algorithm when compared with such usage incurred by the single period spraying algorithm.

We also computed the percentage of the savings achieved in the number of copy counts with the proposed multi-period algorithms. Figure 3.11 charts the fraction $(L\text{-}L_{avg})/L$ with the given t_d. Here, L is the average copy count used in single period spraying and L_{avg} is the average copy count achieved in the multi-period spraying algorithm. This time, we present the results when random waypoint model is used (the results with random walk model are similar). From the results shown in Figure 3.11, we observe that 3p algorithm provides higher savings than 2p algorithm. Moreover, it is clear that the savings with Type II acknowledgment are higher than the savings with Type I acknowledgment in both 2p and 3p algorithms. The difference between the savings of Type I and Type II acknowledgments gets smaller as the deadline increases. This is because larger t_d decreases the number of copy counts sprayed to the network, resulting in acknowledgments reaching all nodes

[5]The values in Figure 3.10 are computed over cases in which the message is delivered after all potential copies are sprayed.

[6]In simulations, we assume that collisions or collision avoidance do not impact message delivery. Indeed, they can only force meeting nodes to communicate sequentially, delaying some pairwise node communications. Yet, the average required communication time (about 0.1s with 1Mb/s bandwidth and 100Kb packets) is small compared to the average meeting time of two nodes (1.72s in random walk setting). Moreover, meeting of four or more nodes is very unlikely (below 1% in our setting). Thus, it is unlikely (below 0.05% in our setting), that a communication delay due to collision or collision avoidance will exceed meeting time, justifying our assumption. As expected, in simulations, all three algorithms achieve the desired p_d by the deadline (for the sake of brevity, the relevant plot is omitted here).

carrying message copies earlier. On the other hand, we also observe fluctuations even in the savings of a single algorithm with different delivery deadlines. This is because for some consecutive t_d values (i.e., t_d = 600s, 700s, 800s), L_{min} value in 1p algorithm which achieves the desired p_d is the same (i.e. $L_{min} = 3$) while L_i values in multi-period algorithms are different. In these cases, multi-period algorithms take the advantage of spraying in multiple periods and delay the spraying further when the deadline is larger (for example in 2p algorithm, when t_d =600s, then x_{d1} =400s and the optimum (L_1, L_2) = (2,5) but when t_d = 700s, then x_{d1} = 525s and the optimum (L_1, L_2) = (2,6)). Hence, multi-period algorithms can provide more saving over single period algorithm in such cases.

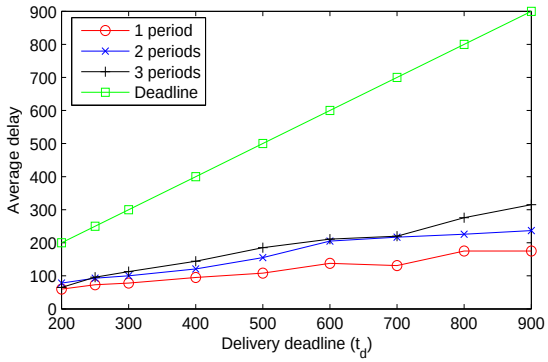


Figure 3.9: The comparison of the average delay for the single period and multiple-period algorithms (random walk model).

We also looked at the effects of the desired p_d on savings achieved by the proposed algorithms. As an example, we plotted the percentage of savings obtained in 2p-Type II algorithm with three different p_d values in Figure 3.12. Here, we performed simulations in a different way to show also the flat behavior of percentage of savings with respect to L (instead of t_d). With the given L and p_d values, we first found the minimum t_d value that achieves the given p_d (in random waypoint model) and then obtained the savings provided by 2p-Type II algorithm (when optimum L_i values are used) at that t_d value while maintaining the given p_d.

From Figure 3.12, we first observe that the savings are almost the same when plotted according to the L values, where t_d is the minimum time that spraying of L copies achieves the given p_d (proving this property analytically is the subject of

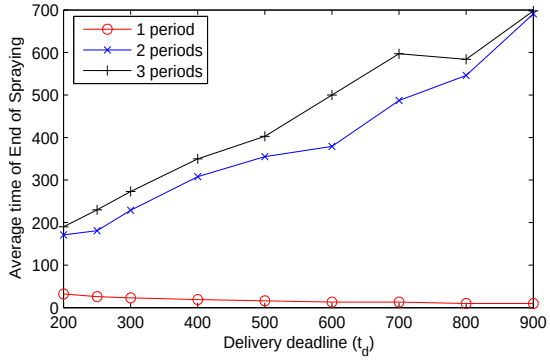


Figure 3.10: The comparison of average end of spraying times in the single period and multiple-period spraying algorithms (random walk model).

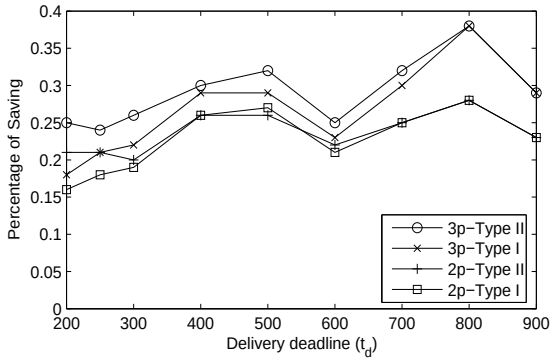


Figure 3.11: The percentage of savings achieved by the proposed algorithms with two different acknowledgment schemes (random waypoint model).

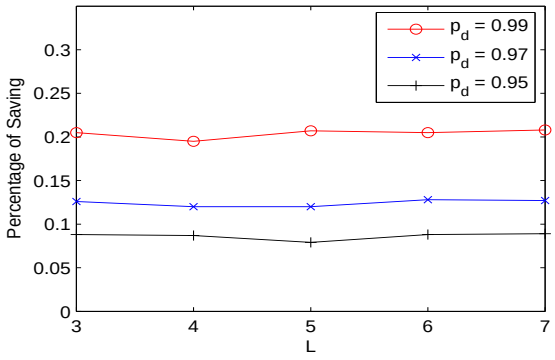


Figure 3.12: The percentage of savings achieved by 2p-Type II algorithm with three different p_d values (random waypoint model).

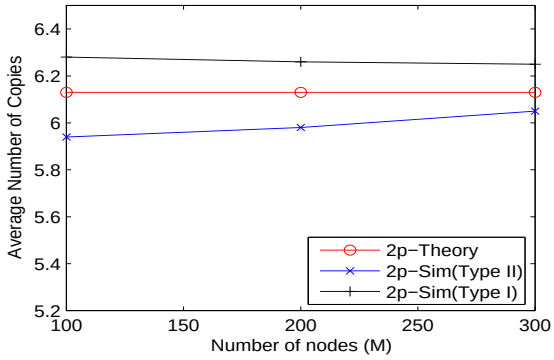


Figure 3.13: The effect of number of nodes on the difference between the analysis and simulations results.

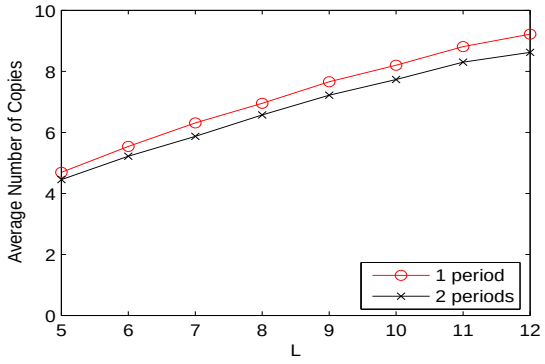


Figure 3.14: The average number of copies used per message in the simulations of real traces from RollerNet.

our future work). Additionaly, we observe that as the given p_d value decreases, the savings provided by multi-period algorithm decreases. This is because as p_d decreases, minimum t_d value achieving p_d with the given L decreases and the cdf of delivery probability gets more vertical around the t_d value. Because of these two reasons, the chance of saving in multi-period algorithm decreases with lower values of the desired p_d.

In the above simulations, we always assumed a constant number of nodes (M=100) in the network. However, the value of M affects the performance of the algorithm as well. For example, in Figure 3.13 we plot the simulation and analysis results in random walk model for 2p algorithm with three different M values (where

$p_d = 0.99$). It is clear that as M increases the difference between 2p-Sim (Type II) and analysis gets smaller. This is the result of fast spraying with increasing M[7]. Moreover, the difference between 2p-Sim (Type I) and 2p-Sim (Type II) results decreases because larger M values enable faster acknowledgment process.

In addition to the evaluation of the proposed protocol with random mobility models, we have also looked at its performance on real DTN traces. From the several data sets released so far, we have selected RollerNet [45] traces thanks to its easy usability (for example, all the meetings between nodes are mutually recorded). RollerNet traces include the opportunistic sightings of Bluetooth devices by groups of rollerbladers carrying iMotes in the roller tours in CityplaceParis. Since the nodes are not identical, the generated intermeeting times between pairs of nodes vary significantly. Although our protocol is not designed for networks with heterogeneous intermeeting times between nodes, we simply applied our multi-period spraying idea using the results from our analysis and left the design of an algorithm specifically for heterogeneous networks as a future work.

The RollerNet traces starts at 1156084064s and ends at 1156094040s. In each 10s starting from the beginning (until the time after which the last message will not have enough time to be delivered), we have generated a message from a random source to a random destination in the network. In single period routing, for each L (number of copies allowed), we found the delivery times of each message and also the time (we call it discovered t_d) at which 99% (p_d) of all messages are delivered since their generations at the source nodes. Then, we ran the two period routing on the same traces with the same set of messages. In the first period, we allowed the spraying of $L/2$ copies (which is the most frequent case according to the results of our analysis). In the second period we tried different copy counts and found the necessary copy count that achieves the same delivery rate of all messages by the discovered t_d. Since the intermeeting times between nodes and also the delivery times of messages are different from each other, we computed the start of second period individually for each message. That is, for each message, we used its delivery time in the single period routing with L messages as the message's own t_d and

[7]It should be noted that EM does not change with increasing M. Only the rate of meeting with new nodes increases, which results in the fast spraying of messages.

computed x_d accordingly. In Figure 3.14, we show the average copy counts used per message in 1p and 2p spraying algorithms. Clearly, multi-period spraying idea can reduce the average copy count used in real DTN traces even when the frequencies of node meetings show heterogeneous behavior. The savings in this case are in the range of 6%-8%. However, we believe that a more careful design of multi-period idea can increase the savings even further. The design of a multi-period spraying based routing algorithm for heterogeneous networks will be the subject of our future work.

3.6 Summary of Contributions

In this chapter, we introduced a general multi-period spraying algorithm for DTNs which distributes the message copies depending on the remaining time to delivery deadline. Then, using formal analysis and simulations, we evaluated its performance. We first showed analytically how to partition time until deadline in a single period spraying algorithm into two and three separate periods, each period consisting of spraying phase followed by the wait phase. Then, we presented a generalization of this approach to a larger number of periods to reduce the cost even further. Finally, we discussed the results of simulations of our algorithm confirming that the average number of copies used by our algorithm is smaller than the average number of copies used by the single period spraying algorithm, while its delivery rate by the deadline matches the performance of the latter.

In the future work, we will investigate how more realistic radio links and mobility models affect our algorithm. Moreover, we also plan to update the proposed protocol for networks in which node meeting behavior varies between nodes.

CHAPTER 4
ROUTING WITH ERASURE CODING OF MESSAGES

In this chapter, we first overview erasure coding technique with comparison to replication based routing and formulate the problem in the context of deadline driven routing in DTNs. Then, we describe several message distribution schemes and analyze their performance in terms of message delivery delay and the total delivery cost. Finally, we present two different complementary techniques that can decrease the cost of routing in DTNs [53].

For easy description, we give a list of symbols and their meanings used in the rest of the chapter in Table 4.1.

4.1 Overview of Erasure Coding and Problem Description

Erasure coding (EC(k,R)) [26] is a coding scheme which processes and converts a message of k data blocks into a large set of Φ blocks such that the original message can be constructed from a subset of Φ blocks. Here, Φ is usually set as a multiple of k and $R = \Phi/k$ is called replication factor of erasure coding. Under *optimal erasure coding*, k blocks are sufficient to construct the original message. But, because of the fact that optimal coding is expensive in terms of CPU and memory usage, near optimal erasure coding is used requiring $k+\epsilon$ blocks to recover the original message. In [43], the average value of ϵ is reported as $k/20$ for Tornado codes. Therefore, following the previous studies, for simplicity we ignore ϵ.

There are various erasure coding algorithms including Reed-Solomon coding and Tornado coding. These algorithms differ in terms of encoding/decoding efficiency, replication factor R and minimum number of code blocks needed to recover the original message. Due to its simplicity and linear time complexity, we use Tornado codes in our scheme. The erasure/decoding complexity in Tornado coding is proportional to $\Phi \ln(1/(\epsilon - 1))P$ where P is the length of encoded packets.

We illustrate the basic principles of replication and erasure coding based routing in Figure 4.1. In a replication based routing with limited copying allowed, L

Table 4.1: Notations

Symbol	Definition
L	Number of copies of a message
M	Average size of a message (bytes)
k	Number of equal size blocks that a message is split into (k_{max} is upper bound for k)
R	Replication factor used in erasure coding of a message
Φ	$k \times R$, total number of blocks generated in erasure coding based routing
Φ_i	Total number of messages distributed to the network by the end of i^{th} period (L_i is used for same value in replication based routing)
R_{opt}	Optimum value of R in single period case
R^*	Replication factor used in multi-period case
t_d	Message delivery deadline (time units)
$P(x)$	Probability of delivery at time x
d_r	Desired delivery rate
τ	Total cost of delivery of a message
$1/\lambda$	Average inter-meeting time of nodes
T_s	End of distributing all messages
$EC(k, R)$	Erasure coding with parameters k and R
α	The percent of kR messages that are distributed in the first period of $EC(k, R)$

message copies are distributed to L distinct nodes in the network and the delivery of just one copy is needed in the destination node. In an erasure coding based routing, a message is divided into k blocks, transformed into $\Phi > k$ blocks of which any k blocks are sufficient for the destination node to reconstruct the original message. The most important advantage of erasure coding over replication based routing is that erasure coding algorithms strengthen the robustness of the network against failures (in other words, it increases reliability). That is, the more packets are spread to the network, the higher is the probability of message delivery to the destination, regardless of the rate of message failures.

In both algorithms, the delivery of the messages depends on the spread of many original or encoded copies of the message over the network. Therefore, any reduction

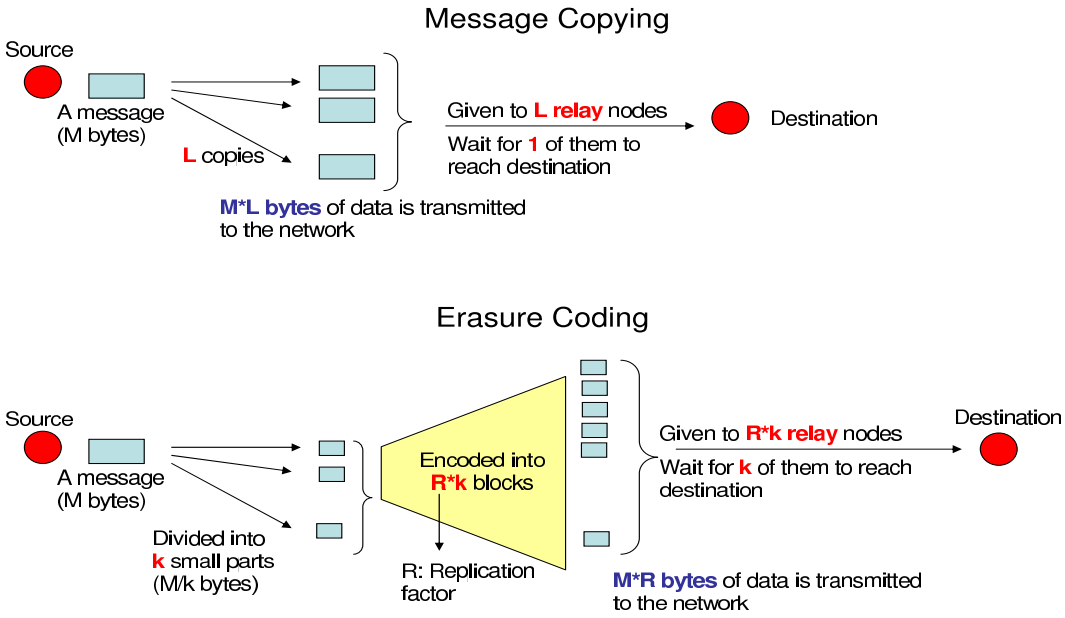


Figure 4.1: Comparison of replication based and erasure coding based routing.

of number of these copies without diverging from the goals (i.e. maintaining the desired delivery rate by the deadline) prolongs the network lifetime and enables the network to operate longer. Consider Figure 4.1 where the ideas of replication based routing and erasure coding based routing algorithms are illustrated. When R and L are equal, the cost (τ) is same in both techniques[8]. However, as it is seen in Figure 4.2, the cumulative distribution function (cdf) of delivery probability will be different. In erasure coding based routing, the message spraying duration takes more time because more messages are transferred to more relay nodes. Therefore, the delivery probability increases slowly in early times. Besides, once the spraying is finished, since k of the messages are expected to be delivered to the destination, the delivery probability still continues at lower rates for some time. However, when the expected delivery time for messages gets closer, k messages are easily and quickly gathered because of high number of messages spread to the network. Consequently, worst case delivery rate for the erasure coding is better than for the replication

[8]Here, note that when there are L copies given to L relay nodes, there will be $L+1$ copies of the message in the network including the one at the source node. The cost of message distribution is $O(ML)$, but including the delivery of one of them to the destination, the overall cost of delivery is $O(M(L+1))$ (or $\approx O(M(R+1))$ in erasure coding based routing with replication factor of R).

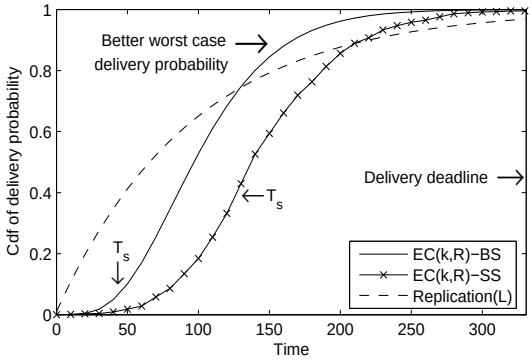


Figure 4.2: Comparison of delivery probabilities in erasure coding and replication based routing.

based routing.

In Figure 4.2, we also demonstrate the effect of different message distribution algorithms on the cdf of message delivery probability. When only source node is eligible to distribute data blocks to other nodes in the network (Source Spraying (SS)), the distribution period takes longer than distributing them with binary spraying (BS) [47]. Consequently, spraying ends (T_s) quite late, delaying the start of the gathering of k messages at the destination. We conclude that even the same number of messages are used, different delivery probabilities may be resulted due to the different message distribution schemes. In the next section, we present some message distribution schemes and discuss their effects on the delivery time and cost.

Considering all these aspects of erasure coding and replication based routing in the context of deadline driven routing, we formulate our problem as follows:

Assume that source node n_0 has a message of M bytes that needs to be delivered to the destination node d before the delivery deadline t_d with a desired delivery rate of d_r. What is the minimum cost of routing this message and which configuration achieves this? Formally, given t_d and d_r minimize τ subject to constraint $P(t_d) \geq d_r$.

4.2 Message Distribution Schemes

Distribution of messages is an important process that directly affects the performance of routing. Since the messages are copies of each other in replication based routing but they are encoded blocks with different contents in erasure coding algorithms, the cost of the distribution of the messages to other nodes in the network is different in these two cases. To the best of our knowledge, no previous work had analyzed the performance of these two routing approaches (in combination with different message distribution algorithms) in terms of both the delivery delay and the transmission cost. When the number of messages to be distributed in the network is high, as in the case of erasure coding algorithms, the distribution algorithm may have high impact on the performance. The faster the copies are distributed, the earlier the delivery process starts with full strength and the higher is the probability of message delivery by the deadline.

In [47], Spyropoulos et al. give the details of two different message copy distribution methods: Source Spraying and Binary Spraying. The authors proved that when node movements are independent and identically distributed (IID) random variables, binary spraying minimizes the expected time until all copies are distributed. However, we think that even when the node movements are IID and node meetings are exponentially distributed with a common mean value, message copies can be distributed faster than by binary spraying. Below we list four different spraying schemes and discuss their merits. The first two are the methods previously introduced in [47] and the latter two are the new spraying methods that we introduced here.

Assume that we have a network consisting of N nodes and the time passing between two consecutive encounters of a pair of nodes (n_1,n_2) is exponentially distributed with mean $EM = 1/\lambda$. Furthermore, assume that a source node n_0 has L (or Φ) messages and want to distribute them to other relay nodes in the network. There are several ways of achieving this task.

4.2.1 Source Spraying

The simplest way of distributing these messages is to have the source node copy the message to the first $L-1$ distinct nodes it encounters. Obviously, to be able to distribute all messages, source node needs to wait for meeting $L-1$ other nodes in the network. Therefore, the expected time to finish the distribution (T_s) is:

$$E[T_s] = \sum_{i=1}^{L-1} \frac{EM}{N-i} = EM(H_{N-1} - H_{N-L}),$$

where H_n is the nth harmonic number. If there are already i nodes having a message in the network, then the expected time of n_0 meeting with a node without a message is $EM/(N-i)$. The cost of source spraying in replication based routing with L copies is $O(ML)$. On the other hand, in erasure coding based routing with $\Phi = R \times k$ encoded blocks in total, the cost is $O(\frac{M\Phi}{k}) = O(MR)$. Both algorithms have equal cost when $R = L$. Here and also in the analysis of other spraying algorithms, we ignore the cost of checking whether the encountered node has the message. This cost is very small compared to the cost of transferring long messages.

4.2.2 Binary Spraying

This method takes the advantage of the speedup that parallel processing can deliver. When a node n_1 carrying messages encounters another node n_2 without a message, it gives $\lfloor \frac{L}{2} \rfloor$ of the messages to node n_2 and keeps the remaining $\lceil \frac{L}{2} \rceil$ for itself. This process continues until only at most one message remains in each node. When L is a power of two, a good approximation of $E[T_s]$ is:

$$E[T_s] = EM \sum_{j=0}^{\log(L)-1} E[Y(j,N)]$$

where $Y(j,N)$ is defined for $j \geq 0, N > 2^{j+1}$ as follows. Consider the binary tree representing node meetings. The nodes of the $(j+1)^{th}$ level get messages from upper level nodes on the average after $\frac{EM}{N-i}, i = 2^j, \ldots 2^{j+1}-1$ time units, respectively. The time by which the last node gets the message defines $E[T_s]$ eventually. We

have 2^j nodes receiving message at the same level. Let $X_1, \ldots X_{2^j}$ represent the exponential random variables of these nodes' message receiving times with rates $\frac{1}{N-2^j}, \cdots \frac{1}{N-2^{j+1}+1}$, respectively and let $Y(j,N) = \max\{X_1, \ldots X_{2^j}\}$. By expanding $F(x) = P(Y \leq x) = \prod_{i=0}^{2^j-1}(1 - e^{-(N-2^j-i)x})$ and computing[9] $\int_{x=0}^{\infty}(1 - F(x))dx$, we get:

$$E[Y(j,N)] = \sum_{t=1}^{2^j} \sum_{<i_1 \ldots i_t> \in U(2^j,t)} \frac{(-1)^{t-1}}{\sum_{m=1}^{t} N - 2^j - i_m},$$

where $U(z,t)$ for $1 \leq t \leq z$ is a set of t-tuples defined as $U(z,t) = \{< i_1, , \ldots i_t > \mid 0 \leq i_1 < \ldots < i_t < z\}$. Then, the value of $E[T_s]$ is the summation of $E[Y(j,N)]$s for all levels[10]. In the above formula, we ignored the cases in which a node that reached the current level early sprays the message to two nodes at the next level. Such cases happen with small probability, so have little impact on the result. We also assume that the last node in the current level will spray the last message, which again is the most likely outcome.

The cost in binary spraying is different in replication based routing and erasure coding based routing. In the former, only copies of the original message are generated, so a node with a right to make L copies do not need to give $\lfloor \frac{L}{2} \rfloor$ of them to the first node it meets. Instead it can give only one copy to this encountered node together with the right to make $\lfloor \frac{L}{2} \rfloor - 1$ more copies in the same manner. Consequently, the cost is kept at $O(ML)$. But in erasure coding based routing, the source node generates Φ encoded packets with different contents, so the same update in the message transfer process between nodes can not be applied. Hence, the cost of distributing Φ messages ($\tau(\Phi)$) in erasure coding based routing via binary spraying is:

$$\begin{aligned} \tau(\Phi) &= 2\tau(\Phi/2) + (M/k)(\Phi/2), \text{ where } \tau(1) = 0 \\ \tau(\Phi) &= \frac{M\Phi \log \Phi}{2k} = O(MR\log(kR)) \end{aligned}$$

[9]Clearly, the following integral is the same as $\int_{x=-\infty}^{\infty} xF'(x)dx$ for a non-negative random variable Y. For large N and small j we have $E[Y(j,N)] \approx \frac{EM}{2^j}(H_{N-2^j} - H_{N-2^{j+1}})H_{2^j}$.

[10]Hence, asymptotically, this average is proportional to $\ln(L)\log(L)$ while the average for source spraying is proportional to L.

If a node has Φ messages, it transfers half of them to the first node it meets with cost $(M/k)(\Phi/2)$ then each of these nodes continues distribution of their own messages independently, incurring the cost $\tau(\Phi/2)$.

4.2.3 Optimal (Fastest) Spraying

Although binary spraying achieves optimal spraying on average [47], the actual fastest distribution of messages in any case can be obtained by message exchange in the first $L-1$ meetings of two nodes, one of which has the message while the other does not. Formally, let $N^1(t)$ and $N^0(t)$ denote the set of nodes with message and without message at time t, respectively. Furthermore, let $r_{ij}(t)$ be the remaining time to the first meetings of nodes i and j at time t and let t_i be the time at which the i^{th} message is given to i^{th} node in the network. The fastest distribution of all messages results when t_i values satisfy the following equation:

$$\begin{aligned} t_i &= t_{i-1} + \min\{r_{ij}(t_{i-1})\}, \\ &\quad \text{where } i \in N^1(t_{i-1}) \text{ and } j \in N^0(t_{i-1}) \\ N^1(t_i) &= N^1(t_{i-1}) \cup \{j\} \\ N^0(t_i) &= N^0(t_{i-1}) - \{j\}, \\ &\quad \text{where } j = \arg\min\{r_{ij}(t_{i-1})\} \end{aligned}$$

Note that $t_0 = 0$ since a message (i.e. first one) will be kept in the source node n_0. Similarly, the expected time of distribution of all messages can be computed as:

$$E[T_s] = \sum_{i=1}^{L-1} \frac{EM}{i(N-i)}$$

The expected time until the meeting of any of the i nodes possessing messages with any of the nodes having no message is $\frac{EM}{i(N-i)}$. As a result, the total time needed to distribute all messages can be calculated by summing times $\frac{EM}{i(N-i)}$ for all i's from 1 to $L-1$.

The cost of optimal spraying is also different for replication based routing and erasure coding based routing. In the first one, at most one copy of the message is

transferred between nodes each time there is a meeting of two nodes. Therefore, the cost is $O(ML)$. However, in the second case (in which messages are different), the structure of the node meeting tree defines the cost. The lowest possible cost of copying ($O(MR)$) happens if only the source node meets other nodes in the network. The worst case, ($O(MkR^2)$), happens if the node meeting tree in the optimal spraying reduces to a line (i.e., first the source n_0 meets a node n_1 and transfers $\Phi - 1$ encoded blocks each of which is M/k bytes, then n_1 meets n_2 and transfers $\Phi - 2$ messages, and so on).

As a result, in optimal spraying, although the minimum $E[T_s]$ can be obtained easily, the cost can vary according to the node meeting schedule. Moreover, in optimal spraying, each node needs to know meeting times of all other nodes. Yet delay tolerant networks have mobile nodes moving independently. Hence, nodes cannot obtain such a knowledge most of the time. Therefore, the optimal spraying approach is inappropriate in this setting. Still, the resulting time is a useful lower bound for the duration of spraying stage.

4.2.4 Cooperative Binary Spraying

In this method, the messages are distributed in the same manner as in binary spraying except that when two nodes with messages meet each other, they equalize their message loads. In other words, if a node n_i with L_i messages meets another node n_j with L_j messages, the message load in each node is updated as follows:

$$\beta = L_i + L_j,\ L_i = \lceil \frac{\beta}{2} \rceil \text{ and } L_j = \lfloor \frac{\beta}{2} \rfloor$$

The advantage of cooperative spraying algorithm[11] over the original binary spraying method is the former's ability to deal with irregular patterns of node meetings. If a node n_1 (with many rights to copy a message) does not meet with nodes without a message for a long time (longer than expected), the rights to copy that it holds are unused, so that time of spraying T_s increases because of late distribution of some messages. Clearly, the other nodes can not be aware of this unexpected

[11] Here, we give an intuitive analysis of cooperative binary spraying showing its superiority over binary spraying leaving the more formal analysis as future work.

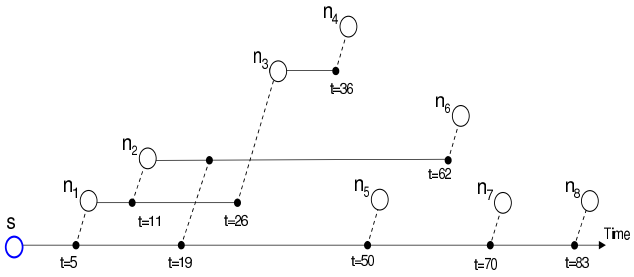


Figure 4.3: Node meeting times in a sample network. Each line shows the time-line of a node and the connections between time lines indicate meetings between the corresponding nodes.

Table 4.2: Execution of different message copy distribution algorithms on the network with node meeting times shown in Figure 4.3

Algorithm	$\mathbf{N^1(T_s)}$	$\mathbf{T_s}$
Source	n_0, n_1, n_2, n_5, n_7, n_8	83
Binary	n_0, n_1, n_2, n_3, n_5, n_7	70
Optimal	n_0, n_1, n_2, n_3, n_4, n_5	50
Co-Binary	n_0, n_1, n_2, n_3, n_5, n_6	62

situation of slow distribution. However, if any of the nodes already holding some messages meet with node n_1 holding more messages, it can help node n_1 in the distribution of its messages. As a result, the speed of binary spraying can be increased with such help, yielding smaller $E[T_s]$ than the one achieved in binary spraying. Such help incurs no cost in replication based routing but requires additional copying in erasure coding based routing.

To explain the differences of these spraying strategies, we will show the execution of each strategy on a sample network shown in Figure 4.3. In the figure, each node is accompanied with a time line showing meetings with other nodes. For example, source node n_0 meets node n_1 at time 5 and node n_2 at time 19. The table 4.2 shows the result of executing each scheme on this sample network when there are $L = 6$ copies at the source node in a replication based routing. In source spraying, source waits to meet five other nodes to distribute all copies. In binary spraying, first source gives three of the copies to node n_1 and both the node n_1 and n_0 distribute two more copies in parallel (while n_0 gives one copy to node n_5 and

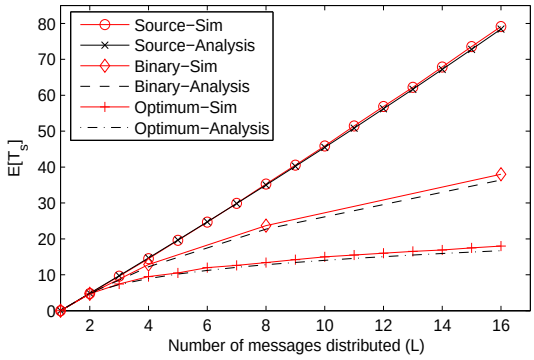


Figure 4.4: $E[T_s]$ values based on analysis and simulation for different message distribution schemes.

n_7, node n_1 gives one copy to n_2 and n_3). In cooperative binary spraying algorithm, node n_2 helps node n_0 in the distribution of copies. When they meet at time 19, node n_2 has just one copy for itself (not eligible for distributing copies to others) and node n_0 has three copies. According to the rule of cooperative binary spraying, they share copy loads equally so that they both have two copies. Then node n_2 and n_0 give one more copy to node n_6 and n_5, respectively. As a result, the total distribution time (T_s) is shortened by the help from other nodes. In optimal spraying, the first five meetings between the nodes having copy and the nodes without copy are (n_0, n_1), (n_1, n_2), (n_1, n_3), (n_3, n_4), (n_0, n_5) in chronological order. Therefore, the possible minimum T_s value is achieved.

We also validated the analysis results obtained above by simulations. We deployed 100 identical nodes with transmission range 10 m on a 300 m by 300 m torus. The nodes move according to the random direction mobility model [44] in which a node moves with random speed for an epoch and then randomly changes its direction and speed. In the simulation, we set the average speed to 8.5 m/s and average epoch duration to 11.5 s. With these settings the average inter-meeting time is $EM = 480$. Figure 4.4 shows the comparison of analysis and simulation results of $E[T_s]$ values for different message spraying algorithms with different number of message counts. Clearly, there is a good match between simulation and analysis results. In Figure 4.5, we compare $E[T_s]$ values obtained in the simulations when binary, cooperative binary and optimum spraying approaches are used. The graph confirms

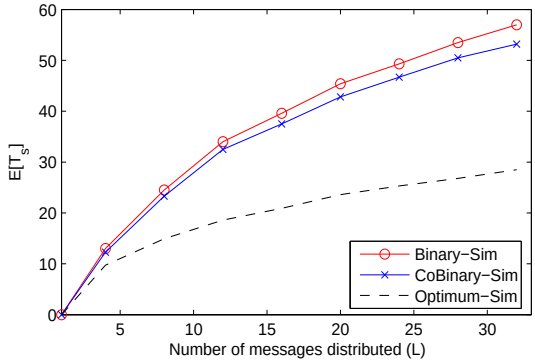


Figure 4.5: $E[T_s]$ from simulations of binary, cooperative binary and optimum spraying algorithms.

the advantage of cooperative binary spraying over the original binary spraying. It gets more pronounced as the number of messages to distribute increases.

4.3 Reducing Cost in Single Period

As shown in the previous section, using different message distribution schemes incurs different costs, yields different spraying times and impacts the delivery time differently. For example, in Figure 4.1, if R in erasure coding algorithm is equal to L in replication based routing and the source spraying is used, the cost is same for both algorithms. Yet, as shown in Figure 4.2, erasure coding reaches high delivery rate earlier than replication based routing. Moreover, we observe in the same figure that by using binary spraying in erasure coding routing, we can shorten T_s and increase the delivery probability even in early times. Unfortunately, using binary spraying increases the cost of erasure coding routing. Therefore using binary spraying, we are trading delivery probability for the cost. In an erasure coding routing with parameters k and R, if the source spraying is used, the cost is MR bytes. The same cost is also paid in binary spraying of erasure coded messages with R' satisfying $R' \log(kR') \leq 2R$. Clearly, R' must be smaller than R. But using kR' messages even with quick message distribution with binary spraying cannot provide the same delivery rate as is achieved by distributing more messages (kR) with source spraying. As a result, a careful selection of message distribution scheme needs to be made to

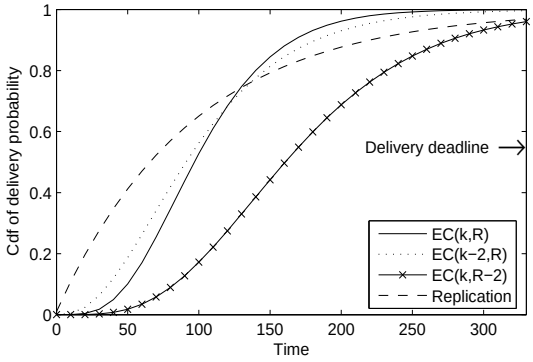


Figure 4.6: Comparison of delivery probabilities in erasure coding based routing with different parameters where (k,R)=(4,5)

reduce the cost.

Moreover, when source spraying is used, the cost can be reduced by selecting right parameters. Consider the graphs in Figure 4.6. They show the cdf of delivery probabilities of three erasure coding based algorithms with different parameters and one replication based routing with $L = R$. For example, if t_d for messages is 330 time units and d_r is 98%, we can achieve the d_r at t_d in all of these four algorithms. However, among these four algorithms, the cost is same ($O(MR)$) except $EC(k, R-2)$ (cost in this case is $M(R-2)$). It should be noted that changing k does not change the cost, it only affects the slope of the curve, and therefore the delivery rate.

Let $p(x)$ denote the cdf of a single node's meeting with the destination at time x after it is given an encoded message. To simplify analysis, we assume that the relay nodes in the network get encoded messages around the same times (at least if measured using t_d as a unit). This assumption is justified if the total number of encoded packets to distribute is not too large and binary spraying is used for message distribution (this is often the case in practice). Then, the probability that there are already k messages gathered at the destination node at time x becomes:

$$P(x, \Phi) \;=\; \sum_{i=k}^{\Phi} \binom{\Phi}{i} p(x)^i (1 - p(x))^{\Phi - i}$$

It should be noted that the erasure coding based routing reduces to the replication

based routing when $k = 1$.

Assuming that t_d and the desired delivery rate (d_r) at t_d are given, we can determine the optimum parameters minimizing the cost while achieving d_r at t_d using the following relation:

$$(k, R) = \min\{\tau | P(t_d, \Phi) \geq d_r\}$$

The value of k can change from 1 to infinity, in theory. However, when this value is large, many small blocks are created (in some cases exceeding the total number of nodes in the network) incurring high processing cost and low bandwidth utilization. Therefore, we assume an upper bound (k_{max}) for k. Once, we know $p(x)$, k_{max}, d_r and t_d, we can find the parameters that minimize the cost of spraying using the above inequality by enumeration of all possible k's.

4.4 Reducing Cost in Multiple Periods

The cost of erasure coding algorithms can further be reduced with the multi-period approach that we used in previous chapter. Instead of distributing all messages at the beginning (single period), we spray only some of them at time 0 and wait for the delivery of sufficient number of messages at the destination. If the delivery has not happened yet until a certain time (x_d), we distribute more messages to the network so that we increase the probability of delivery. The question is, *'can we reduce the average cost while maintaining d_r at t_d?'*

In Figure 4.7, we illustrate the goal we want to achieve in two periods with plot $EC(k, R^*, \alpha)$. Assume that in single period case, the optimum parameters are k and R_{opt}. In erasure coding routing with two periods, source node generates $\Phi_2 = kR^*$ (complexity of encoding is linear) encoded messages at the beginning and allows the distribution of only $\Phi_1 = \alpha kR^*$ of them ($0 < \alpha < 1$) in the first period. Then with the beginning of second period (after time x_d), the remaining messages are distributed so that the probability of gathering of k messages at the destination is increased.

In the first period of two period erasure coding routing, the cdf is P(x,Φ_1).

But in the second period, we need to combine the independent delivery probabilities of first period messages and second period messages which are distributed to the network with a delay of x_d time units. The delivery probability in the second period at time x is:

$$P(x,\Phi_1,\Phi_2) = \sum_{i=k}^{\Phi_2}\left(\sum_{j=l_1}^{l_2} P'(x,j,\Phi_1)P'(x\text{-}x_d,i\text{-}j,\Phi_2\text{-}\Phi_1)\right)$$
$$\text{where } P'(x,j,\Phi_1) = \binom{\Phi_1}{j}p(x)^j(1-p(x))^{\Phi_1-j}$$
$$l_1 = \min\{i,\Phi_1\} \text{ and } l_2 = \max\{0,i-\Phi_2+\Phi_1\}$$

The goal is, for a given Φ, to find a (Φ_1,Φ_2) pair that lowers the average cost while maintaining the d_r by t_d. First of all, to be able to catch the delivery rate of single period, the following inequalities must be satisfied:

$$\begin{aligned} R^* &> R_{opt} \\ P(t_d,\Phi_1,\Phi_2) &\geq P(t_d,\Phi) \end{aligned}$$

Moreover, to lower the average cost compared to the cost in a single period, the following inequalities must be satisfied as well:

$$\begin{aligned} P(x_d,\Phi_1)\Phi_1 + (1-P(x_d,\Phi_1))\Phi_2 &\leq \Phi \\ \frac{\Phi_2-\Phi}{\Phi_2-\Phi_1} &\leq P(x_d,\Phi) \end{aligned}$$

In the design of a two period erasure coding routing, the selection of the message distribution scheme is also important. Since we need a fast distribution of the first period messages, at first glance it seems better to use binary spraying in the first period. However, binary spraying has a bigger cost than source spraying. Therefore, it is better to use source spraying. Similar conclusion is also valid for the second period. Moreover, using source spraying in the second period has also an extra benefit in terms of transmission cost. As soon as the second period starts at time x_d, source node starts distributing remaining messages at a slower rate than binary spraying. With each message distributed to a relay node, the probability of gathering of k encoded packets at the deadline increases. In the mean time,

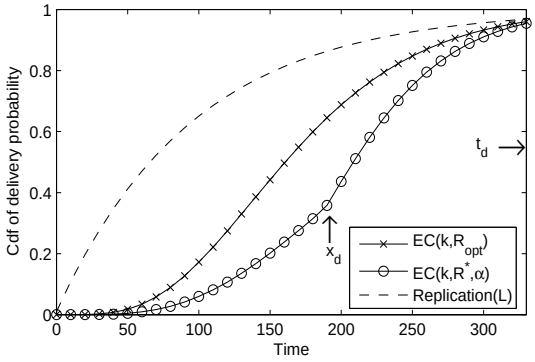


Figure 4.7: Cumulative distribution function of delivery probability in two period erasure coding routing.

if the message is delivered (k encoded packets have arrived destination) and the source node receives acknowledgment of this delivery, then the source node stops distributing remaining messages, so the average cost can further be reduced.

4.5 Simulation Model and Results

We used the same Java based simulator (details of which are given in previous chapter) to evaluate the performance of proposed cost reduction schemes. We randomly deployed 100 mobile identical nodes (including the sink) on a 300 m $\times$ 300 m torus. The nodes move according to random direction mobility model[12] [44]. Each node selects a random direction ([0, 2π]) and a random speed in the range of [4, 13]m/s then goes in that direction during a randomly selected epoch of duration in the range of [8, 15]s. When the epoch ends, the same process runs again and new direction, speed and epoch duration are selected. The transmission range of the nodes is set at $R = 10$ m, yielding $EM = 480$.

Since our goal is to reduce cost, we modeled the simulation environment in such a way as to eliminate the effects of the other parameters. We assume that the buffer space in each node is large enough to avoid any buffer overflows. We also assume a high bandwidth that allows the transmission of all messages during each

[12]Although we used only random direction mobility model in our simulations here, other mobility models such as random waypoint and random walk can be used yielding similar results. Also, with community based models, the advantage of cooperative binary spraying can be more pronounced since the node inter-meeting times are unequal.

node meeting. All messages are assumed to have an average size of $M = 100$ Kbytes. Each message is generated at a randomly selected source node and then addressed to the sink node whose initial location is also chosen randomly. After all the messages are distributed, the destination waits until it receives sufficient number of messages. It is also important to remark that if one of the nodes carrying a message (or messages) meets the destination during the spraying period, it transfers all of its messages to the destination. Therefore, some messages can be directly transferred to the destination without being given to relay nodes, thus yielding a saving in total message transfer cost. For the simulations here, we set $d_r = 99\%$, which is a reasonable delivery rate for real scenarios. The presented simulation results are averaged over 1000 different runs.

We first present results regarding the right spraying algorithm selection. Figure 4.8 shows the average costs per message achieved with different deadlines when source spraying and binary spraying are used[13]. In both algorithms, assuming that the same k value is used, we first found R values achieving the same delivery rate by the deadline (Note that erasure coding algorithm with binary spraying (EC-BS) uses a smaller R value than it is used in the erasure coding algorithm with source spraying (EC-SS) to achieve the same delivery rate by the deadline) and computed the cost when these R values are used. As the results show, EC-BS generates higher cost than EC-SS even though it uses smaller R value. This result demonstrates again the advantage of source spraying over binary spraying (when the contents of distributed messages are different as in erasure coding based routing).

Next, we look at the performance of multi-period spraying approach in erasure coding based routing. Since, the contents of erasure coded messages are different than each other, we use source spraying while distributing messages in multi-period approach. Also, we present results using two different types of acknowledgments for delivered messages (see previous chapter for details of acknowledgment mechanisms).

Table 6.1 shows the minimum costs incurred by *EC-1p* (suffix "xp" denotes

[13]We did not give results for cooperative binary spraying because the superiority of cooperative binary spraying over binary spraying does not provide benefits in random direction mobility model. However, in our future work we will do simulations with heterogeneous mobility models where we believe that the benefit of cooperative binary spraying will be more pronounced.

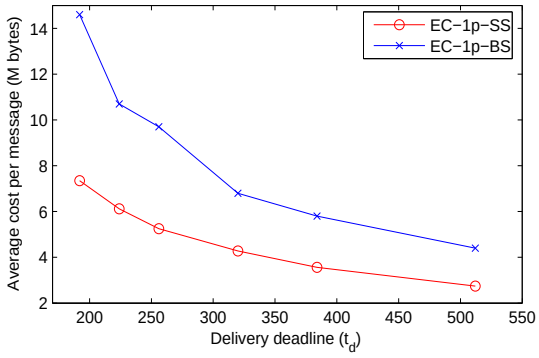


Figure 4.8: Average costs incurred by the single period erasure coding routing when the source and binary sprayings are used in message distribution.

x-period version of EC routing) and *EC-2p* algorithms with the aforementioned two different types of acknowledgments. In both algorithms, we found the optimal parameters which provide minimum average costs using the formula from the previous section. In *EC-2p* algorithm, we used $k_{max} = 5$.

First of all, even though we did not show it here for the sake of brevity, in both algorithms the desired delivery rate is achieved by the given deadlines. Yet, their costs are different. For all t_d values shown, as it is expected, the cost of an algorithm when Type I acknowledgment is used is higher than the cost of the same algorithm when Type II acknowledgment is used. This is simply because of the extra time needed in Type I acknowledgment to inform the source node about the delivery with epidemic like acknowledgment. During this extra time, the source node will still continue distributing the remaining encoded messages it has, increasing the algorithm cost. Moreover, for almost all t_d values, the cost of *EC-2p* algorithm is smaller than the cost of *EC-1p* regardless of the type of acknowledgment used. This clearly shows the superiority of *EC-2p* over *EC-1p* algorithm. We also observe that as the deadline gets tight (decreases), the improvement achieved (percentage of reduced cost with respect to *EC-1p* algorithm) by *EC-2p* algorithm decreases. This is because with shorter deadline, more encoded blocks are generated. Hence, the time to distribute all encoded blocks and also the time needed to inform source node in Type I acknowledgment increases. Consequently, in some cases (t_d=200s),

the cost of *EC-2p* algorithm becomes higher than the cost of *EC-1p* algorithm. Still, in most of the cases, *EC-2p* performs better than *EC-1p* algorithm does. It should also be noted that for the same t_d values, the cost difference between Type I and Type II acknowledgments in *EC-2p* is larger than it is in *EC-1p* algorithm. This is because in Type I acknowledgment, more blocks must report reaching destination in *EC-2p* algorithm than in *EC-1p* algorithm.

t_d	Cost of EC-1p-SS (Kbytes)			Cost of EC-2p-SS(Kbytes)		
sec	Opt(R,k)	Type I	Type II	Opt(R^*,α, x_d)	Type I	Type II
600	(3,2)	343	342	(5, 0.4, 410)	323	319
500	(3,3)	357	356	(5, 0.4, 345)	299	295
400	(4,3)	445	443	(6, 0.5, 270)	412	398
300	(5,3)	536	532	(7, 0.5, 200)	515	495
250	(5,5)	525	517	(8, 0.5, 185)	538	510

Table 4.3: Minimum average costs of single and two period erasure coding algorithms at the time the message is delivered (Type II) and after all nodes receive acknowledgment of the delivery (Type I).

To see the effect of delivery rates on the average cost, we also simulated the algorithms with different d_r values. We selected higher d_r values ($\geq 75\%$) since it is a reasonable design concern of real delay tolerant networks. Figure 4.9 shows the average costs obtained in single period replication based routing and single period erasure coding based routing algorithms when source spraying is used. In these simulations we set $t_d = 256$. The graph demonstrates that with low delivery rates, *Rep-1p* algorithm performs well. However, for higher delivery rates, *EC-1p* is better than *Rep-1p*. This also confirms the ability of erasure coding based routing algorithms to improve the delivery rate at later times (better worst case delivery).

4.6 Summary of Contributions

In this chapter, we studied the erasure coding based routing problem in DTNs. Assuming that we are given a desired delivery rate by the given deadline, we discussed the most cost efficient ways of achieving this goal. We first analyzed different message distribution schemes and discussed their effects on message delivery ratio

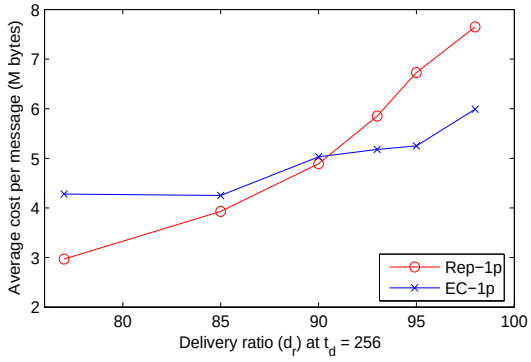


Figure 4.9: Comparison of average costs per message achieved in single period replication based routing and erasure coding based routing with different delivery rates. In both algorithms source spraying is used in message distribution.

and cost. We demonstrated that even though source spraying is the slowest message distribution scheme among others, it is the best strategy to be used in erasure coding based routing since it provides the lowest cost. We also applied multi-period idea to erasure coding based routing and showed that we can decrease the cost even further than the cost of single period erasure coding based routing while still achieving the desired delivery rate by the deadline.

CHAPTER 5
EFFICIENT SINGLE-COPY BASED ROUTING WITH CORRELATED NODE MOBILITY

Delay tolerant networks consist of mobile agents that contact intermittently. However, the intermittent connectivity rate between different pairs of nodes in a network may be different due to the heterogeneity in real networks. Therefore, recent studies on DTN routing have focused on the analysis of real mobility traces (human [62, 63], vehicular [20] etc.) and utilized extracted characteristics of the mobile objects in the design of forwarding algorithms for DTNs. Reviewing these designs and analysis, we have made the following observations:

- The pairwise intermeeting times between nodes usually follow a log-normal distribution [59, 45]. Hence they are not memoryless[14]. This makes future contacts of nodes dependent on their past.

- Most of the previous routing protocols that make the forwarding decisions at the time of node meetings use a metric (encounter frequency [25], time elapsed since last encounter [64]-[66], social similarity [68, 48] etc.) computed from the pairwise relations of nodes with the destination node only. They do not use any information about their *meetings with each other* while computing their delivery metrics and meetings of nodes are usually assumed independent from each other.

- The mobility of many real objects are non-deterministic but periodic [32]. For example, in a cyclic MobiSpace [81], if two nodes were frequently in contact at a particular time in previous cycles, then they are likely to meet around the same time in the next cycle.

The above three observations motivated us to do the research on the correlation between the node meetings for designing more efficient routing algorithms. Hence,

[14] Assume that X is the random variable representing the intermeeting time between two nodes, then the meetings are memoryless if $P(X > s + t \mid X > t) = P(X > s)$ for $s, t > 0$ holds.

we introduce a new link metric, *conditional intermeeting time*, that computes the average intermeeting time between two nodes relative to a meeting with a third node using only local knowledge of the past contacts. For example, let $\tau_A(D|B)$ denote the conditional intermeeting time of node A with destination D given the condition that it has just met B. We define $\tau_A(D|B)$ as the average time it took in the past (computed from previous contact history) to meet with D after its meeting with B. Hence, at each meeting of node A with a potential destination node D, we compute the meeting frequency of A with D conditioned on meetings with all the other nodes A met since its last meeting with node D. Throughout this chapter, we discuss how this metric addresses all the above three observations.

The definition of intermeeting times has natural interpretation in the context of message routing during which messages are received from a node and given to another node on the way towards the destination. Hence, conditional intermeeting time refers to the message holding duration on a given node during the message routing.

We analyze the conditional intermeeting time and discuss when and why it is beneficial in providing accurate information to nodes making routing decisions. We also present statistical results from three different data sets (RollerNet [45], Cambridge [86], Haggle [89]) which contain the contact traces of real objects logged during real life experiments.

We then propose modifications to the existing DTN routing protocols using the proposed metric and demonstrate how their performance improves as the result. Firstly, for the algorithms which route messages over shortest paths (SP) [82, 83], we propose to use conditional intermeeting times rather than standard intermeeting times and route the messages over conditional shortest paths (CSP) [72]. Secondly, for the algorithms which make message forwarding decisions depending on a delivery metric (we call them metric-based forwarding algorithms), we propose to use conditional intermeeting time as a secondary delivery metric and allow the forwarding of messages if and only if both the algorithm's original delivery metric and conditional intermeeting time agree to forward the message to the encountered node [73].

Through simulations based on three different real DTN traces and synthetic traces formed using community based mobility model of nodes, we compare the modified protocols with the original ones. The results show that modifications improve performances of those protocols remarkably. This shows that the conditional intermeeting time represents inter-node link costs more accurately (in the context of routing), hence guiding forwarding decisions effectively while routing a message.

Shortly, the contributions of this chapter are four-fold:

1. We introduce a new metric, conditional intermeeting time, which computes the average intermeeting time between two nodes relative to a meeting with a third node using only the local knowledge of the past contacts.

2. We analyze the proposed metric and show its significance for routing.

3. We improve the existing DTN routing protocols by exploiting conditional intermeeting time.

4. Finally, we perform extensive simulations based on both real and synthetic DTN traces and measure the performance improvement achieved in the modified versions of the existing algorithms.

5.1 Conditional Intermeeting Time

Recently, the research community working on routing algorithms in DTNs has focused on the analysis of real mobility traces to understand the main characteristics of mobile objects. Several experiments in different environments (office [59], conference [89], city [86], skating tour [45]) with different objects (human [62], bus [20], zebra [88]) and with variable number of attendants were performed. From the analysis of the data sets collected during these experiments, significant results about the aggregate and pairwise mobility characteristics of real objects were obtained.

As opposed to the random mobility models [44] according to which the intermeeting times between pairs follow an exponential distribution and are considered independent from each other, recent analysis [59, 45] on real mobility traces showed that the intermeeting times between most (more than 99% in many datasets) of

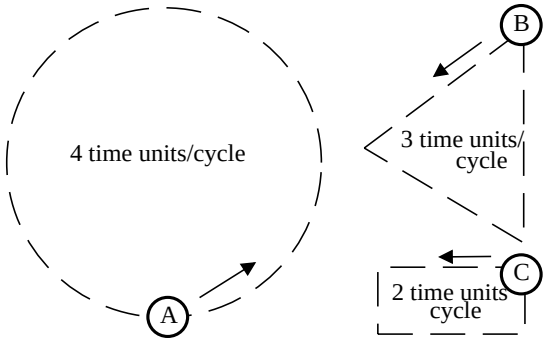


Figure 5.1: A physical cyclic MobiSpace with a common motion cycle of 12 time units.

the node pairs fit to log-normal distribution. This revokes the validity of a common assumption [47, 66, 51] that the pairwise intermeeting times are exponentially distributed and memoryless, and it makes the pairwise contacts between nodes dependent on their pasts. Consequently, the residual time until the next meeting of two nodes can be predicted more accurately if we know that they have not met during the last t time units [59]. Such prediction can be further improved if we consider that meetings of a node with other nodes are not independent from each other (i.e. meetings are correlated). Hence, the nodes can benefit from their entire history of past contacts to predict their future contacts more accurately.

Previous works have used pairwise node relations for guiding forwarding decisions in the DTN routing protocols. The more frequently a node meets with destination, the higher the probability that it can deliver a message to the destination. Therefore, after the meeting of two nodes, the node with smaller average intermeeting time with destination should carry the message. Thus, each node involved in routing, holds the message from the time the message was created or passed at the meeting with a node, to the time of the meeting with another node that received this message. That is, the duration that a non-source node holds a message is between this node's contacts with two other nodes.

Considering the above facts together, we introduce a new metric called *conditional intermeeting time* to define the node relations more precisely within the

context of routing. This metric computes the average intermeeting time between two nodes relative to a meeting with a third node using only the local knowledge of the past contacts.

Algorithm 5 update (node m, time t)

```
1: if m is seen first time then
2:    firstTimeAt[m] ← t
3: else
4:    increment β_m by 1
5:    lastTimeAt[m] ← t
6: end if
7: for each neighbor j ∈ N and j ≠ m do
8:    start a timer t_mj
9: end for
10: for each neighbor j ∈ N and j ≠ m do
11:    for each timer t_jm running do
12:       S[j][m] += time on t_jm
13:       increment C[j][m] by 1
14:    end for
15:    delete all timers t_jm
16: end for
17: for each neighbor i ∈ N do
18:    for each neighbor j ∈ N and j ≠ i do
19:       if S[j][i] ≠ 0 then
20:          τ_s(i|j) ← S[j][i] / C[j][i]
21:       end if
22:    end for
23:    τ_s(i) ← (lastTimeAt[i] − firstTimeAt[i] ) / β_i
24: end for
```

The proposed metric can also provide more accuracy if the nodes move in a cyclic MobiSpace [32, 81]. According to the definition of a cyclic MobiSpace, if two nodes contact frequently at a particular time in previous cycles, the probability that they will be in contact around the same time in the next cycle is high. In Figure 5.1, a sample cyclic MobiSpace with three objects is illustrated. The common motion cycle is 12 time units. In this example, the discrete probabilistic contacts between A and B happen every 12 time units (1, 13, 25 ...) and the discrete probabilistic contacts between B and C occur every 6 time units (2, 8, 14 ...). When we consider the intermeeting time between nodes B and C, we can expect that node B will

forward its message to node C in 6 time units, however conditional intermeeting time of B with C based on the condition that it has met (received the message from) node A lets us know that the message will be forwarded to node C within 1 time unit.

Each node in a DTN can compute the average of its standard and conditional intermeeting times with other nodes using its contact history. In Algorithm 6, we show how a node, s, can compute these metrics from its previous meetings. The notations we use in this algorithm (and also throughout the chapter) are listed below with their meanings:

- $\tau_A(B|C)$: Average time it takes for node A to meet node B after it meets node C. If B=C, the notation (in short $\tau_A(B)$) shows the standard intermeeting time between nodes A and B.
- S: $N \times N$ matrix where $S(i, j)$ shows the sum of all samples of conditional intermeeting times with node j relative to the meeting with node i. Here, N is the neighbor count of current node (i.e. $N(s)$ for node s).
- C: $N \times N$ matrix where $C(i, j)$ shows the total number of conditional intermeeting time samples with node j relative to its meeting with node i.
- β_i: Total meeting count with node i.

To find the conditional intermeeting time of $\tau_A(B|C)$, each time node A meets node C, it starts a different timer. When it meets node B, it sums up the values of these timers and divides the results by the number of active timers before deleting them. This computation is repeated again each time node C is encountered. Then, the total of times collected from each timer is divided by the total count of timers used, to find the value of $\tau_A(B|C)$. We can also use a sliding window with an appropriate size over the past contacts [83] and take into account the edge effects [20] to make the computation more local and updated. Moreover, we do not consider the contact durations in these computations since inter-contact times are usually much longer than contact times in real DTNs. However, if the last assumption does not hold, it is easy to modify all computations accordingly.

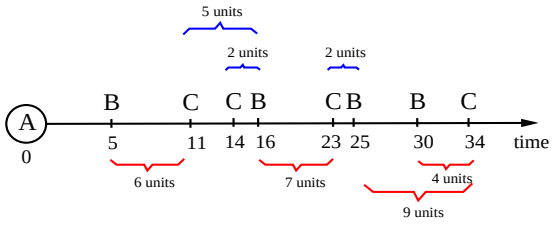


Figure 5.2: Example meeting times of node A with nodes B and C. Upper and lower values are used to compute $\tau_A(B|C)$ and $\tau_A(C|B)$, respectively.

Consider the sample meeting times of a node A with its neighbors B and C in Figure 5.2. Node A meets with node B at times $\{5, 16, 25, 30\}$ and with node C at times $\{11, 14, 23, 34\}$. Following the procedure described above, we find that $\tau_A(B|C) = (5+2+2)/3 = 3$ time units and $\tau_A(C|B) = (6+7+4+9)/4 = 6.5$ time units while $\tau_A(B) = 8.33$ time units and $\tau_A(C) = 7.67$ time units.

5.2 Analysis of Conditional Intermeeting Time

In this section, we give the analysis of conditional intermeeting time and show why it is significant in accurate prediction of future meetings within the context of routing.

Assume that node A has two different contacts, B and C, and the sets S_B and S_C include the meeting times of node A with nodes B and C during the time frame $[0, T]$, respectively.

$$\begin{aligned} S_B &= \{B_1, B_2, B_3, \ldots B_n\}, \text{n meetings} \\ S_C &= \{C_1, C_2, C_3, \ldots C_m\}, \text{m meetings} \end{aligned}$$

Furthermore, assume that the intermeeting time of node A with node B is well represented by a random variable X_1 with the cdf $D_1(x)$ and probability density $d_1(x) = D_1'(x)$, whereas the intermeeting time of node A with C is well represented by a random variable X_2 with the cdf $D_2(x)$ and probability density $d_2(x) = D_2'(x)$.

To find $\tau_A(C|B)$, we need to compute the following (without loss of generality,

we assume that $C_m \geq B_n$):

$$\tau_A(C|B) = \frac{\sum_{k=1}^{n}(C_{d(k)} - B_k)}{n},$$

where, $d(k) = \min\{i : C_i \geq B_k\}$.

The conditional intermeeting time of node A with node C under the condition that it has met node B is then a random variable that we will denote as Y. Then $\tau_A(C|B) = E[Y]$. Let's consider j^{th} meeting of node A with node B and denote $t = B_j - C_{d(j)-1}$. Consider a family of random variables $Y(t)$'s with cdfs $D_t(x)$ and probability densities $d_t(x)$. $Y(t)$ represents the distribution of the remaining time for A to meet C given that t time units has passed since the last meeting with C (Note that, t is defined by meeting with node B). Then:

$$D_t(y) = \frac{D_2(y+t) - D_2(t)}{1 - D_2(t)}$$
$$d_t(y) = \frac{d_2(y+t)}{1 - D_2(t)}$$

$$E[Y(t)] = \frac{\int_{y=0}^{\infty} y d_2(y+t)dy}{1 - D_2(t)}$$

Then, using $[z(1 - D(z))]' = 1 - D(z) - zD'(z)$:

$$E[Y(t)] = \frac{\int_{z=t}^{\infty}(1 - D_2(z))dz}{1 - D_2(t)}$$

When X_1 and X_2 are exponentially distributed random variables, then $D_t(x) = D_2(x)$, showing the memoryless property of exponential distribution. However, as the previous work suggests, X_1 and X_2 fit well with log-normal distribution. Then:

$$E[Y(t)] = \frac{e^{\mu_2 + \frac{\sigma_2^2}{2}}\left[1 - \text{erf}\left(\frac{\ln t - (\mu_2 + \sigma_2^2)}{\sigma_2\sqrt{2}}\right)\right]}{1 - \text{erf}\left[\frac{\ln t - \mu_2}{\sigma_2\sqrt{2}}\right]} - t$$

where erf is the error function and μ_2 and σ_2 are mean and variance of X_2, respectively. Since in this case, as well in general, the expected value of $Y(t)$ depends

on the value of t. Then, denoting by $d_{ACB}(t)$ the probability density of the time difference between the meeting of node A with C to time of meeting of node A with B before A meets C, we get:

$$E[Y] = \int_{t=0}^{\infty} \frac{\int_{z=t}^{\infty}(1 - D_2(z))dz}{1 - D_2(t)} d_{ACB}(t)dt$$

Clearly, $E[Y]$ depends on probability density function $d_{ACB}(t)$ that is defined by the correlation between the meetings of node A with nodes B and C. This formula analytically expresses the fact that the time it takes to meet C depends on the identity of B ($\tau_A(C|B)$ is not the same for all Bs). Due to the nature of DTNs, and also the possible cyclic behavior of nodes [81], there is often a repeating pattern between the meetings of node A with node B and C, creating a strong correlation. Consequently, $E[Y]$ strongly depends on the aforementioned correlation.

Given A, to see the distribution of $\tau_A(C|B)$ according to different B and C nodes, we computed the average conditional intermeeting time maps (C-Map) of the most popular nodes (having the highest total meeting count with other nodes) in three different datasets. In Figure 5.3, we show these results. For each dataset, the left plot shows the 3D view of $\tau_A(C|B)$ and the right plot shows the contour plot displaying the isolines of $\tau_A(C|B)$. The darker the color in the plots, the smaller the $\tau_A(C|B)$ values. Interestingly, the diagonals and the A^{th} row and column in the plots are the darkest places because $\tau_A(C|B)$ values are assumed zero in these cases. Moreover, if there is no instance of the case in which node A meets node C after it meets node B, we set $\tau_A(C|B) = -1$.

If there were no correlation between the meetings of node A with other nodes, $\tau_A(C|B)$ would have been the same for all B's other than A or C and entire rows would have been of the same color in contour plots. In contrast, we observe different colors within the rows, demonstrating that the meetings of node A with different nodes are not independent of each other. Indeed, this observation is consistent with the real world scenarios. For example, consider the meetings of a man who goes from home to work every morning. After meeting his family members (while leaving home), he meets later his office friends. Yet, on the way to his office, he meets the security guard at the gate of his workplace a few moments before meeting

his office friends. In other words, the meetings of the man with his office friends is correlated with his meetings with security guard.

5.3 Proposed Algorithms

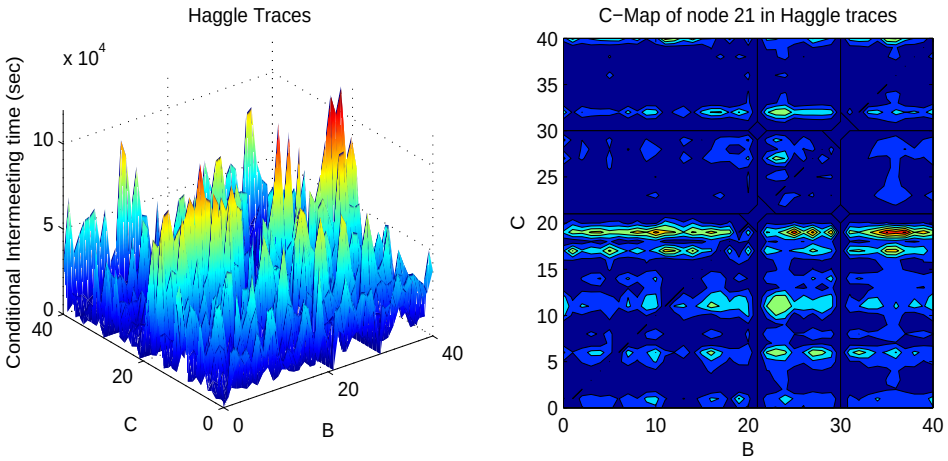
In this section, we present two different applications of conditional intermeeting time to the existing DTN routing algorithms. In the first part, we look into the shortest path based routing algorithms and propose to use conditional shortest paths to route messages. Then, in the second part, we propose to revise message forwarding decisions of metric-based forwarding algorithms by including the conditional intermeeting time.

5.3.1 Shortest Path based Routing

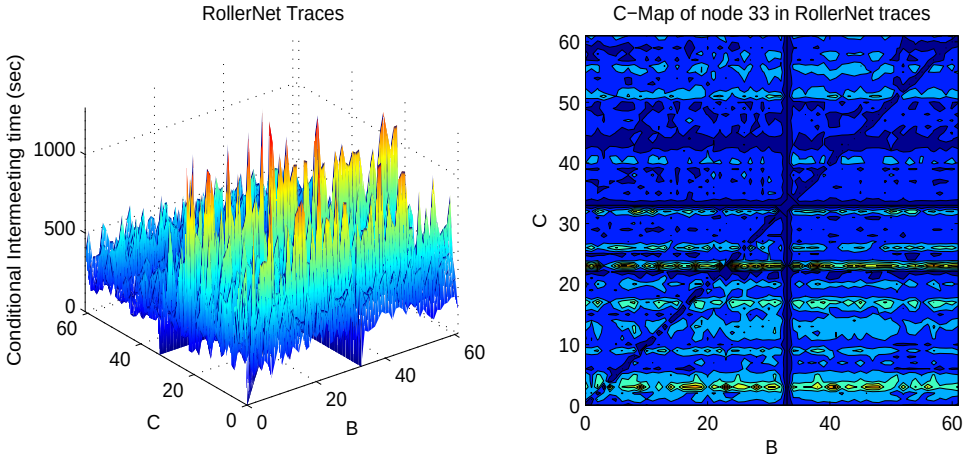
5.3.1.1 Overview

Shortest path routing (SPR) protocols for DTNs are based on the designs of routing protocols for traditional networks. The links between each pair of nodes are assigned a cost and messages are forwarded over the shortest paths between the source and the destination. Furthermore, the dynamic nature of DTNs is also addressed in these designs. Two of the common metrics used to define the link cost are minimum expected delay (MED [82]) and minimum estimated expected delay (MEED [83]). These metrics compute the expected waiting time plus the transmission delay between each pair of nodes. However, while the former uses the future contact schedule, the latter uses only observed contact history.

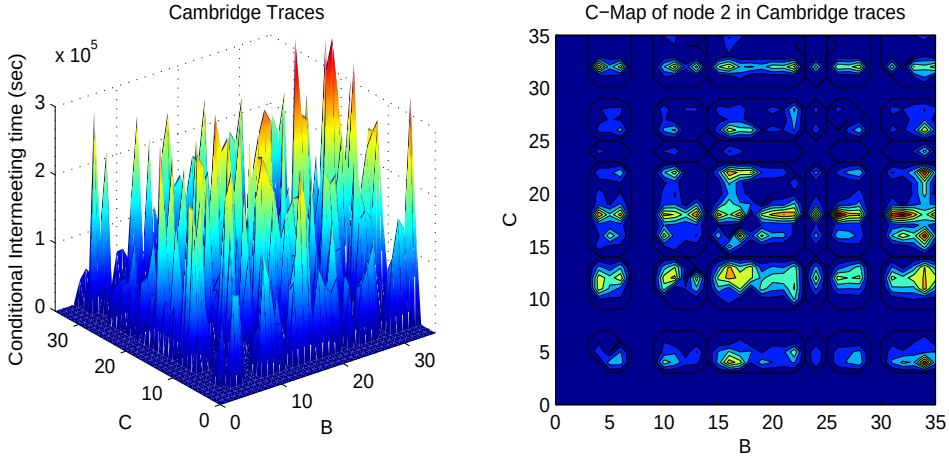
Routing decision can be made at three different points in shortest path based routing: *i*) at source node, *ii*) at each hop, and *iii*) at each contact with other nodes. In the first case (source routing), shortest path of the message is decided at the source node and the message follows that path. In the second one (per-hop routing), when a message arrives at an intermediate node, the node determines the next hop for the message towards the destination and the message waits for that node. Finally, in the third one (per-contact routing), the routing table is recomputed at each contact with other nodes and the forwarding decision is made accordingly. In these algorithms, utilization of recent information increases from the first to the



(a) C-Map of node 21 in Haggle traces



(b) C-Map of node 33 in RollerNet traces



(c) C-Map of node 2 in Cambridge traces

Figure 5.3: C-Maps of popular nodes in three datasets. In figures, B represents the id of the node already met and C represents the id of the node to be met.

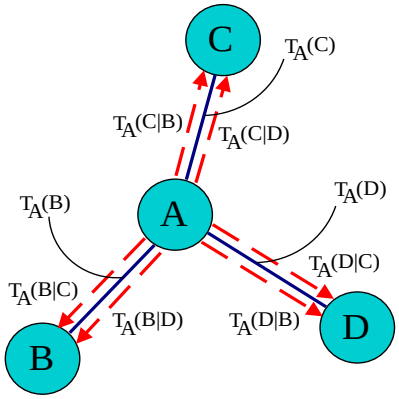


Figure 5.4: A sample DTN graph with four nodes and nine edges.

last one improving the quality of the forwarding decisions; however, more processing resources are used as the routing decision is computed more frequently.

The suitability of SPR algorithms for DTNs and the scalability and complexity of their designs have been already discussed in [82, 83], hence, in this chapter, we focus on the enhancements of the performance of SPR algorithms achieved by utilizing our metric, conditional intermeeting time, rather than using standard intermeeting time. To this end, in the rest of this section, we show the necessary changes to the current designs of SPR algorithms.

5.3.1.2 Network Model

We model a DTN as a graph $G = (V, E)$ where the mobile nodes are represented by vertices (V) and the possible connections between these nodes are represented by the edges ($E = E_u \cup E_b$). Unlike previous DTN graph models, there can be multiple and both unidirectional (E_u) and bidirectional (E_b) edges between the nodes. The neighbors of a node i are denoted by $N(i)$ and the edge sets are given as follows:

$$\begin{aligned} E_b &= \{(i,j) \mid \forall j \in N(i)\} \quad w(i,j) = \tau_i(j) = \tau_j(i) \\ E_u &= \{(i,j) \mid \forall j \neq k \in N(i)\} \quad w(i,j) = \tau_i(j|k) \end{aligned}$$

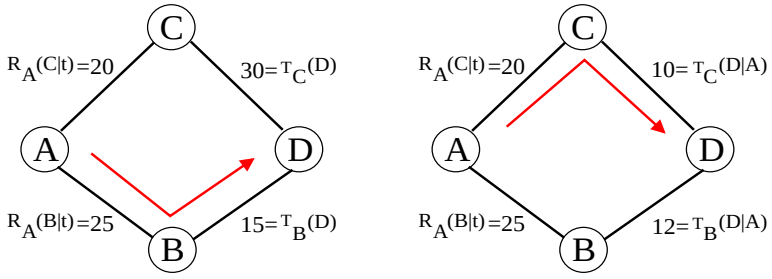


Figure 5.5: An example case where CSP can be different than SP.

There can be multiple unidirectional edges (E_u) between any two nodes but these edges differ from each other in terms of their weights ($w(i,j)$) and the corresponding third node which is the reference point used while computing the conditional intermeeting time. In Figure 5.4, we illustrate a sample DTN graph with four nodes and nine edges. There are three bidirectional edges with weights of standard intermeeting times and six unidirectional edges with weights of conditional intermeeting times. In the graph, there are no unidirectional edges originating from nodes B, C and D because these nodes meet only with a single node, thus there are no conditional intermeeting times associated with them.

5.3.1.3 Conditional Shortest Path Routing

Our algorithm basically finds conditional shortest paths (CSP) for each source-destination pair and routes the messages over these paths. We define the CSP from a node n_0 to a node n_d as follows:

$$CSP(n_0, n_d) = \{n_0, n_1, \ldots, n_{d-1}, n_d \mid \Re_{n_0}(n_1|t) + \sum_{i=1}^{d-1} \tau_{n_i}(n_{i+1}|n_{i-1}) \text{ is minimized.}\}$$

Here, t represents the time that has passed since the last meeting of n_0 with n_1 and $\Re_{n_0}(n_1|t)$ is the expected residual time to the next meeting of n_0 and n_1 given that they have not met in the last t time units. $\Re_{n_0}(n_1|t)$ can be computed as in [59] with parameters of distribution representing the intermeeting time between n_0 and n_1. It can also be computed in a discrete manner from the observed intermeeting times

of n_0 and n_1. Assume that n_0 observed k intermeeting times with n_1 in the past. Let $\tau^1_{n_0}(n_1)$, $\tau^2_{n_0}(n_1)$,...$\tau^k_{n_0}(n_1)$ denote these values. Then, discrete computation of $\Re_{n_0}(n_1|t)$ can be defined formally as follows:

$$\Re_{n_0}(n_1|t) = \frac{\sum_{s=1}^{k} f^s_{n_0}(n_1)}{|\{\tau^s_{n_0}(n_1) \geq t\ \}|} \text{ where,}$$

$$f^s_{n_0}(n_1) = \begin{cases} \tau^s_{n_0}(n_1) - t & \text{if } \tau^s_{n_0}(n_1) \geq t \\ 0 & \text{otherwise} \end{cases}$$

If none of the k observed intermeeting times is bigger than t (this case occurs less likely as the the contact history grows), $\Re_{n_0}(n_1|t)$ becomes 0, which is a good approximation.

Next, we give an example showing that CSP gives the right path with more accurate delay than SP. Consider Figure 5.5 which shows the two different views of the same DTN at node A. While the left graph is formed using the standard network model where the link weights (after the first hop) are intermeeting times, the right one is formed by the network model proposed in previous section. From the left graph, we conclude that SP(A, D) is $<A, B, D>$. Thus, it is expected that on average a message from node A will be delivered to node D in 40 time units. However, this might be an overestimation because the actual delay might be smaller due to the correlation between the meetings of B with A and D. As $w(C, D)$ states in the right graph, node C can have a smaller conditional intermeeting time (than the standard intermeeting time) with node D assuming that it has met node A. In other words, node C can compute its faster transfer capability of messages (received from node A) to node D. Hence, in the right graph, CSP(A, D) is $<A, C, D>$ with the path cost of 30 time units. As a result, node A decides to send a message to node D over C.

Each node forms the DTN using the aforementioned network model and collects the standard and conditional intermeeting times of other nodes via epidemic link state protocol as it is described in the original study [83]. However, once the weights are known, it is not as easy to find CSPs as it is to find SPs. Consider Figure 5.6 where the CSP(A, E) follows path 2 and CSP(A, D) follows $<A, B, D>$.

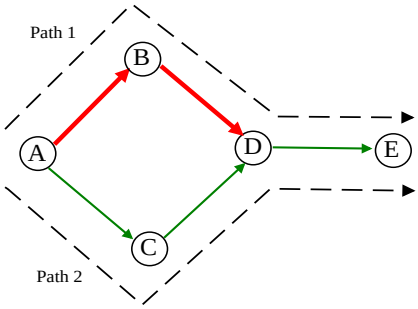


Figure 5.6: Path 2 may have smaller conditional delay than path 1 even though CSP from A to D is through B.

This situation is likely to happen in a DTN, if $\tau_D(E|B) \geq \tau_D(E|C)$ is satisfied. Running Dijkstra's or Bellman-Ford algorithm on the current graph structure cannot detect such cases and concludes that CSP(A, E) is $<A, B, D, E>$. Therefore, to obtain the correct CSPs for each source destination pair, we propose the following transformation on the current graph structure.

Given a graph $G = (V, E)$, we obtain a new graph $G' = (V', E')$ where:

$$
\begin{aligned}
V' &\subseteq V \times V \text{ and } E' \subseteq V' \times V' \text{ where,} \\
V' &= \{(i_j) \mid \forall j \in N(i)\} \text{ and } E' = \{(i_j, k_l) \mid i = l\} \\
&\text{where, } w'(i_j, k_l) = \begin{cases} \tau_i(k|j) & \text{if } j \neq k \\ \tau_i(k) & \text{otherwise} \end{cases}
\end{aligned}
$$

Notice that the edges in E_b (in G) are made directional in G'. Also, the unidirectional edges (E_u) between the same pair of nodes in G are separated in E'. This graph transformation keeps all the historical information that conditional intermeeting times require and also keeps the paths only with a valid history. For example, for path $<A, B, C, D>$ in G, an edge like (C_D, D_A) does not exist in G'. Hence, only the correct τ values will be added to the path calculation. To solve the conditional shortest path problem however, we add one vertex for source S (apart from its permutations) and one vertex for destination node D. We also add outgoing edges from S to each vertex $(i_S) \in V'$ with weight $\Re_S(i|t)$. Furthermore, for the destination node, D, we only add incoming edges from each vertex $i_j \in V'$ with weight $\tau_i(D|j)$

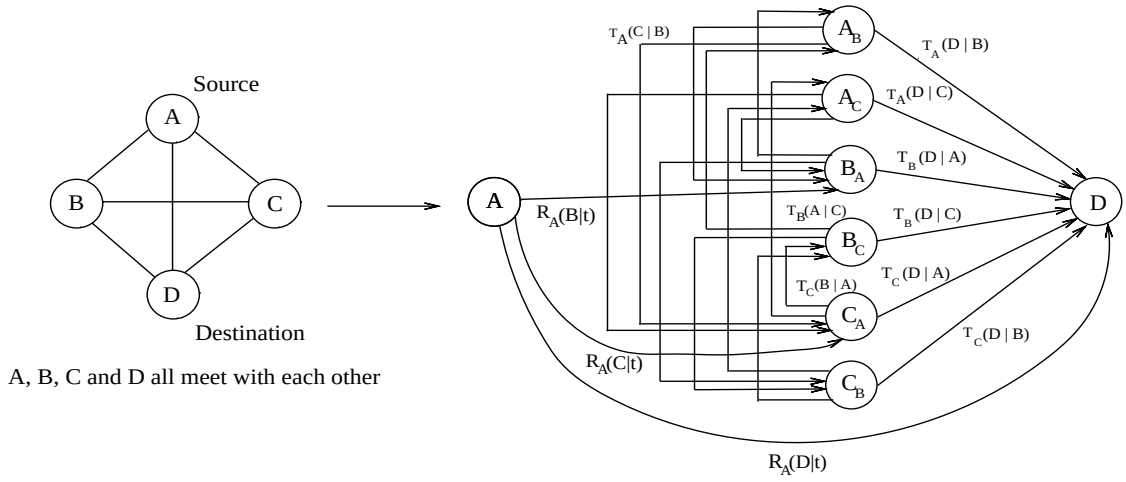


Figure 5.7: Graph transformation to solve CSP with 4 nodes where A is the source and D is the destination.

and from S with weight $\Re_S(D|t)$.

In Figure 5.7, we show a sample transformation of a clique of four nodes to the new graph structure. In the initial graph, all mobile nodes A to D meet with each other, and we set the source node to A and destination node to D (we did not show the directional edges in the original graph for brevity). Note that we set any path to begin with A on transformed graph G', but we also put the permutations of A, B and C with each other.

Running Dijkstra's shortest path algorithm on G' given the source node S and destination node D will give the shortest conditional path. In G', $|V'| = O(|V|^2)$ and $|E'| = O(|V^3|) = |E|^{3/2}$, and therefore Dijkstra's algorithm will run in $O(|V|^3)$ (with Fibonacci heaps) while computing the original shortest paths (with standard intermeeting times) takes $O(|V|^2)$.

Using conditional intermeeting times instead of standard intermeeting times only requires (over original design) extra space to store the conditional intermeeting times and additional processing, as complexity of running Dijkstra's algorithm increases from $O(|V|^2)$ to $O(|V|^3)$. We believe that in current DTNs, wireless devices have enough storage and processing power not to be unduly taxed with such an increase. Moreover, to lessen the burden of collecting and storing link weights, an

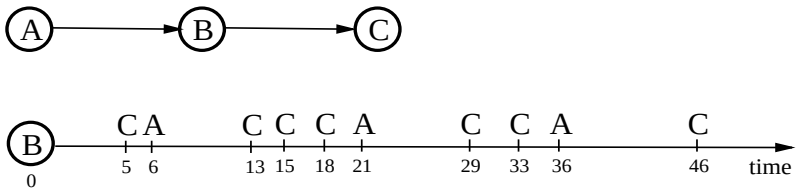


Figure 5.8: A sample case showing how the conditional intermeeting time can be bigger than standard intermeeting time. Here, while $\tau_B(C) = 6.83$, $\tau_B(C|A) = 8.33$. This makes $|\mathbf{CSP}(A,C)| > |\mathbf{SP}(A,C)|$.

asynchronous and distributed version of the Bellman-Ford algorithm can be used, as described in [58].

For conditional shortest paths, an interesting case may arise. For some source-destination pairs, the length of CSP can be longer than the length of SP if the conditional intermeeting time values between the nodes on the path are bigger than the standard intermeeting time values. For instance, consider Figure 5.8 where the meetings of a node B with its neighbor nodes A and C are shown. Here, node B concludes that $\tau_B(C) = 6.83$ and $\tau_B(C|A) = 8.33$. Then, node A finds a CSP with longer delay than SP. Although our algorithm may generate such CSPs and this may result in using a path longer than the SP between the two nodes, in reality this does not cause our algorithm to perform worse than the algorithms using SPs. This is because the goal in DTNs is to route messages with minimum delay and a node receiving a message from one of its neighbors relays the message to another neighbor after an average time equal to the conditional intermeeting time of this node with its neighbors. Shortest paths formed by standard intermeeting times between nodes do not consider the arrival times of messages at nodes and underestimate the delay on source-destination path. Therefore, an SP from a source to destination with shorter but inaccurate delay causes the message follow a non-optimal path and increases the delivery delay. Our simulations in the next section confirm this conclusion.

5.3.2 Metric-based Forwarding Algorithms

5.3.2.1 Overview

A common method of routing in DTNs is to forward the message to the encountered node that is more likely to meet with destination than the current message carrier. However, making effective forwarding decisions in single-copy based routing in DTNs is a challenging task. When two nodes meet, one of them forwards a message to the other one if it decides that the message will have a higher chance to be delivered to the destination at the other node.

In previous work, depending on the observed contact history between nodes, several metrics have been used to define the delivery quality of nodes. Some of the popular ones are encounter frequency [25], time elapsed since last encounter [64, 66], residual time [59], social similarity [68, 48] and location similarity [84]. For example, in Prophet [25], messages are forwarded to the nodes that meet with the destination more frequently.

5.3.2.2 Proposed Revision

In most of the previous work, meetings of a node with other nodes are assumed independent from each other and the forwarding decision at the encounter of two nodes is made depending on their individual relations with the destination node. In some algorithms such as [25, 66], with additional processing (i.e. applying transitivity) on pairwise meetings, more accurate metrics are used to reflect the effect of other nodes on the delivery quality of a node. However, these improvements can also be applied to all other metrics, including the one introduced in this chapter. Our contribution is the introduction of a new metric having this property by default in its basic definition.

To make forwarding decisions of these algorithms more effective, thus to improve their performance, we propose to use conditional intermeeting time as an additional delivery metric. That is, when two nodes meet, they will also compare their conditional intermeeting times with destination (depending on the condition that they met each other). If the current carrier of the message learns that other node also has a shorter remaining time (according to conditional intermeeting time)

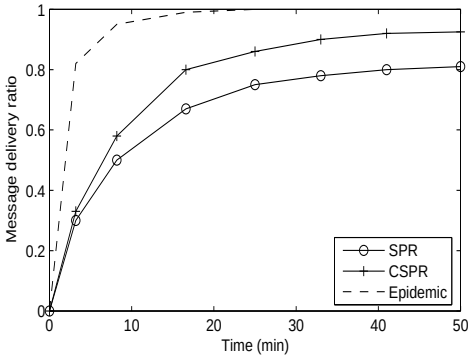
to meet the destination than itself, the message is forwarded. At first glance, this additional condition seems to cut down the number of times the message is forwarded so that the probability of delivery will be reduced. However, as simulation results show, the necessary number of hops are preserved and the less beneficial ones are not performed. Therefore, more effective forwarding decisions are made so that the cost of message delivery declines while the delivery ratio and average delay are maintained (in some cases, even the delivery ratio increases and average delay decreases).

5.4 Performance Evaluation

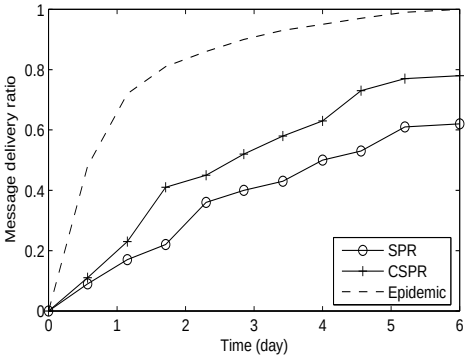
To evaluate the performance of proposed modifications of algorithms, we used our Java based custom DTN simulator. It uses either the traces of real objects from real DTN environments or the traces which are built synthetically. The network parameters (number of nodes etc.) are set according to the traces used.

5.4.1 Algorithms in Comparison

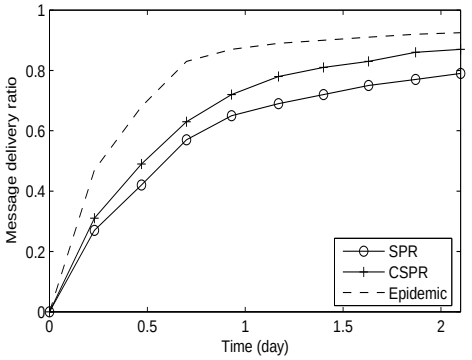
In simulations, we compared existing DTN algorithms with their modified versions that utilize conditional intermeeting time in their designs. In the first part, we compared Shortest Path Routing (SPR) with Conditional Shortest Path Routing (CSPR) which is described in Section 5.3.1.3. Then in the second part, we compared the existing and revised versions of two metric-based DTN routing algorithms: Prophet [25] and Fresh [64]. In Prophet, when two nodes, A and B, meet, A forwards its message to B if and only if its delivery probability to destination D is smaller than B's delivery probability. In Fresh, A forwards the message to B only if B has a more recent meeting with destination D than itself. In the revised versions of these algorithms (we refer to them as C-Prophet and C-Fresh to underline that they use conditional intermeeting time), A forwards the message to B if $\tau_A(D|B) > \tau_B(D|A)$ is also satisfied (in addition to algorithm's own forwarding condition). Although we obtained results (showing performance improvement) with many metric-based algorithms (including [59]), we show only the results of two benchmarking algorithms for brevity. However, we give also the results obtained by



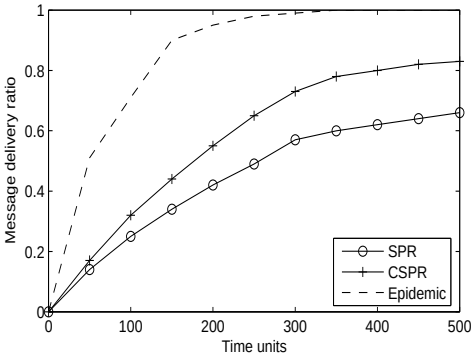
(a) RollerNet Traces



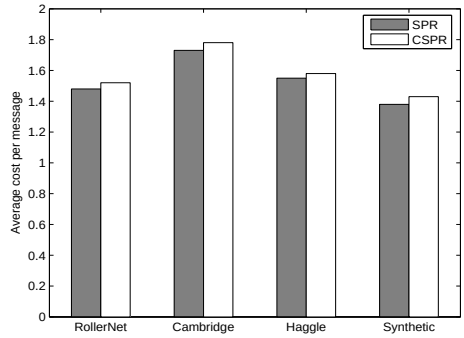
(b) Cambridge Traces



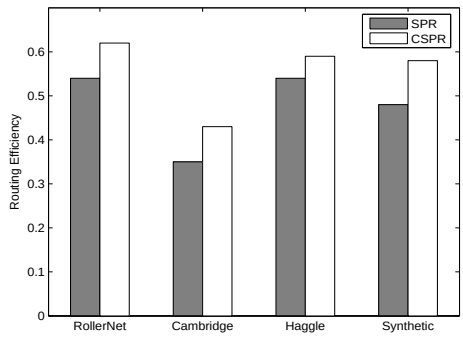
(c) Haggle Project Traces



(d) Synthetic Data



(e) Average Cost



(f) Routing Efficiency

Figure 5.9: Comparison of SPR and CSPR: Message delivery ratio (a-d), Cost (e) and Routing Efficiency (f) vs. time.

Epidemic Routing [22] since it achieves the optimum delivery ratio and delay (at high cost, however).

5.4.2 Data Sets

5.4.2.1 Real DTN Traces

We used the following three real DTN traces from the crawdad archive [87]:

- **RollerNet** [45] traces include the opportunistic sightings of Bluetooth devices distributed to 62 rollerbladers in the 3 hour roller tour of Paris on August 20, 2006.

- **Cambridge Dataset** [86] includes a number of traces from Bluetooth sightings by 36 students from Cambridge University who were asked to carry the iMotes with them at all times for the duration of the experiment that started on October 28, 2005 and ended on December 21, 2005.

- **Haggle Project Dataset** [89] consists of many traces from different experiments. We selected the Bluetooth sightings recorded between the iMotes carried by 41 attendants of Infocom 2005 Conference held in Miami. Devices were distributed on March 7th, 2005 between lunch time and 5pm and collected on March 10th, 2005 in the afternoon.

5.4.2.2 Synthetic Mobility Traces

We also generated synthetic mobility traces using a community-based mobility model which is similar to the models in [25, 44, 94]. In a 1000 units by 1000 units square region, we generated N_c randomly located non-overlapping community regions (home, work, school etc.) of size 100 units by 100 units and distributed N_p nodes (i.e. people) to these community regions. For each node, we randomly assigned V communities to visit (i.e. commonly visited places for a person in a day). Each node first selects a random point within the next community region in its list, assigns a random speed in range $[V_{min}, V_{max}]$ and moves towards the target point with that speed. Once it reaches that point, it randomly assigns a visit duration in range $[T_{min}, T_{max}]$ and randomly walks within the community region for that visit

duration. Once that duration expires, it moves to the next community in its list in a similar way. Each node visits all the communities in its list as indicated, then once all of them are done (i.e. end of day), they again start the same process and start visiting the communities in their list. While nodes are moving, we record the meetings between nodes assuming they have a transmission range of R. The default values for the parameters are N_c=10, N_p=50, V=5, V_{min}=10 units, V_{max}=50 units, T_{min}=20 time units, T_{max}=50 time units. However, we also looked at the effects of different values of parameters in simulations.

5.4.3 Performance Metrics

We use the following three metrics to compare the algorithms: message delivery ratio, average cost, and routing efficiency. Delivery ratio is the proportion of messages that are delivered to their destinations among all messages generated. Average cost is the average number of forwards (hop counts) done per message before delivery. Finally, routing efficiency [90] is defined as the ratio of delivery ratio to the average cost. In the results, we did not give separate plots for delivery delay because they can be obtained from the delivery ratio plots.

5.4.4 Simulation Results

To collect several routing statistics, we have generated traffic on the aforementioned traces. For each simulation run, after a warm up period, we generated 5000 messages from a random source node to a random destination node at each t seconds. In RollerNet, since the duration of experiment is short, we set t = 1s, but for Cambridge and Haggle data sets, we set t = 1min and t = 30s, respectively. For synthetic trace, we set t = 10 time units. Besides this single difference, we compare all algorithms in the same conditions.

For main simulations, we assume that the nodes have enough buffer space to store every message they receive, the bandwidth is high and the contact durations of nodes are long enough to allow the exchange of all messages between nodes[15]. These assumptions are reasonable in view of capabilities of today's technology and

[15]We also performed simulations with limited resources and different values for parameters and present these results later.

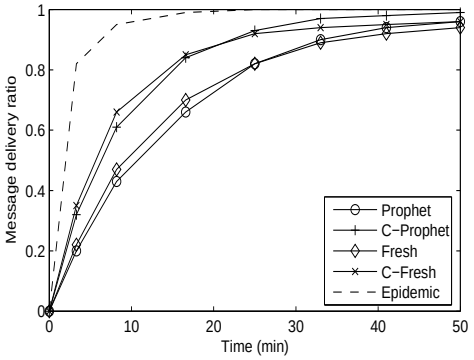
are also used commonly in previous studies [85, 51]. Any change in the current assumptions is expected to affect the performance of compared algorithms in the same way since they use one copy of the message. Moreover, we used a simplified slotted CSMA MAC model as in [47]. We ran each simulation 10 times with different seeds and in each run, we collect statistics by running each algorithm on the same set of messages. All results plotted in figures show the averages of results obtained in all runs.

5.4.4.1 Comparison of CSPR and SPR

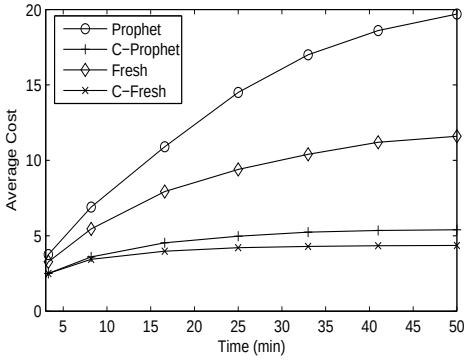
Figure 5.9a shows the delivery ratios achieved in CSPR and SPR algorithms with respect to time (i.e. TTL of messages) in RollerNet traces. Clearly, CSPR algorithm delivers more messages to their destinations than SPR algorithm. Moreover, it achieves lower average delivery delay than SPR algorithm. For example, CSPR delivers 80% of all messages after 17 minutes with an average delay of almost 6 minutes, while SPR achieves the same delivery ratio only after 41 minutes and with an average delay of 12 minutes. Moreover, as it is shown in Figure 5.9e, average costs in SPR and CSPR are very close (1.48 and 1.52 respectively) to each other (and much smaller than the average cost in epidemic routing which is around 25).

We also observe better delivery ratios achieved by CSPR algorithm in Cambridge and Haggle traces in Figure 5.9b and Figure 5.9c, respectively. In Cambridge traces, after 6 days, CSPR delivers 78% of all messages with an average delay of 2.6 days, however SPR can only deliver 62% of all messages to their destination with an average delay of 3.2 days. Moreover, average costs in SPR and CSPR are 1.73 and 1.78 respectively while it is around 16 in epidemic routing. Similarly, in Haggle traces, with an average cost close to each other, CSPR delivers 87% of all messages by the end of simulation whereas SPR can only achieve 78% delivery ratio. The results with synthetic data in Figure 5.9d also support the results based on real traces. While SPR delivers 65% of messages, CSPR delivers 82% of them when TTL of messages is set to 500 time units.

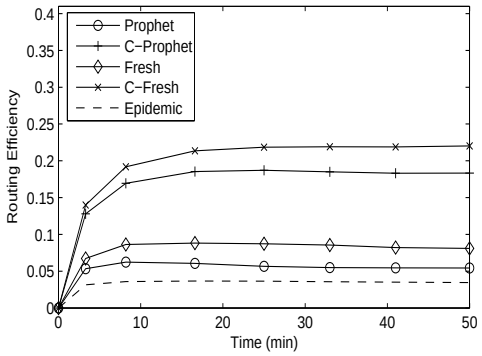
Figure 5.9f compares the routing efficiency of SPR and CSPR in all four traces. It shows an increase of 10%-22% in routing efficiency with the usage of conditional



(a) Message delivery ratio vs. time



(b) Average cost vs. time

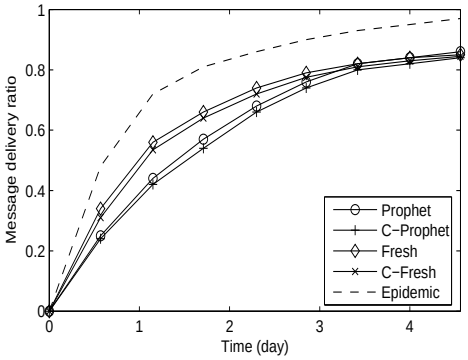


(c) Routing Efficiency vs. time

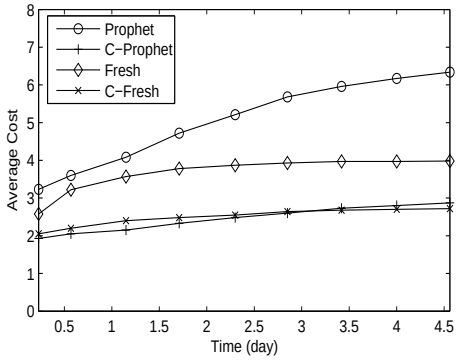
Figure 5.10: Comparison of metric-based forwarding algorithms using RollerNet traces

intermeeting times. However, if we compare the percentage of undelivered messages in these algorithms that are delivered in Epidemic routing, we can observe a higher performance increase. For example in Haggle traces, Epidemic routing delivered 94% of all messages. CSPR lost only 7% of these messages, while SPR lost 16% of them. Hence, CSPR achieved more than 55% improvement over SPR.

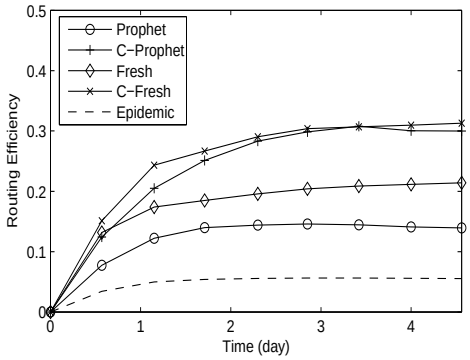
These results show that in the context of routing, conditional intermeeting time provides more accurate link costs than standard intermeeting time. Therefore, in CSPR, more effective paths with similar average hop counts are selected during the routing of a message towards the destination. Thus, higher delivery ratios with lower end-to-end delays are achieved. In SPR and CSPR algorithms here, we used source-routing [82] and let the messages follow the paths which are decided at the source nodes. We also observed similar results in our simulations with other routing



(a) Message delivery ratio vs. time



(b) Average cost vs. time



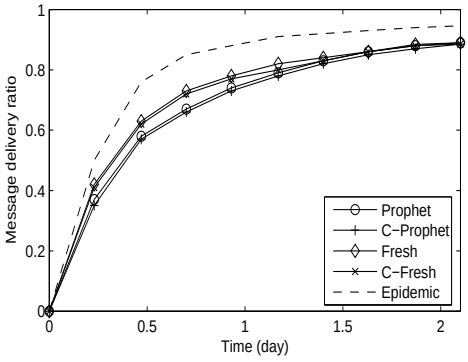
(c) Routing Efficiency vs. time

Figure 5.11: Comparison of metric-based forwarding algorithms using Cambridge traces

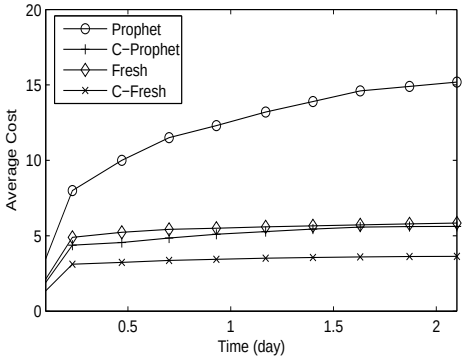
approaches (per-hop and per-contact routing).

5.4.4.2 Comparison of revised and original versions of metric-based algorithms

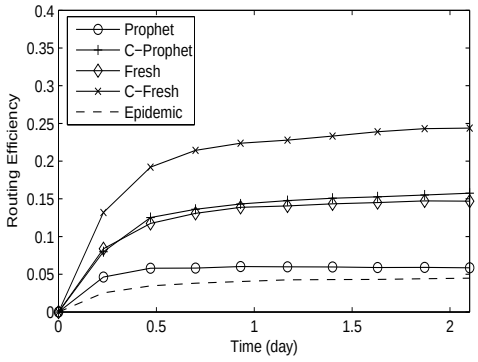
In Figure 5.10a, we show the delivery ratios achieved in RollerNet traces. Clearly, the modified algorithms provide higher delivery ratio than the original ones. Moreover, as Figure 5.10b shows, average cost is lower for the modified versus the original algorithms. For example, C-Prophet delivers 90% of all messages after 23 minutes with average delay of 7.8 minutes and average cost of 4.83 hops. However, the original Prophet reaches the same delivery ratio only after 33 minutes with average delay of 13.5 minutes and 17.02 average cost. A similar situation is also observed between C-Fresh and Fresh. As a result, more than 100% increase in



(a) Message delivery ratio vs. time



(b) Average cost vs. time

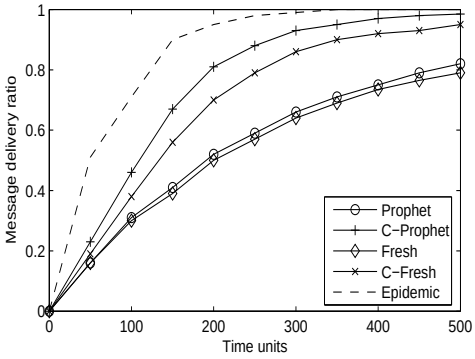


(c) Routing Efficiency vs. time

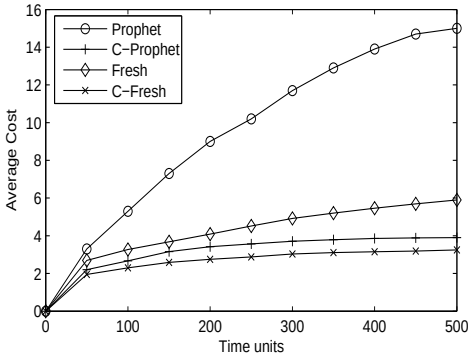
Figure 5.12: Comparison of metric-based forwarding algorithms using Haggle Project traces

routing efficiency is achieved.

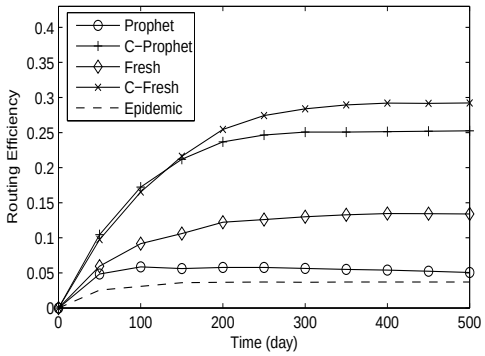
When we look at the results obtained from Cambridge and Haggle traces in Figure 5.11 and Figure 5.12, we observe a different improvement. As it is seen in Figure 5.11a and Figure 5.12a, revised and original versions of algorithms have similar delivery ratios (and therefore similar average delays). However, as Figure 5.11b and Figure 5.12b show, average costs in modified versions are lower than they are in the original ones. Moreover, in Cambridge traces, the mean hop counts of Prophet, C-Prophet, Fresh and C-Fresh are 5.21, 2.48, 3.83 and 2.53, respectively and in Haggle traces, they are 12.7, 4.98, 5.23 and 3.44. This shows that when conditional intermeeting time is used as an additional delivery metric, the nodes choose better next hops so that the cost is decreased while still keeping the original delivery ratio. Therefore, again more than 100% gain is achieved in routing efficiency. The results



(a) Message delivery ratio vs. time



(b) Average cost vs. time

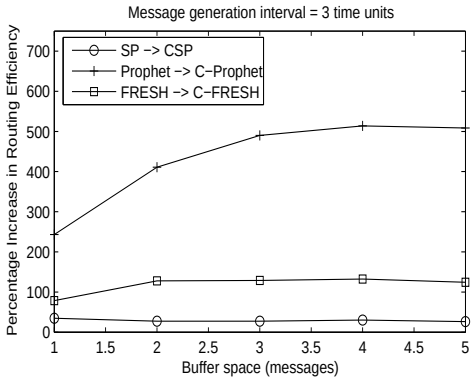


(c) Routing Efficiency vs. time

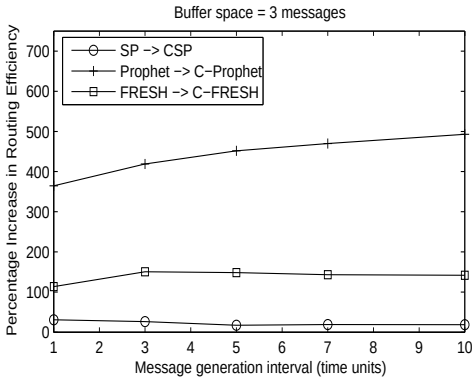
Figure 5.13: Comparison of metric-based forwarding algorithms using Synthetic Data

with synthetic data in Figure 5.13 also demonstrates the superiority of revised algorithms. More messages are delivered with lower cost when compared to the original algorithms.

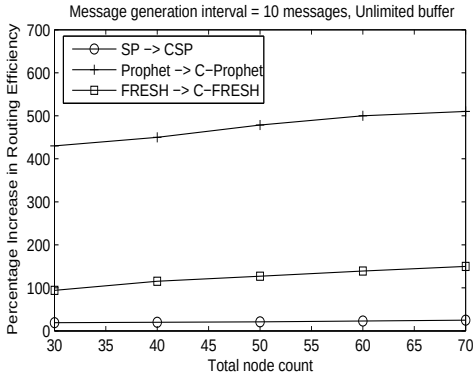
From the above results, we observe the benefit of conditional intermeeting time in metric-based forwarding algorithms clearly. Moreover, we also notice that in different environments, the improvements obtained thanks to utilization of conditional intermeeting times are different. In RollerNet and synthetic traces, we observed improvement in all metrics. However, in Cambridge and Haggle traces we observed an improvement only in average cost, thus, in routing efficiency. From our initial analysis of the traces, we conclude that this is caused by the repetitive contacts between objects in RollerNet and synthetic traces. For example, as it is stated in [45], during the tour, fluctuations in the motion of the rollerbladers cause a typical accordion



(a) Percentage of increase in routing efficiency vs. buffer space



(b) Percentage of increase in routing efficiency vs. message generation interval



(c) Percentage of increase in routing efficiency vs. total node count

Figure 5.14: Effects of parameters on simulations with synthetic data.

phenomenon (the topology expands and shrinks with time). When the topology shrinks, nodes are in contact with each other, but when the topology expands, they are disconnected. This property in the contacts of nodes creates a cyclic behavior and lets the proposed algorithms perform better. Similarly in synthetic traces, people are assigned outer communities to visit regularly, thus repetitive mobility behaviors occur. On the other hand, in Cambridge and Haggle traces, the repetitive behavior of node meetings is not apparent. From these results, we conclude that the benefit of using conditional intermeeting times in metric based algorithms becomes more pronounced in the environments in which the repetitive motion of nodes is clearly observed. However, even in the environments where this is not the case, average cost of routing can still be decreased and routing efficiency can be improved remarkably thanks to use of conditional intermeeting times.

5.4.4.3 Effects of Simulation Parameters on Results

We also look at the effects of some parameters on the results. Since the effects are turned out to be similar on each data set, here we show the results with only synthetic traces.

First, we look at the scenarios where the buffer space at nodes are limited. Assuming that nodes use FIFO buffer management scheme, we computed the increase obtained in the routing efficiency via utilization of proposed metric over the original algorithms in these environments. Figure 6.17 shows the results for different buffer sizes in the range of [1-5] messages. For these simulations we kept the message generation interval, t=10 time units and TTL=1000 time units. The results show that in the modified versions of algorithms, the increase in the routing efficiency grows as the buffer space increases. Moreover, the amount of increases converge to some constant values after sufficient buffer spaces. CSP, C-Prophet and C-Fresh offers 20%, 100% and 500% increase in the routing efficiency over their original algorithms, respectively.

In Figure 5.14b, we observe similar results with different message generation intervals. As the messages are generated more frequently, due to buffer overflow, some messages are lost. However, the routing efficiency of algorithms is still remarkably increased with modified versions. Finally, in Figure 5.14c, we observe the results with different node counts in the network. Clearly, the increase in routing efficiency rises as the node count increases. This is because in synthetic data, the correlation between the meetings of nodes increases due to higher number of nodes in each community. Thus, conditional intermeeting time provides more accurate information about node relations.

The results with different values of simulation parameters show that the improvement achieved in the performance of algorithms with the utilization of the proposed metric is solid and valid with different settings. Other than the above parameters, we also performed simulations with different bandwidth, failure rate, speed ranges for nodes etc. All of these results still show the better performance of revised versions of the algorithms over the original ones with a remarkable increase in routing efficiency.

5.5 Summary of Contributions

In this chapter, we studied the effect of correlated mobility in single-copy based DTN routing algorithms. First, inspired by the results of the recent studies showing that intermeeting times between nodes are not memoryless and the motion patterns of mobile nodes are frequently repetitive, we introduced a new metric called conditional intermeeting time which is the average time that passes from the time a node meets with a neighbor node until the time it meets another one. Next, we presented an analysis of this metric showing why it can be beneficial in more accurate representation of node relations. Then, we looked at the effects of this metric on existing DTN routing algorithms. To this end, we modified their current designs using conditional intermeeting time. Finally, through extensive simulations based on both the real DTN traces and synthetic mobility traces, we evaluated the modified algorithms and demonstrated the superiority of them over original ones.

As a future work of our study, we would like to extend the definition of conditional intermeeting time by using more meetings from the contact history. For instance, assume that a node A wants to compute its conditional intermeeting time with a node B after the time it has met another node C. Here, we want to differentiate the following two cases which are considered together in our current approach; when node A has met node D before node C and when node A has met node E before node C. To apply this algorithm in our work, we plan to use probabilistic context free grammars (PCFG) and utilize the construction algorithm presented in [57, 54].

CHAPTER 6
EXPLOITING SOCIAL RELATIONS FOR EFFICIENT DTN ROUTING

Although there are remarkable amount of studies proposing routing algorithms for DTNs, very few of them take into account the effect of social structure of the network or the social relations between nodes on the design of the routing algorithms. It is always noted in many studies (i.e. [44]) that the mobility of nodes in a mobile (social) network and the interactions between nodes is not purely random and homogeneous but it is somewhat a mixture of homogeneous and heterogeneous behaviors. In other words, in a real mobile network, we always see grouping of nodes into classes or communities such that the nodes within the same community behave similarly and the nodes from different communities show different behaviors.

Consider a Pocket Switched Network (PSN) [61] which is a kind of social network in which people are intermittently connected via different wireless devices including cell phones and GPS devices. The connectivity among these human-carried devices (i.e. people) is achieved when they get into the range of each other. In a social network, the relationship defining the connectivity frequency among the nodes can be various interdependencies including friendship, trade and status. Therefore, for an efficient routing of messages in such networks, the mobility of nodes and the underlying community structure among the members of the whole society has to be carefully considered. For example consider a high school network. Students in the same class have higher chance to see (so to transfer data) each other than the students from other classes (i.e. they can probably meet only during breaks).

In this chapter, we study the routing problem in people-centric DTNs (i.e. mobile social networks, pocket switched networks) considering the social network concepts.

In the first section, we study the effect of the social structure of DTNs in Spray and Wait routing and show that making routing decisions considering the

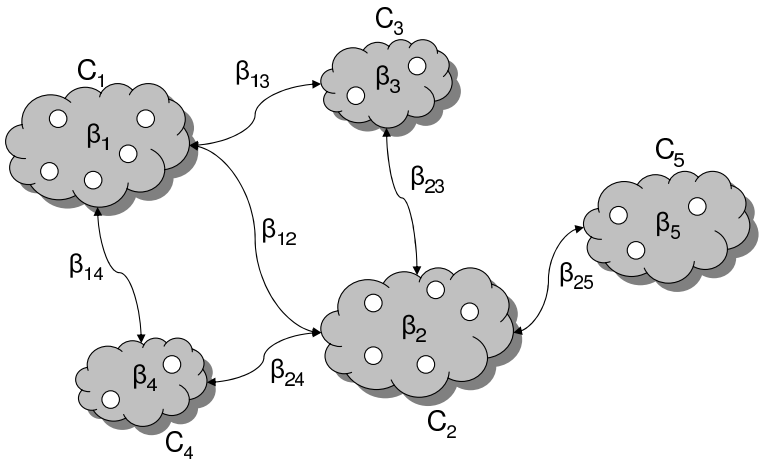


Figure 6.1: A sample social network structure with five communities. Each community has different inner and inter-community meeting rates.

inner structure of the network increases the performance of routing [75]. More specifically, we discuss the problem of selecting the nodes to which the message is replicated in social (community-based) delay tolerant networks in which the nature of standard delay tolerant networks changes because of the heterogeneous inter-meeting time between the nodes in the network (Note that, a homogeneous network model is used in chapter 3 and in [37]).

In the second section, we analyze the real DTN traces and utilize social relations between nodes and introduce a friendship based routing algorithm [70]. First, inspired by the properties of friendship relations between people, we define a new metric to understand social relations between nodes more properly. Second, we propose a local community formation algorithm based on this new friendship detection metric. We use not only direct relations but also indirect ones in a different way than it was considered previously. Third, we introduce a new approach to handle periodic changes of node relations.

6.1 Impact of Social Structure on Spray and Wait Routing

6.1.1 Network Model and Assumptions

To illustrate the general picture of communities in a social network, we use the following model. Assume that there are m communities (C_1 to C_m) in the whole network and there are N_i nodes in community C_i. Moreover, assume that the nodes in community i get contact with the nodes in community j with an average intermeeting time of β_{ij} (for simplicity $\beta_i = \beta_{ii}$). In other words, they find chance to exchange their data in every β_{ij} time units on the average (they contact each other after each t time units where t is an exponential distributed with mean β_{ij}). Here note that the nodes within the same community are considered identical in terms of meeting behavior with other nodes, but the nodes from different communities are considered having different behaviors. Accordingly, both the homogeneity and the heterogeneity structures are embedded into the network structure. A sample network with five communities is shown in Figure 6.1.

The beauty of this model is that it successfully monitors the general behavior of nodes in community-based social networks. It avoids dealing with individual behaviors of nodes and provides only the average intermeeting time of nodes both inside and outside the community. Consider the examples of real life PSN scenarios. Nodes get in contact with each other depending on their relations in the society. Moreover, this contact times may sometimes happen unpredictably. However, even in such cases, we claim that on the average there may occur stable intra- and inter-community intermeeting values in the whole network and these can be found using the histories of node meetings [67].

6.1.2 Challenges and Tradeoff of Efficient Routing

In multi-copy based routing algorithms, the main goal is to deliver a message in a source node s to a destination node d by generating multiple copies of the message and spreading them to different nodes in the network. Once one of the copies is delivered, the message itself is delivered. Clearly, the number of copies generated and distributed to the network defines the characteristics (i.e. delay, cost) of the delivery. This is also the main reason of why researchers always have

focused on the design of routing algorithms with efficient number of copies of the message.

It is obvious that we can increase the delivery probability and decrease delivery delay of a message, by just increasing the number of copies that will be distributed to the network. However, we also need to distribute the copies by taking into account the meeting frequencies between nodes (effect of community structure). Assume that s has a message to deliver to d in the network. Furthermore, assume that s is allowed to distribute at most $L-1$ copies of the message to the other nodes (the other nodes are not allowed to replicate the message). Therefore, once all copies of the message are given to other nodes, the total number of copies in the network will be L. The instant strategy that comes to mind is to allow s to give these copies to the first $L-1$ nodes that it meets in the network. By this way, the fastest distribution of the copies is achieved and the waiting phase is immediately started and the delivery of the message is attempted independently by any of the nodes having the message copy. If s and d are in the same community, this strategy is reasonable and works well especially in scenarios where the future node meetings are unknown.

However, if s is not in the same community with the destination, this strategy loses its effectiveness due to its copy distribution without considering the community information. The copies may be given to nodes which have low chance to meet d, thus to deliver the message. For example, consider a society with three communities (source's community (C_s), destination's community (C_d) and another community (C_e)). Moreover, assume that the intermeeting times between the nodes of each community and different communities hold the following reasonable relations: $\beta_s = \beta_d = \beta_e$, $\beta_{sd} = \beta_{se} = \beta_{de}$ and $\beta_s << \beta_{sd}$. In this sample scenario, there are three cases of message copying in terms of its effects on the copying and delivery time:

- s can give copies to nodes within its own community. Since it meets these nodes more frequently than other nodes, the duration of message copy distribution to these nodes takes less time than copying to C_d's nodes. But, on the other hand, since the nodes in C_s meet the nodes (i.e. d) in C_d less frequently than C_s's nodes, the probability of message delivery is lower, so that average

delivery delay gets longer.

- s can give copies to nodes that are in C_d. This provides less waiting for nodes to meet d after they have copies. However, s meets with these nodes less frequently than the nodes in C_s so that the copying phase is longer.

- s can give copies to nodes that are in C_e. Here, since s meets these nodes infrequently and after the copying process is done, these nodes meet the destination infrequently, giving copies to such nodes is not an efficient strategy to reduce the delivery delay.

When we look at the above three cases, we figure out that the first and second cases have tradeoffs in terms of copying and waiting durations. But the third case has disadvantages during both the copying and waiting times. Therefore, an efficient strategy to decrease the delivery delay must take into account the first two cases in the distribution of message copies, but the number of copies used in either case must be carefully decided to obtain the optimum delivery delay. In the next section, we provide an analysis of delivery delay with different number of copies given to source's community (L_{in}) and destination's community (L_{out}) in this sample scenario.

6.1.3 Analysis of Delivery

In this part, we will compute the expected delivery delay that can be achieved in the network where the source node s gives the copies of the message either to the nodes in C_s or C_d. For the sake of simplicity, we make the following assumptions. Let $N{=}n{+}1$ denote the number of nodes in C_s and C_d ($N_s{=}N_d{=}N$). We know that, on the average, s meets all other n nodes in its own community within β_s time units. Therefore, if we assume that the average time of meeting any other node is a single time unit, then it follows that $\beta_s = n$. Moreover we assume that $\beta_{sd} = k\beta_s = k\beta_d$ where $k > 1$.

In this model, as it is seen in Figure 6.2, there are two independently running processes by which delivery can happen:

Local Spraying: Source distributes $L_{in} - 1$ additional copies of the message (in total there are L_{in} copies with the copy in s) to the other nodes that are in the same

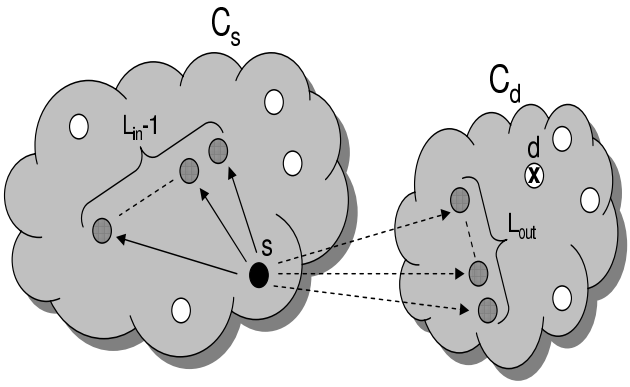


Figure 6.2: Distribution of copies to source's and destination's communities.

community with itself (C_s). Then, each of these nodes can deliver the message to the destination with probability $\frac{1}{nk}$ in each time unit. Since from time $i-1$ to i, on the average, there are i[16] nodes having the message copy in C_s, the total gained probability of delivery by the nodes in C_s becomes $\frac{i}{nk}$ at time i. Here, note that the case of direct delivery of the message (to d) by source is also included in this type of delivery which occurs of course with the same probability.

Global Spraying: Source gives $L_{out} = L - L_{in}$ copies of the message to the nodes that are in the same community with destination (C_d). Then, each of these nodes can deliver the message to the destination with probability $\frac{1}{n}$ in each time unit. The number of nodes having copy in C_d is zero at the beginning and on the average source can give a copy to a node in C_d in every k^{th} unit. As a result, we can assume that in a time unit, there is only $1/k$ copies given to such nodes so that the total probability of delivery by the nodes (in C_d) having copy becomes $\frac{i-1}{nk}$ (i.e. until time 1 it is zero).

Now, we will calculate both the probability of delivery and the expected delivery time of a message in such a network model. We need to combine the probabilities of two processes in a time unit. One can easily see that there are three different phases in the delivery process of the message. In the first (*All Spraying*), both the

[16]Here, we ignore the cases where s meets the nodes already having copy for simplicity. Since we mostly study the cases where $L_{in} << N_s$, the effect of these cases on the total probability is very low.

local and global spraying will continue and at each time unit the delivery probability will be increased by both processes. In the second phase (*Mixed*), only one of these processes will continue spraying, the other one will stop spraying and enter waiting phase. Here, note that depending on the parameters L_{in} and k, either of these processes can end up spraying before the other one. We need to consider this in our calculation. Finally, in the third phase (*All Waiting*), both of these processes stop spraying and run their waiting processes which means that they contribute to the delivery probability with constant copy counts.

We can assume that local spraying ends before global spraying if the following condition is satisfied:

$$\begin{aligned} L_{in} - 1 &\leq k(L - L_{in}) \text{ , so when} \\ L_{in} &\leq \frac{k}{k+1}L + \frac{1}{k+1} \end{aligned}$$

According to these observations, if local spraying ends before global spraying (case A), then the delivery probability of a message in *All Spraying* phase can be calculated as:

$$\begin{aligned} P_1 &= \sum_{i=1}^{L_{in}-1} D'_1(i) \left(\frac{2i-1}{nk}\right), \text{ where} \\ D'_1(i) &= \prod_{j=1}^{i-1} \left(1 - \frac{2j-1}{nk}\right) \end{aligned}$$

Here, $\frac{2i-1}{nk}$ denotes the probability of delivering at the i^{th} time unit and the product term denotes the probability of not delivering before the i^{th} time unit.

In the second phase (*Mixed* phase), since the local process finishes its spraying, the probability of delivery at a time unit changes and the total delivery probability becomes:

$$\begin{aligned} P_2 &= \sum_{i=L_{in}}^{k(L-L_{in})} C_1 D'_2(i) \left(\frac{L_{in} + (i-1)}{nk}\right), \text{ where} \\ D'_2(i) &= \prod_{s=L_{in}}^{i-1} \left(1 - \frac{L_{in} + (s-1)}{nk}\right) \end{aligned}$$

$$C_1 = \prod_{j=1}^{L_{in}-1} \left(1 - \frac{2j-1}{nk}\right)$$

In the *All Waiting* phase, since spraying of copies ends in both processes, then the delivery probability is increased by a constant probability at each time unit. Hence, the total delivery probability in the third phase is computed as:

$$\begin{aligned} P_3 &= \sum_{i=k(L-L_{in})+1}^{\infty} C_1 C_2 D_3'(i) \left(\frac{L_{in} + k(L-L_{in})}{nk}\right), \text{ where} \\ D_3'(i) &= \left(1 - \frac{L_{in} + k(L-L_{in})}{nk}\right)^{i-k(L-L_{in})-1} \\ C_2 &= \prod_{s=L_{in}}^{k(L-L_{in})} \left(1 - \frac{L_{in} + (s-1)}{nk}\right) \end{aligned}$$

But if the global spraying ends before local spraying (Case B) then the formulations need to be updated due to changes in the boundaries between the three phases:

$$\begin{aligned} P_1 &= \sum_{i=1}^{k(L-L_{in})} D_1'(i) \left(\frac{2i-1}{nk}\right) \\ P_2 &= \sum_{i=k(L-L_{in})+1}^{L_{in}} C_1 D_2'(i) \left(\frac{k(L-L_{in})+i}{nk}\right) \\ P_3 &= \sum_{i=L_{in}}^{\infty} C_1 C_2 D_3'(i) \left(\frac{L_{in} + k(L-L_{in})}{nk}\right) \end{aligned}$$

where, $D_1'(i)$ remains same as in above but $D_2'(i)$, $D_3'(i)$, C_1 and C_2 change as follows:

$$\begin{aligned} D_2'(i) &= \prod_{s=k(L-L_{in})+1}^{i-1} \left(1 - \frac{k(L-L_{in})+s}{nk}\right) \\ D_3'(i) &= \left(1 - \frac{L_{in} + k(L-L_{in})}{nk}\right)^{i-L_{in}} \\ C_1 &= \prod_{j=1}^{k(L-L_{in})} \left(1 - \frac{2j-1}{nk}\right) \end{aligned}$$

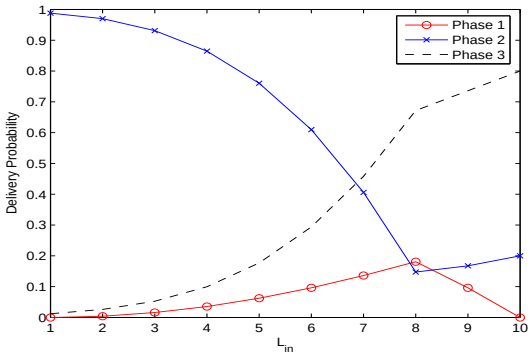


Figure 6.3: Delivery probabilities when $k = 5$ and $L = 10$

$$C_2 = \prod_{s=k(L-L_{in})+1}^{L_{in}} \left(1 - \frac{k(L - L_{in}) + s}{nk}\right)$$

Using the above formulations, we can compute the average delivery probability in each of the three phases separately. As an example, we calculated these probabilities for two different configurations and plotted the results in Figure 6.3 and Figure 6.4. While in the former graph (L, k) pair is assumed to be (10, 5), in the latter they are assigned (15, 3) values (N=50). In both of these figures, note that, the delivery probability in the first period has a maximum point which is obtained at the biggest integer value of L_{in} that is less than the boundary value. That point is also the optimum point for second period where the minimum probability value is achieved. This is because the duration of second period gets smaller when L_{in} gets closer to boundary point. It is also important to note that when $L_{in} = 1$, the message is most probably (≈100%) delivered in the second phase (only global spraying) but on the other hand, when $L_{in} = L$, the delivery probability in this mixed phase (only local spraying) is much smaller than 100%. This is caused by the longer duration of global spraying than local spraying (when $L_{in} = 1$ and $L_{out} = L - 1$) which increases the delivery probability of the message (in $Mixed$ period) by nodes already having copy (in C_d) while source is still trying to distribute remaining copies to the nodes in C_d (which of course takes longer).

The above formulations are to estimate the delivery probability in each of the three phases. To estimate the expected delivery time in a period i, ED_i, we

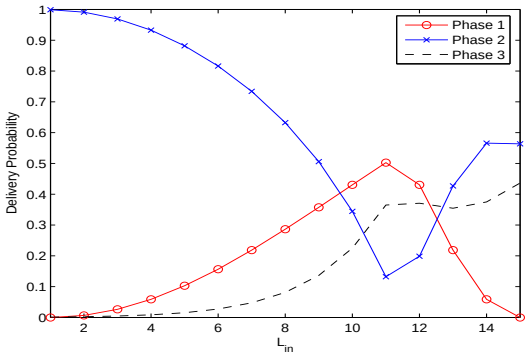


Figure 6.4: Delivery probabilities when $k = 3$ and $L = 15$

simply multiply P_i by i. Then, summing these ED_i values gives us the expected delivery time in such a spraying algorithm. We will show the computed ED values for the same (L, k) pairs used in the previous figures and validate the results with simulations in the next section.

6.1.4 Simulation Results

We updated our Java-based DTN simulator to see the effects of different L_{in} and L_{out} values. For our initial simulations, we work on a network where the messages are distributed to the nodes in either the source's community or destination's community (we will work on more complex social network models in the future work). That's why we created a network with two communities where there are 50 mobile nodes in each community. We deploy the nodes onto a torus of the size 300 m by 300 m. All nodes are assumed to be identical and their transmission range is set at $R = 10$ m. Nodes move according to random direction mobility model [44]. The speed of a node is randomly selected from the range [4, 13]m/s and its direction is also randomly chosen. Then, each node goes in the selected random direction at the assigned speed for an epoch duration. Each epoch's duration is again randomly selected from the range [8, 15]s. The meeting times of nodes are assumed to be independent and identically distributed (IID). Furthermore, we also assume that the buffer space in a node is infinite and the communication between nodes is perfectly separable, that is, any communicating pair of nodes do not interfere with any other

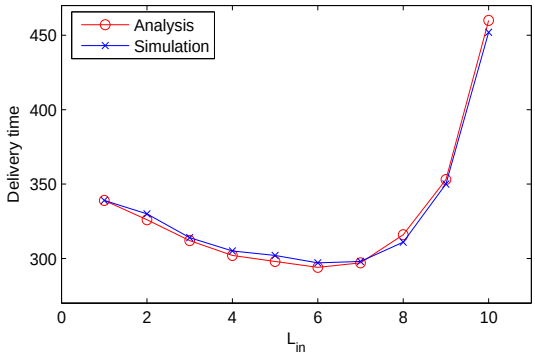


Figure 6.5: Simulation vs. analysis showing the expected delivery time when $k = 5$ and $L = 10$

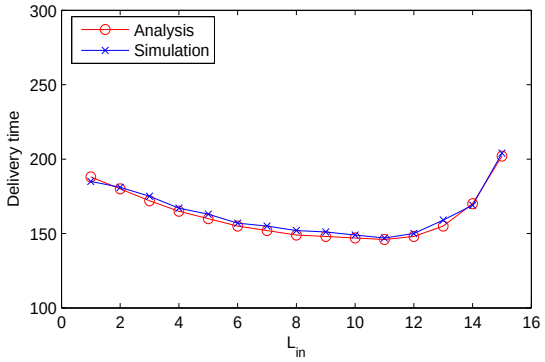


Figure 6.6: Simulation vs. analysis showing the expected delivery time when $k = 3$ and $L = 15$

simultaneous communication. We used different values of k to see its effect on the performance of the algorithm. To simulate nodes from different communities (C_s and C_d) which meet each other in every $\beta_{sd} = k\beta_s$ time units on the average, we ignored the first $k - 1$ meetings of such node pairs and treated the k^{th} meeting as a real meeting (here note that average meeting time between two encounters of any pair of nodes is β_s or β_d).

We have created messages at a randomly selected source node for delivery to a randomly selected destination node in the other community. Then, we collected some useful statistics from the network. The results are averaged over 3000 runs.

First of all, to validate the analysis computation of average message delivery

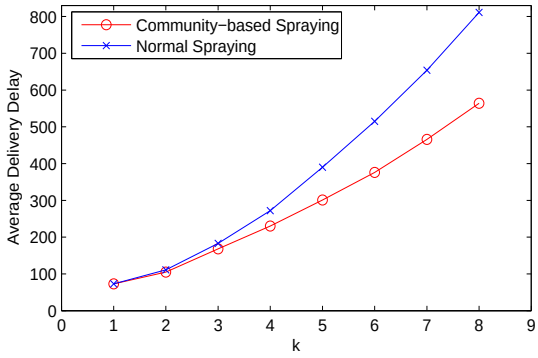


Figure 6.7: Average delivery delay with different k values when $L = 10$

delay, we did simulations with two different (L, k) pairs. Figure 6.5 and Figure 6.6 show the comparison of analysis and simulation results in terms of average delivery delay when (10, 5) and (15, 3) pairs are used, respectively. Since a single time unit is defined differently in our analysis, we adjusted results of the analysis accordingly. From these two graphs, we observe that the analysis and simulation results are matching, proving the correctness of the analysis.

We have also compared two spraying strategies: 1) Community based spraying where the L_{in} and L_{out} (in total L) values are set such that the minimum delay is achieved 2) Normal spraying algorithm [37] in which copies are given to the first $L - 1$ nodes met by the source node. Figure 6.7 and Figure 6.8 show the average message delivery delay and average message copy count (and therefore the cost) achieved in both algorithms with different k values when $L = 10$. It is clear that, as k increases, the difference of delivery delay obtained in both algorithms gets bigger. Furthermore, community based spraying algorithm also outperforms normal spraying algorithm in terms of used copy count per message (when $k = 8$, the improvement is around 15%). Since in the former, the distribution of copies to other nodes is designed considering the community structure in the network, we get improvements in both of these metrics.

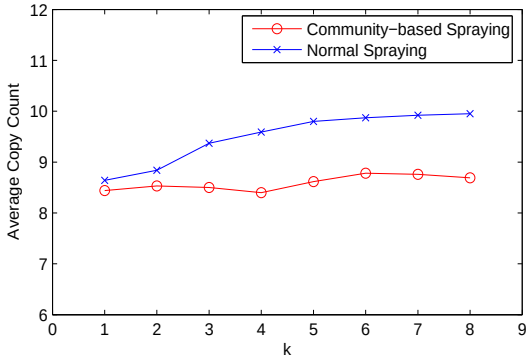


Figure 6.8: Average copy count used per message with different k values when $L = 10$

6.2 Utilizing Friendship Relations for Efficient Routing in Mobile Social Networks

In this section, different from the previous section which worked on random mobility models, we work on real DTN traces (especially mobile social networks (MSN) which are special kind of DTNs where the nodes are carried by people) and try to understand the social relations between nodes to be used in developing an efficient single-copy based routing algorithm.

First, we define a new metric to understand social relations between nodes more properly. Second, we propose a local community formation algorithm based on this new friendship detection metric. We use not only direct relations but also indirect ones in a different way than it was considered previously. Third, we introduce a new approach to handle periodic changes of node relations.

6.2.1 Analysis of Node Relations

The intermittent connectivity between nodes in an MSN makes the routing of messages possible only in opportunistic manner. That is, message exchanges occur only when two nodes come within the range of each other and one of them assesses that the other has higher delivery chance than itself. Hence, the link quality between each pair of nodes needs to be estimated accurately (from contact history) to consider the possible forwarding opportunity arising from the encounter. As a

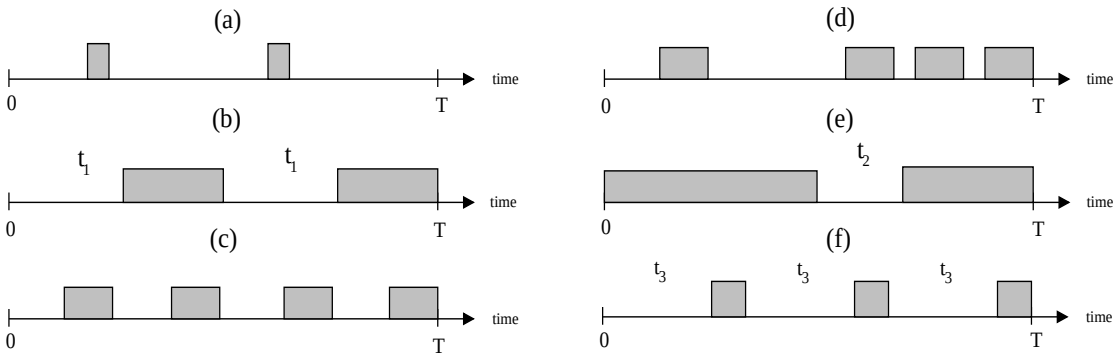


Figure 6.9: Six different encounter histories between nodes i and j in the time interval $[0, T]$. Shaded boxes show the encounter durations between nodes.

result, the periodic encounters between nodes can be condensed to a single link weight and the neighboring graphs of nodes can be constructed.

In previous works, several metrics, including encounter frequency, total or average contact period and average separation period [76], were used to extract the quality of links between pairs of nodes. However, all these metrics have some deficiencies in accurate representation of forwarding opportunity arising from encounters between nodes. For example, consider the six different encounter histories of two nodes, i and j, in Figure 6.9, where shaded boxes show encounter durations between these nodes in the time interval T. In cases a and b, the encounter frequencies are the same but the contact durations between nodes are longer in case b than in case a. Hence, encounter pattern b offers better forwarding opportunities than a does[17]. Comparing cases b and c, we notice that the contact durations are the same but the encounter frequencies are different. Since frequent encounters enable nodes to exchange messages more often, case c is preferable to case b for opportunistic forwarding.

Among the previously proposed metrics, encounter frequency fails to represent

[17]It should be noted that the comparison of all configurations in terms of message exchange opportunity obviously depends on the application scenario which defines the packet size. However, without loss of generality, here we assume that the encounter durations are long enough for sending of a packet.

the stronger link when cases a and b are considered, and the total contact duration fails for cases b and c. Although average separation period can assign correct link weights representing the forwarding opportunity in cases a, b and c, it fails in other cases. When we compare cases c and d, both the contact durations and the encounter frequencies are the same. However, case c is preferred to d due to the even distribution of contacts. In [76], preference of case c is achieved by utilizing irregularities in separation period as a penalty factor. However, deciding on how much it will affect the link quality in different cases is still difficult. Furthermore, for the cases such as e an f, average separation period fails to assign accurate link weights. If $t_1 = t_2$, average separation period cannot differentiate between cases b and e but case e is preferable due to its longer contact duration (average separation period can even give preference to case b if t_1 is slightly less than t_2). Similarly, if $t_1 = t_3$, average separation period cannot differentiate between cases b and f, even though case b offers better forwarding opportunity.

To find a link metric that reflects the node relations (also the forwarding opportunities) more accurately, we considered the following three behavioral features of close friendships: high frequency, longevity, and regularity. In other words, for two nodes to be considered friends of each other, they need to contact frequently and regularly in long-lasting sessions. Here, frequency refers to average intermeeting time while regularity refers to the variance of the intermeeting time. Hence, two nodes may meet infrequently but regularly (e.g. once a week) and still be considered friends. This is of course a weaker friendship than the one with both frequent and regular contacts. The previous metrics take into account some of these features but not all of them at the same time. We account for these properties in a new metric that we called *social pressure metric* (SPM). It may be interpreted as a measure of a social pressure that motivates friends to meet to share their experiences. In our setting, this amounts to answering the question 'what would be the average message forwarding delay to node j if node i has a new message destined to node j at each time unit when the time unit tends to zero (so the result is independent of time units)?'. Then, we define the link quality ($w_{i,j}$) between each pair as the inverse of

this value. More formally:

$$SPM_{i,j} = \frac{\int_{t=0}^{T} f(t)dt}{T} \text{ and } w_{i,j} = \frac{1}{SPM_{i,j}}$$

where $f(t)$ represents the time remaining to the next encounter of the two nodes at time t. If at time t, the nodes are in contact, then $f(t) = 0$, otherwise, $f(t) = t_{next} - t$, where t_{next} is the time of the next meeting between nodes i and j. Hence, each intermeeting time t_{inter} contributes term $t_{inter}^2/(2T)$ to SPM. If there are n intermeeting times in the time period T, then $SPM_{i,j} = \left(\sum_{x=1}^{n} t_{inter,x}^2\right)/(2T)$ and $w_{i,j} = (2T)/\left(\sum_{x=1}^{n} t_{inter,x}^2\right)$.

The larger the value of $w_{i,j}$, the closer the friendship (the higher the forwarding opportunities) between nodes i and j. Clearly, increasing the time the nodes are in contact decreases SPM as does equalizing the time between encounters. Finally, splitting the intermeeting times into larger number of smaller pieces also decreases the SPM. Hence, indeed, this metric combines the three desired properties of the friendship behaviors discussed above into a single measure. To illustrate the benefits of this metric, we notice that when it is used to evaluate all cases in Figure 6.9, the resulting weights will accurately indicate which case offers more forwarding opportunities. It should also be noted that SPM is computed from the history of the encounters of the node. As additional node encounters happen, the corresponding SPM value is updated easily.

6.2.2 Friendship Community Formation

Using its encounter history, each node can compute qualities ($w_{i,j}$ values) of its links with other nodes. Then, it can define its friendship community as a set of nodes having a link quality with itself larger than a threshold (τ). This set will include only direct friends. However, two nodes that are not close friends directly (they even may not have contacts at all) still can be close indirect friends. This happens if they have a very close friend in common so that they can contact frequently through this common friend. Moreover, the relations between nodes may show periodic changes. For example, they could depend on the time of the day or the day of the week considered. Therefore, both strong indirect relationships and periodic variations of

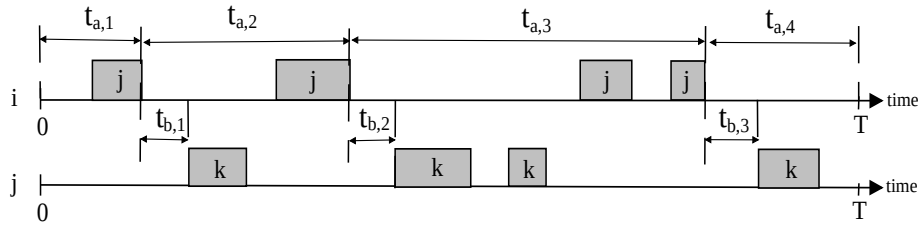


Figure 6.10: Encounter history between nodes i and j (upper diagram) and between nodes j and k (lower diagram) in the same time interval $[0, T]$.

relationships must be addressed when forming friendship communities.

6.2.2.1 Handling Indirect Relationships

To find indirect friendships between nodes in a way relevant for routing, we propose to use *relative* SPM (or simply $RSPM$) metric. Consider the sample encounter history shown in Figure 6.10 in which the upper diagram shows the contacts between nodes i and j, while the lower one shows the contacts between nodes j and k. We define $RSPM_{i,k|j}$ as the answer to the question 'what would be the average delivery delay of node i's continuously generated messages if they followed the path $<i,j,k>$?'. Each indirect information passing consists of two stages. The first one starts at the last meeting of node i with node j and ends at the time node i's next contact with node j ends (assuming that any message generated at node i can be transferred to node j when they are in contact). Here, if there are several subsequent meetings with j before any meeting of j with k, then the last one is considered. We denote duration of this stage as $t_{a,x}$ where x denotes the number of indirect information passing occurring. During this stage, node i transfers messages to node j. The second stage starts when the first one ends and it finishes when node j meets node k. The duration of this session is denoted $t_{b,x}$. During this stage, the messages accumulated at j merely wait for the meeting with the destination (without accumulating further at node j). Example is given in Figure 2, in which in time T, there are three full information passing sessions between nodes i and k via

node j, and the beginning of the fourth one. Denoting the number of such sessions as n, $RSPM_{i,k|j}$ is computed as:

$$\begin{aligned} RSPM_{i,k|j} &= \left(\sum_{x=1}^{n} \int_{0}^{t_{a,x}} (t_{b,x} + t_{a,x} - t)dt \right) /T \\ &= \frac{\sum_{x=1}^{n} (2t_{b,x}t_{a,x} + t_{a,x}^2)}{2T} \end{aligned}$$

Since the intermediate node, j, records all of its past contact times with i and k, it can compute the value of $RSPM_{i,k|j}$.

In Algorithms 6-8, we give the details of computation of SPM and $RSPM$ values from a node's point of view. At the beginning, each node j initializes its parameters as described in Algorithm 6. Then, when a new node m is encountered, it updates the value of $SPM[m]$ and for each of its other contacts, i, it updates the value of $RSPM[i][m]$ if node m is encountered first time after its meeting with node i (Algorithm 7). When the meeting of a node with another node m ends, it also updates the value of $SPM[m]$ and sets the end time of current $t_a(m,k)$ and start time of next $t_b(m,k)$ to the current time t for each of its other contacts k (Algorithm 8).

Algorithm 6 initialize (node j)

```
1: for each i ∈ N and i ≠ j do
2:    cur_total1[i] = 0
3:    t_pre^end(i) = 0
4:    for each k ∈ N and k ≠ j ≠ i do
5:       t_a^start(i,k) = 0
6:       t_a^end(i,k) = 0
7:       cur_total2[i][k] = 0
8:    end for
9: end for
```

In an MSN, each node can detect its direct friendships from its own history (by computing SPM values). However, to detect indirect friendships, a node needs $RSPM$ values from its friends. Once such $RSPM$ values are received and updated at the encounter times with its friends, each node can form its friendship community

Algorithm 7 neighborDetected (node m, time t)

1: $t_{cur}^{start}(m) = t$
2: $nt = t_{cur}^{start}(m)$ - $t_{pre}^{end}(m)$
3: $cur\_total1[m]$ += $\frac{nt(nt+1)}{2}$
4: $SPM[m] = cur\_total[m]/2t$
5: **for** each $i \in N$ and $i \neq m$ **do**
6: $t_b^{end}(i,m) = t$
7: **if** $t_a^{end}(i,m) > t_a^{start}(i,m)$ **then**
8: $t_b(i,m) = t_b^{end}(i,m)$ - $t_b^{start}(i,m)$
9: $t_a(i,m) = t_a^{end}(i,m)$ - $t_a^{start}(i,m)$
10: $cur\_total2[i][m] += 2t_b(i,m)t_a(i,m) + (t_a(i,m))^2$
11: $RSPM[i][m] = cur\_total2[i][m]/2t$
12: $t_a^{start}(i,m) = t_a^{end}(i,m)$
13: **end if**
14: **end for**

Algorithm 8 neighborLeft (node m, time t)

1: $t_{pre}^{end}(m) = t$
2: $SPM[m] = cur\_total1[m]/2t$
3: **for** each $k \in N$ and $k \neq m$ **do**
4: $t_a^{end}(m,k) = t$
5: $t_b^{start}(m,k) = t$
6: **end for**

using the following definition:

$$\begin{aligned} F_i &= \{j \mid w_{i,j} > \tau \text{ and } i \neq j\} \cup \\ &\quad \{k \mid w_{i,j,k} > \tau \text{ and } w_{i,j} > \tau \text{ and } i \neq j \neq k\} \end{aligned}$$

where $w_{i,j,k} = 1/RSPM_{i,k|j}$. The above equation enables nodes to detect their one-hop direct and two-hop indirect friends. Indirect friendships can also be generalized to friends more than two hops away. However, we have not included such extension because [76] demonstrated that nodes in the same community are usually at most two hops away from each other.

Clearly, the introduced method for detecting the indirect strong links between nodes is different than previous approaches (based on transitivity [25, 46, 76]) which basically consider the links between node pairs separately and assume a virtual link between node i and k if $w_{i,j}w_{j,k} > \tau$. However, in our model we can detect indirect

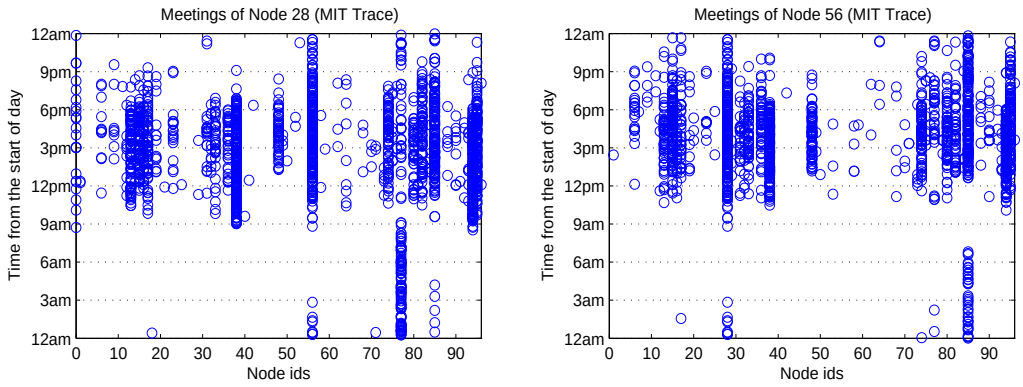


Figure 6.11: Encounter distributions of node 28 and 56 in MIT traces.

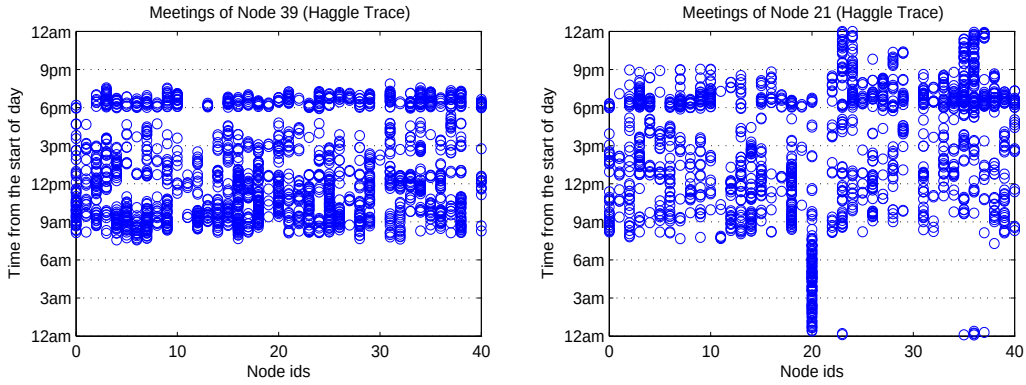


Figure 6.12: Encounter distributions of node 39 and 21 in Haggle traces.

relations more accurately. For example, if node j has a weak link with node k, $w_{i,j}w_{j,k}$ may be less than τ. However, if node j usually meets node k in a short time right after its meeting with node i, our metric can still identify node k as a friend of node i. This definition of indirect node relations is particularly meaningful within the context of routing because a node receives a message from one of its contacts and sends it to another contact. That is, it holds the message between its contacts with two different nodes. Hence, this metric accurately estimates the indirect opportunistic message exchange quality between two nodes.

6.2.2.2 Handling Periodic Variation in Node Relations

Node relations in an MSN often change with time periodically. Such periodic changes must be addressed for accurate computation of link qualities between nodes.

When we analyzed two commonly used mobile social network data (MIT Reality dataset [80] and Haggle dataset [89], see Section 6.2.4 for details), we have observed periodic [79] variations in node relations.

In Figures 6.11 and 6.12, we plotted the distribution of encounter times of two different nodes[18] in each dataset with other nodes in their dataset. Clearly, nodes encounter other nodes in some specific periods of the day. For example, in MIT traces, node 28 meets with node 38 usually between 9am to 7pm while it meets with node 48 usually between 1pm to 7pm. Similar behavior is also seen in Haggle traces.

Considering the fact that main activities of people are periodic, it is reasonable to expect similar behaviors in other MSNs. For instance, i can be a school, work or home friend of a node j and their encounter times then would differ accordingly. Moreover, i can be both school and home friend of j in which case they stay together during the entire day.

To capture the impact of temporal changes of node relations on the link quality, previous works have proposed to use some aging mechanisms [25] [69]. However, these mechanisms do not take into account the periodicity of node relations and react slowly to temporal changes of link quality. For example, around 7pm, the quality of link from node 56 to node 38 (see Figure 6.11) starts to decrease with aging effect[19] but still keeps a high value for some time. Yet, node 56 usually does not meet with node 38 until 10am next day. Therefore, forwarding a message considering an aged but still strong link quality may cause high delays when the link is already in its periodic low.

To reflect the periodic variation of the strength of friendship, we propose to use periodic friendship communities in our protocol. That is, each node i computes its F_i for different periods of the day and has different friendship communities in different periods. For example, if we divide a day into three hour periods, as shown in Figure 6.11, node 85 can be the only friend of node 56 in period 3am-6am, whereas nodes 28, 85 and 95 can be friends of node 56 in period 9pm-12am. In Haggle traces (Figure 6.12), we also observe similar situations. While nodes 23, 24, 35 and 36 are

[18] These nodes are 28 and 56 in MIT traces, 39 and 21 in Haggle traces. We selected these nodes because they are the ones with the highest number of encounters (with other nodes).

[19] $w_{i,j} = w_{i,j}\alpha^t$ where t is the time since the last encounter and $0 < \alpha < 1$ is aging parameter.

Algorithm 9 periodEventHandler (event e, node m, time t, time $start\_index$, time end_index)

```
1: t_upd = update_time(t, start_index, end_index)
2: if (Stack is not empty) then
3:    if (value at top of Stack ≠ t_upd) then
4:       if (Stack.size = 2) then
5:          if (e is neighbor detection) then
6:             t_2 = Stack.pop()
7:             t_1 = Stack.pop()
8:             neighborDetected(m, t_1)
9:             neighborLeft(m, t_2)
10:         else if (e is neighbor leaving) then
11:            Stack.pop()
12:         end if
13:      end if
14:      Stack.push(t_upd)
15:   else
16:      if (Stack.size = 1) then
17:         Stack.pop()
18:      end if
19:   end if
20: else
21:   Stack.push(t_upd)
22: end if
```

main friends of node 21 in period 9pm-12am, node 20 is its only friend in period 12am-3am. However, if an aging mechanism were used, these four nodes would have still been considered good friends of node 21 in period 12am-3am because even though the corresponding link weights were decreasing, they were still high enough to indicate friendship. In Haggle traces, for some nodes, all of the contacts may be squeezed into 9am-6pm range of work hours. However, with a careful look, one can easily detect similar examples even within this time range (e.g. in Figure 6.12 (left), node 11 is the friend of node 39 only from 9am to 1pm.).

To be able to compute its friendship community for each period, a node i first needs to convert the time of its encounters to the local time of each period. Consider the upper graph in Figure 6.13 where a sample four day contact history between two nodes is illustrated. If three-hour ranges are used to define periods, the encounters within each specific three-hour period should be considered separately to analyze

Algorithm 10 update_time (time t, time $start\_index$, time end_index)

1: d=$\lfloor (t - end\_index)/86400 \rfloor$
2: $period\_length = end\_index - start\_index$
3: **if** (($t < start\_index$ + (d+1)× 86400) and (t>end_index+d×86400)) **then**
4: $t_{upd} = (d + 1) \times period\_length$
5: **else**
6: $t_{upd} = (t\%86400) - start\_index + (d + 1) \times period\_length$
7: return t_{upd}
8: **end if**

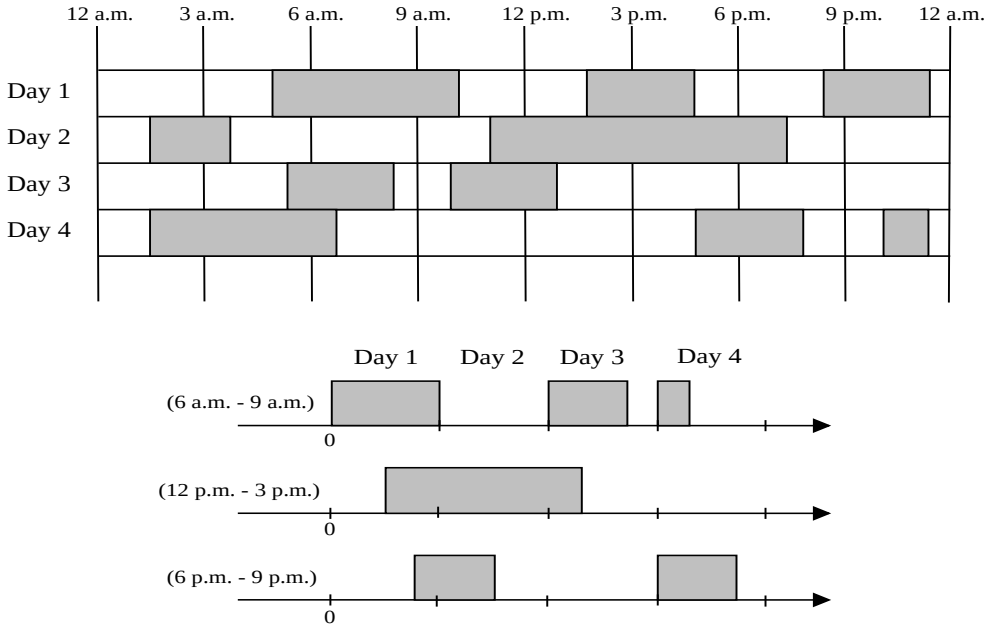


Figure 6.13: Sample contact history between two nodes (upper) and the updated contact history for three different periods (lower).

the friendship relations within the periods. Therefore, the encounter history has to be updated for each period as it is shown in the lower graph in Figure 6.13. As it is seen clearly, corresponding periods of the contact history in a day (global time) are concatenated to form the updated contact history of a period (local time). In Algorithm 9, we show how each period forms and maintains its own contact history as the events (neighbor detection or neighbor leaving) occur in global time. Once each node generates an event handler for each of its periods with given start and end indexes (e.g. start and end indexes for period (6 a.m.-9 a.m) are 21600s and 32400s), local contact history for each period can be generated following the steps

in Algorithm 9. First the global time of the event is converted to local time (in seconds) using the *update_time*() method (Algorithm 10), then with the help of a *stack*, the start and end times of contacts within the period are detected. Note that, as it is seen in lines 8-9 of Algorithm 9, when the event handler of a period notices that a full encounter according to the local clock of the period has finished, it calls $neighborDetected()$ (Algorithm 7) and $neighborLeft()$ (Algorithm 8) methods to compute the SPM and $RSPM$ values within the period (using encounter times according to local clock of the period).

Contact id	Global time of encounter (t_start, t_end)	Local time in (6 a.m.-9 a.m) period	Local time in (12 p.m.-3 p.m) period
1	(17800, 37000)	(**0**, **10800**)	(0, 0)
2	(48600, 61200)	(10800, 10800)	(**5400**, 10800)
3	(73800, 83700)	(10800, 10800)	(10800, 10800)
4	(91800, 100800)	(10800, 10800)	(10800, 10800)
5	(126000, 154800)	(21600, 21600)	(10800, 21600)
6	(191700, 203400)	(**21600**, **30600**)	(21600, 21600)
7	(207720, 219600)	(32400, 32400)	(21600, **25200**)
8	(264600, 283320)	(**32400**, **34920**)	(32400, 32400)
9	(318600, 331200)	(43200, 43200)	(43200, 43200)
10	(338400, 343800)	(43200, 43200)	(43200, 43200)

Table 6.1: Updated times (in seconds) of encounters in Figure 6.13 for two different periods. The bold values in local times show the start and end times of local encounters in corresponding period.

In Table 6.1, we give the list of the encounter times in Figure 6.13 according to both the global time (in seconds) and the local time of periods (6 a.m.-9 a.m) and (12 p.m.-3 p.m). The bold values in local times show the start and end times of local encounters (that trigger the run of $neighborDetected()$ and $neighborLeft()$) in corresponding period.

6.2.3 Forwarding Algorithm

After a node constructs its friendship community for each period based on its current encounter history, it decides whether to forward a message to the encoun-

tered node. If a node i having a message for node d meets with node j, it forwards the message to j if and only if node j's current friendship community (in the current period) includes[20] node d and node j is a stronger friend of node d than node i is. Accordingly, even if node j has a stronger link with node d than node i has, if node j does not include d in its current friendship community (i.e. weight of link between j and d is less than τ), node i will not forward the message to node j.

For an efficient routing, we also need to handle period boundary cases which arise when the encounter of two nodes is close to the end of the current period. In such a case, nodes use their friendship communities in the next period. For example, if we use three hour periods for community formation and node i meets node j at 2:45 p.m., it would be better if the nodes use their communities (so the link weights) in the next three hour period (3 p.m.-6 p.m.) to check whether the destination is included. Since the time remaining in the current period is very short, using the current communities may lead to inefficient forwarding decisions. In our algorithm, we use threshold t_b and let the nodes use next period's community information if remaining time to the end of current period is less than t_b.

6.2.4 Evaluations

6.2.4.1 Data Sets

For the evaluation of the proposed algorithm, we used three DTN traces in our simulator (see previous chapter for simulator details). In addition to the Haggle Project dataset and synthetic mobility traces that we introduced in simulations section of previous chapter, we also used MIT Reality Mining dataset [80] from crawdad archive [87]. MIT dataset consists of the traces of 97 Nokia 6600 smart phones which were carried by students and staff at MIT over nine months. In our simulations, we used the contacts logged during a three month period from the beginning of February to the end of April. This is the time of the second academic semester where human relations are relatively stable and participants are active on campus [79]. We also slightly changed the synthetic traces introduced in

[20]This is equivalent to checking whether either of direct or indirect weights between nodes j and d is larger than threshold τ. Considering this usage of friendship in the design of the forwarding algorithm, each node indeed does not need to keep and maintain a separate list of its friends. Friendship concept here is used to define possible good forwarders of a message.

previous chapter as follows. To make the movements of nodes more realistic, we also considered irregular movements of nodes. When a node finishes its visit in one of the communities in its list, it decides to visit a random community (other than the ones in its list) with probability p_r. After this irregular visit, the node then continues its community visits by moving to the next one on its list. The default values we used in the generation of synthetic traces are also changed as follows: N_c=15, N_p=100, V=5, [V_{min}, V_{max}]=[20, 120]m/min, [T_{min}, T_{max}]=[30, 120]min, p_r=0.1, R=30m.

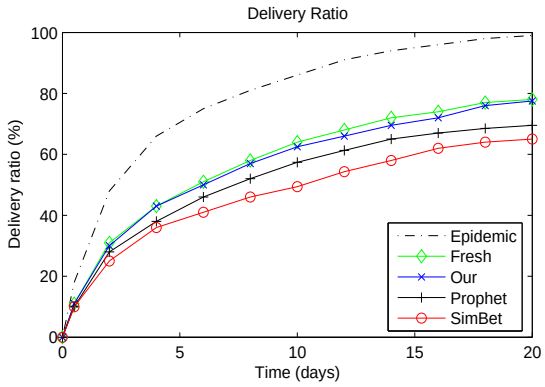
6.2.4.2 Algorithms in Comparison and Performance Metrics

In simulations, we compare the proposed routing algorithm with three other benchmark algorithms: Prophet [25], SimBet [68] and Fresh [64]. In Prophet, each node calculates its *delivery predictability* using its contact history along with transitivity and aging features and each node passes the carried packet if it meets a node with higher predicted delivery probability. In SimBet, each node calculates a simbet metric using two social measures (similarity and betweenness) and during the meetings, the messages are forwarded to encountered nodes with higher simbet metric. Finally, in Fresh [64], a node forwards a message to the encountered node only if the latter has a more recent meeting with destination node than itself. For Prophet and SimBet, we use the same parameters suggested in original studies [25, 68]. To show the optimal delivery ratio that could be achieved with current setting in the network, we also present the results of epidemic routing [22].

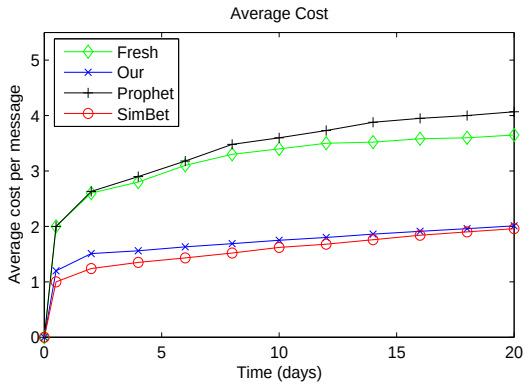
In evaluations, we also use the following three metrics: message delivery ratio, average cost, and routing efficiency. Delivery ratio is the proportion of messages that are delivered to their destinations among the total messages generated. Average cost is measured by the average number of forwards done per message during the simulation. Finally, routing efficiency [90] is defined as the ratio of delivery ratio to the average cost.

6.2.4.3 Simulation Results

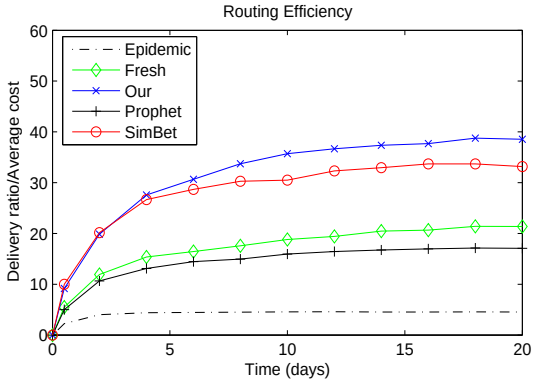
In the simulations, we used 1/5 of each data as warm up period, and let the nodes build their initial contact history. After the warm up period, we generated



(a) Delivery ratio vs. time



(b) Average cost vs. time



(c) Routing Efficiency vs. time

Figure 6.14: Comparison of algorithms using MIT traces

5000 messages, each from a random source node to a random destination node[21] every t seconds. In MIT traces, since the duration of experiment is long, we set t = 300s, but for Haggle and synthetic traces, we set t = 15s. All messages are assigned a Time-To-Live (TTL) value representing the maximum delay requirement. To form friendship communities, we used three hour periods[22] and set $\tau = 1/80$ min^{-1} and $t_b = 15$ min.

For main simulations, we assume that the nodes have enough buffer space to store every message they receive and the bandwidth is high enough to allow the

[21]In MIT traces, nodes that do not have any contacts with others in the selected three month period were not assigned as either source or destination to prevent meaningless messages.

[22]We used eight (equal-sized) three-hour ranges considering the possible behavior change of people in daily life (6 a.m. to 9 a.m. and 3 p.m. to 6 p.m. may be considered as commuting times etc.). However, for a general dataset, the number of periods and their lengths can be decided based on the density of encounters between different times. This will be the subject of our future work.

exchange of all messages between nodes at encounter times[23]. These assumptions are reasonable in view of capabilities of today's technology and are also used commonly in previous studies [85]. Any change in the current assumptions is expected to affect the performance of compared algorithms in the same way since they all use one copy of the message. Moreover, we used a simplified slotted CSMA MAC model as in [47]. We ran each simulation 10 times with different seeds and in each run, we collect statistics by running each algorithm on the same set of messages. All results plotted in figures show the averages of results obtained in such repeated runs.

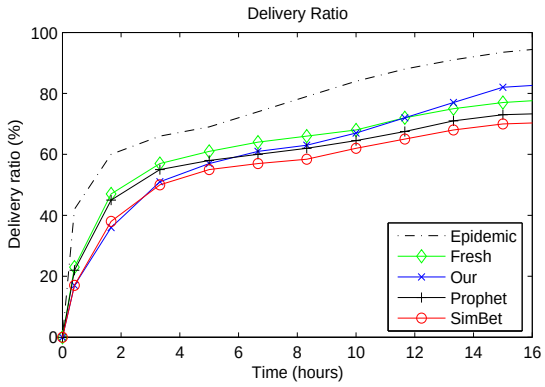
In Figure 6.14, we show comparison of all algorithms in terms of the three aforementioned metrics using MIT traces. Disregarding Epidemic routing, because of its unacceptable cost, our algorithm achieves the highest delivery ratio (78%, similar to Fresh) but it also has the minimum cost (similar to SimBet). Consequently, its efficiency is the best among all algorithms with a 16%, 80% and 125% improvement over Simbet, Fresh and Prophet, respectively.

In simulations with Haggle traces (Figure 6.15), our algorithm delivers 82% of all messages with the cost similar again to SimBet. As a result, it provides 30%, 180% and 420% improvement in the routing efficiency over SimBet, Fresh and Prophet, respectively. From Figure 6.16, we also observe the superiority of our algorithm on the results with synthetic traces. While our algorithm achieves almost 80% of delivery ratio, Fresh achieves only 60% and SimBet and Prophet deliver just 48% of all messages, while the average cost induced by our algorithm is only slightly higher than the cost of SimBet. Therefore, our algorithm achieves the best routing efficiency which is 33%, 47% and 650% higher than the routing efficiencies of SimBet, Fresh and Prophet, respectively.

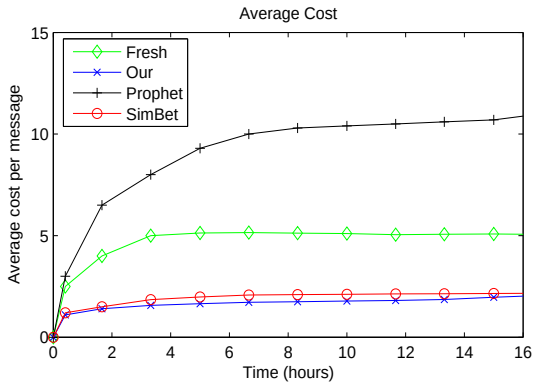
We also look at the effects of some parameters on the results. Since the effects are turned out to be similar on each data set, here we show the results with only Haggle dataset.

First, we look at the scenarios where the buffer space at each node is limited and FIFO buffer management scheme is used. With these assumptions, we computed the routing efficiency achieved in all algorithms with different buffer sizes in

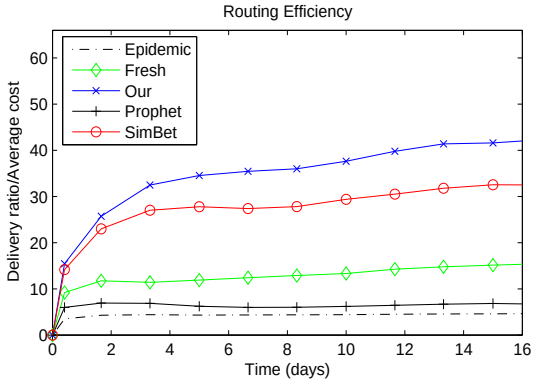
[23]We also performed simulations with limited resources and different values of parameters. We present these results at the end of the section.



(a) Delivery ratio vs. time



(b) Average cost vs. time

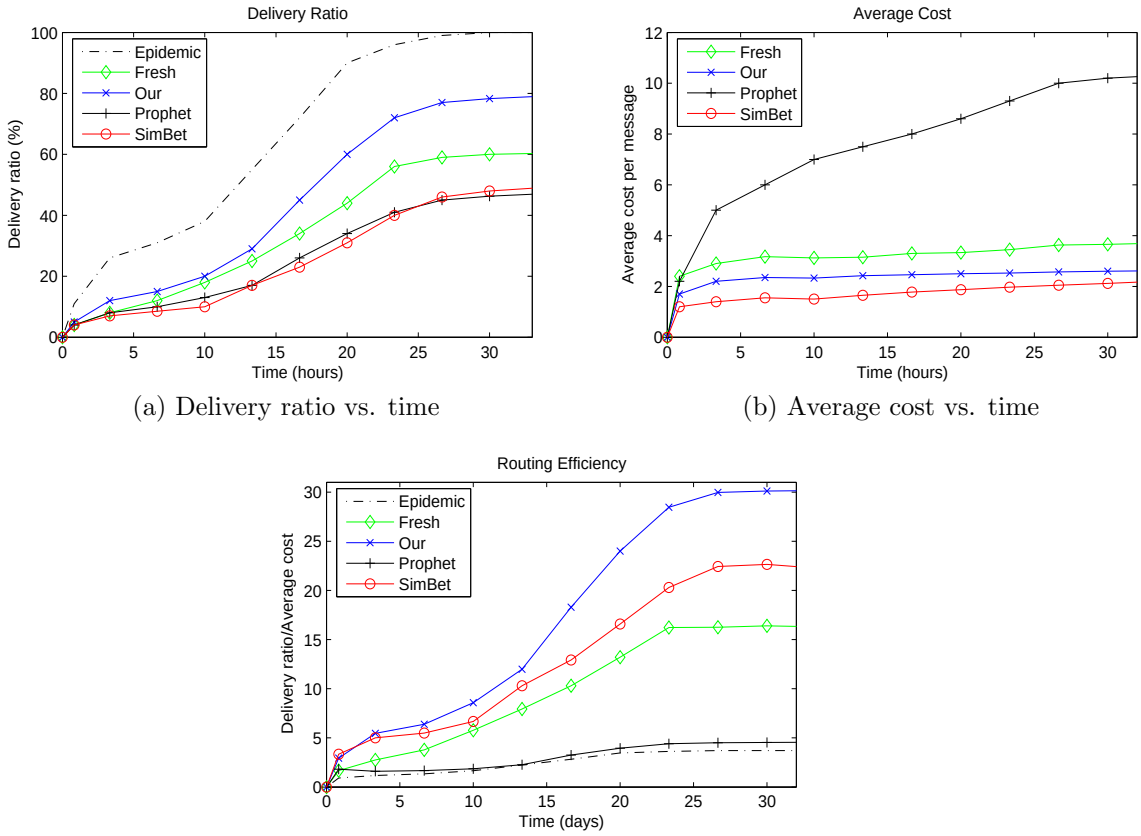


(c) Routing Efficiency vs. time

Figure 6.15: Comparison of algorithms using Haggle Project traces

the range of [10-50] messages. Figure 6.17 shows the results. For these simulations, we set the message generation interval t=12s and TTL=16.6 hours. The results show that routing efficiency of all algorithms increases as buffer space increases because messages are not dropped. Moreover, the routing efficiency of each algorithm converges to some constant value after sufficient buffer space is allocated. In Figure 6.18, we show the routing efficiency of all algorithms with different message generation intervals (when buffer size is 50 messages and TTL=16.6 hours). The results are similar to Figure 6.17, because as fewer messages are generated, fewer messages dropped due to buffer overflows, thus more messages could be delivered, increasing the routing efficiency.

The simulation results with different traces having different number of nodes, contact frequencies and durations and also the results with different values of sim-



(a) Delivery ratio vs. time

(b) Average cost vs. time

(c) Routing Efficiency vs. time

Figure 6.16: Comparison of algorithms using Synthetic traces

ulation parameters (buffer, message generation interval) show that our algorithm performs better than other algorithms over wide range of environments.

6.2.5 Discussions and Future Work

6.2.5.1 Complexity of the Algorithm

In the introduced algorithm, each node determines its friendship community in each period using mainly its own encounter history without much control message overhead. The only information that a node needs from its contacts is their RSPM values with its non-contact nodes (needed to find indirect close friends). However, this information is requested only from close friends of nodes and performed with messages of small size compared to data messages. In contrast, the compared algorithms impose significant control message overhead caused by exchange of summary

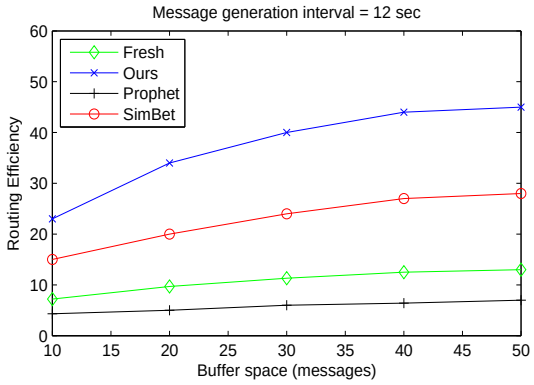


Figure 6.17: Routing efficiency vs. buffer space

vectors during contact times.

6.2.5.2 The Effects of Number of Periods and Thresholds

If we increase the number of periods that a day is divided into (thus the local friendship communities each node has), this may enable the nodes to make better forwarding decisions. On the other hand, the cost of computing the friendship communities (or link weights) in each period and also the space required to hold different communities will increase as well. However, as long as this cost could be handled and there is enough space at nodes, better results could be achieved. Moreover, the thresholds of the algorithm have also effects on its performance. As τ increases (decreases), friend lists of nodes get smaller (bigger), and as t_b changes, the forwarding algorithm may become more sensitive to period boundaries. In future work, we will look at these issues and try to find optimum values of τ and t_b as well as the optimal placement of period boundaries.

6.2.5.3 Extension of the Algorithm

We believe that the performance of the proposed algorithm can be improved by using transitive friendship behavior of different nodes in consecutive periods of the day. For example, assume that node i has a close friend j in period 12 p.m.-3 p.m., and node j has a close friend k in 3 p.m.-6 p.m. period. Then, when node s meets node i and has a message destined to node k in period 12 p.m.-3 p.m., it can forward this message to node i (even though node i has no direct or indirect close

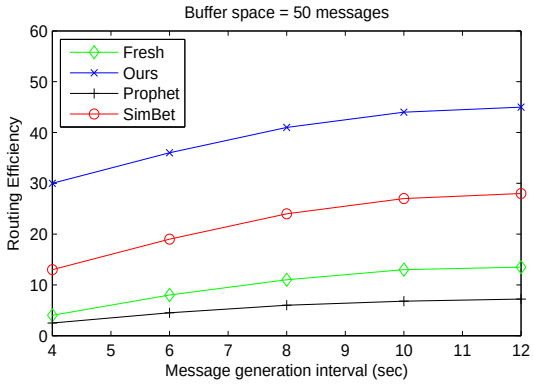


Figure 6.18: Routing efficiency vs. message generation interval

friendship with node k in the current period). This is because with high probability the message will be forwarded from i to j and then from j to k. However, such a solution will increase the algorithm's maintenance cost. We will study this issue in our future work and analyze the cost-benefit tradeoff.

6.3 Summary of Contributions

In this chapter, we looked at the effects of social relations between nodes in designing better DTN algorithms.

In the first part, we looked at the impact of social structure on Spray and Wait algorithm in a DTN where nodes move according to a random mobility model. We first proposed a new social network model illustrating the general picture of node meetings in community-based networks. Then we discussed the effects of distributing different number of copies to different communities on the performance of routing. We analytically calculated the expected delivery delay in a sample network scenario and validate the results with simulations. Furthermore, we also compared the minimum delay achieved when optimal L_{in} is used with the delay of normal spraying algorithm in which message copies are distributed without considering the underlying community structure in the network. We observed that considering the community structure (resulting in heterogeneous meeting behaviors among nodes) and distributing copies accordingly outperforms the normal spraying both in terms of average delivery delay and the average copy count used per message.

In the second part, we studied the routing problem in mobile social networks which is special kind of DTNs where the nodes are human-carried wireless devices. First, we introduced a new metric to detect friendship based node relations accurately. Then, we presented a new routing algorithm in which a node forwards its messages to those nodes that contain the destination node in their friendship communities. To reflect the periodic changes on node relations, our friendship communities depend on the period of day in which forwarding is done. We also treated indirect relations between nodes in a novel way making them amenable to routing. We evaluated the proposed algorithm through simulations using two real DTN traces and synthetic data. The results show that our algorithm performs better than three benchmark algorithms proposed previously.

CHAPTER 7
CONCLUSIONS AND DISCUSSIONS

This chapter provides a summary of this thesis work and discusses how the proposed algorithms in this thesis contributed to the field of DTN routing.

This thesis has made four major research contributions to DTN routing field, with its four different routing algorithms proposed for different DTN environments. In each, different techniques are utilized to achieve the same goal of obtaining high delivery rate, low delay and cost within a given time constraint or deadline.

In Chapter 3, we generalized the spray and wait [37] routing algorithm and proposed a multi-period spray and wait routing algorithm [50, 51, 52]. Rather than distributing all message copies at the beginning (as in [37]), we distribute predefined number of message copies at different spraying periods and wait for the delivery of any of them between spraying durations. By the end of each spraying period, if the delivery does not happened yet, we spray additional copies of the message to increase the probability of its delivery. With this idea of spraying message copies at different periods, we aimed to use the advantage of early delivery and achieve the same or a higher delivery rate by the deadline while using fewer average number of copies per message than the single period spray and wait algorithm [37]. Chapter 3 includes both the analytical and simulation results of this multi-period spray and wait algorithm where we clearly demonstrate that the proposed multi-period algorithm achieves better performance than single-period spray and wait routing algorithm.

Chapter 4 describes a different technique, erasure coding of messages, that is beneficial to increase the reliability and robustness of DTN routing. Beyond this benefit, in Chapter 4, we also show the benefit of erasure coding method in decreasing the cost of routing. By encoding the message at source node into multiple small message blocks and distributing them to other relay nodes, we increase the number of message carriers of the message, however, as we showed in the chapter, we can reach higher delivery rates before the replication based routing algorithms.

Therefore, we utilized this technique in routing of messages and worked on how we can reduce the routing cost both in a single period and multiple periods [53]. To this end, we found optimum parameters (number of encoded blocks that need to be distibuted at each period, the replication factor etc.) that minimize the routing cost in the current setting and used them in routing of messages. In this chapter, we also analyzed the effects of message distribution algorithms on the cost of routing both in replication based and erasure coding based algorithms. By analytical findings and simulation results we demonstrated that even though binary spraying is the correct strategy in replication based routing (since the message copies are exactly same with each other), in erasure coding based routing, since the message contents are different than each other, source spraying is the best strategy. We believe that analyzing the effect of message distribution schemes on different routing types is another significant contribution of this chapter to the literature.

In the next two chapters, we looked at the routing problem from different perspectives. The research that we presented in these chapters are initiated with our analysis on real DTN traces. We notice that the movements of nodes in a real DTN show heterogeneous behavior and they need to be carefully analyzed to be able to develop better routing algorithms.

In Chapter 5, we focused on the correlation between the movements of different nodes. To detect the presence of these correlations between the nodes in a real DTN, we proposed a new metric called conditional intermeeting time and computed this value for each pair of nodes in available real DTN traces. We noticed that these correlations are very clear for many node pairs. Therefore, we used the correlations between the meetings of a node with other nodes for making the existing single copy based routing algorithms more cost efficient. That is, for shortest path based DTN routing algorithms, we proposed to use conditional intermeeting times rather than standard intermeeting times and route the messages over conditional shortest paths (CSP) [72]. Moreover, for the algorithms which make message forwarding decisions depending on a delivery metric, we proposed to use conditional intermeeting time as a secondary delivery metric and allowed the forwarding of messages if and only if both the algorithm's original delivery metric and conditional intermeeting time

agree to forward the message to the encountered node [73, 74]. With the simulation results, where we compared the original algorithms with their revised versions, we detected two different results. If the repetitive motion of nodes is clearly observed in a trace dataset, the benefit of using conditional intermeeting times in metric based algorithms becomes more pronounced. However, even in the environments where this is not the case, average cost of routing can still be decreased and routing efficiency can be improved remarkably. Consequently, in either case, by utilizing the correlation between the movements of nodes, we can increase the performance of single copy based routing algorithms.

Finally, in Chapter 6, we studied the routing problem in mobile social networks which are special kind of DTNs where the nodes are human carried devices. Since in these networks, the contacts between nodes depend on the social relations between the people who carry these nodes, we exploited social network properties of these nodes to understand the interactions between different nodes and to design better routing algorithms. In the chapter, we first looked at the spray and wait routing and analyzed the impact of the grouping behavior of nodes on its performance [75]. We showed that by carefully selecting the number of copies that must be distributed to inner community and inter community nodes, we can increase the performance of spray and wait based routing. Second, we proposed a friendship based routing algorithm [70, 71] in which we use a new social network metric to detect the direct and indirect relations between nodes more accurately and use these extracted social relations between nodes to make better routing decisions. By extensive simulations on both real and synthetic traces, we proved the better performance of proposed algorithm over the existing algorithms.

Throughout each chapter of the thesis, we discussed the future work that we plan to do regarding the work of that chapter. Besides these plans, we also would like to work on the following issues. First of all, we would like to extend the performance of the proposed algorithms by utilizing multicasting [96, 97]. Then, we would like to focus on the security [99, 100, 101] and privacy preserving [98, 102] issues in routing of messages in DTNs, because we believe that the nature of DTNs is very open to different attacks, and the routing of messages can easily fail due to these attacks.

Also, we would like to combine the proposed DTN routing protocols with the service discovery protocols [55] that we developed for DTNs to increase the performance of routing.

LITERATURE CITED

[1] A. Doria, M. Uden, and D. P. Pandey, *Providing connectivity to the saami nomadic community*, in Proceedings of the 2nd International Conference on Open Collaborative Design for Sustainable Innovation (dyd 02), Bangalore, India, Dec 2002.

[2] A. Pentland, R. Fletcher, and A. A. Hasson, *A road to universal broadband connectivity*, in Proceedings of the 2nd International Conference on Open Collaborative Design for Sustainable Innovation (dyd 02), Bangalore, India, Dec 2002.

[3] G. E. Prescott, S. A. Smith, and K. Moe, *Real-time information system technology challenges for NASAs earth science enterprise*, in Proceedings of The 20th IEEE Real-Time Systems Symposium, Phoenix, Arizona, Dec 1999.

[4] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, *Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet*, in Proceedings of ACM ASPLOS, 2002.

[5] *Disruption tolerant networking*, http://www.darpa.mil/ato/solicit/DTN/.

[6] J. Ott and D. Kutscher, *A disconnection-tolerant transport for drive-thru internet environments*, in Proceedings of IEEE INFOCOM, 2005.

[7] G. W. Boehlert, D. P. Costa, D. E. Crocker, P. Green, T. OBrien, S. Levitus, and B. J. Le Boeuf, *Autonomous pinniped environmental samplers; using instrumented animals as oceanographic data collectors*, Journal of Atmospheric and Oceanic Technology, vol. 18, no. 11, pp. 18821893, 2001, 18 (11).

[8] T. Small and Z. Haas, *The shared wireless infostation model - a new ad hoc networking paradigm (or where there is a whale, there is a way)*, in Proceedings of The Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003), June 2003, pp. 233244.

[9] K. Fall, *A Delay-Tolerant Network Architecture for Challenged Internets*, SIGCOMM, August 2003.

[10] A. Beaufour, M. Leopold, and P. Bonnet, *Smart-tag based data dissemination*, in First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA02), June 2002.

[11] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, *Epidemic algorithms for replicated database*

maintenance, in Proceedings of the ACM Symposium on Principles of Distributed Computing, 1987, pp. 112.

[12] W. Vogels, R. V. Renesse, and K. Birman, *The power of epidemics: Robust communication for large-scale distributed systems*, In Proceedings of HotNets-I '02: First Workshop on Hot Topics in Networks, special issue of the ACM SIGCOMM Computer Communication Review, Princeton, NJ. October 2002.

[13] *Delay tolerant networking research group*, http://www.dtnrg.org.

[14] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi, *Hardware design experiences in zebranet*, In Proc. ACM SenSys, pages 227238, 2004.

[15] M. Motani, V. Srinivasan, and P. Nuggehalli, *PeopleNet: Engineering a Wireless Virtual Social Network*, In Proc. ACM Mobicom, pages 243257, Aug. 2005.

[16] J. Partan, J. Kurose, and B. N. Levine, *A Survey of Practical Issues in Underwater Networks*, In Proc. ACM WUWNet, pages 1724, Sept. 2006.

[17] A. Maffei, K. Fall, and D. Chayes, *Ocean Instrument Internet*, In Proc. AGU Ocean Sciences Conf., Feb 2006.

[18] Z. Zhang, *Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges*, Communications Surveys & Tutorials, IEEE, vol.8, pp. 24-37, 2006

[19] A. Balasubramanian, B. N. Levine, A. Venkataramani, *DTN routing as a resource allocation problem*, ACM SIGCOMM, 2007.

[20] 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 Proceedings of ACM Annual Intl. Conf. on Mobile Computing and Networking (Mobicom), 2007.

[21] J. Widmer and J. Boudec, *Network coding for efficient communication in extreme networks*, in Proceeding of the ACM SIGCOMM workshop on Delay Tolerant Networking (WDTN), 2005

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

[23] X. Zhang, G. Neglia, J. Kurose, D. Towsley, *Performance Modeling of Epidemic Routing*, Computer Networks, Vol. 51/10 (2007), pp. 2859-2891.

[24] A. Jindal and K. Psounis, *Performance analysis of epidemic routing under contention*, in Proceedings of Workshop on Delay Tolerant Mobile Networking (DTMN) held in conjunction with IWCMC, 2006.

[25] A. Lindgren, A. Doria, and O. Schelen, *Probabilistic routing in intermittently connected networks*, SIGMOBILE Mobile Computing and Communication Review, vol. 7, no. 3, 2003.

[26] Y. Wang, S. Jain, M. Martonosi, and K. Fall, *Erasure-coding based routing for opportunistic networks*, in Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, 2005.

[27] S. Jain, M. Demmer, R. Patra, and K. Fall, *Using redundancy to cope with failures in a delay tolerant network*, SIGCOMM Comput. Commun. Rev., 2005.

[28] Y. Liao, K. Tan, Z. Zhang, and L. Gao, *Estimation based erasure-coding routing in delay tolerant networks*, in Proceedings of the 2006 international conference on Wireless communications and mobile computing, 2006.

[29] Y. Liao, Z. Zhang, B. Ryu, Bo and L. Gao, *Cooperative robust forwarding scheme in DTNs using erasure coding*, Military Communications Conference (MILCOM), 2007.

[30] L. Chen, C. Yu, T. Sun, Y. Chen and H. Chu, *A hybrid routing approach for opportunistic networks*, in Proceedings of the 2006 SIGCOMM workshop on Challenged networks (CHANTS), 2006.

[31] Z. Li, L. Sun, E. C. Ifeachor, *Adaptive Multi-Copy Routing for Intermittently Connected Mobile Ad Hoc Networks* in Proceedings of GLOBECOM, 2006.

[32] C. Liu, and J. Wu, *Routing in a cyclic mobispace*, in Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc), 2008.

[33] T. Small and Z. Haas, *Resource and performance tradeoffs in delay tolerant wireless networks*, in Proceedings of ACM SIGCOMM workshop on Delay Tolerant Networking (WDTN), 2005.

[34] W. Zhao, M. Ammar, and E. Zegura, *A message ferrying approach for data delivery in sparse mobile ad hoc networks*, In Proceedings of MobiHoc04, May 2004.

[35] K. Harras, K. Almeroth, and E. Belding-Royer, *Delay Tolerant Mobile Networks (DTMNs): Controlled Flooding Schemes in Sparse Mobile Networks*, In IFIP Networking, Waterloo, Canada, May 2005.

[36] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, *MaxProp: Routing for Vehicle-Based Disruption- Tolerant Networks*, In Proc. IEEE Infocom, April 2006.

[37] T. Spyropoulos, K. Psounis,C. S. Raghavendra, *Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks*, ACM SIGCOMM Workshop, 2005.

[38] M. Musolesi, S. Hailes and C. Mascolo, *Adaptive routing for intermittently connected mobile ad hoc networks*, in Proceedings of WoWMoM 2005, pp. 183-189, 2005.

[39] Z. Li, L. Sun and E. Ifeachor, *Adaptive Multi-Copy Routing for Intermittently Connected Mobile Ad Hoc Networks*, in Proceedings of IEEE Globecom 2006, 27 Nov - 1 Dec 2006, San Francisco, USA.

[40] T. Camp, J. Boleng, and V. Davies, *A Survey of Mobility Models for Ad Hoc Network Research*, Wireless Communication & Mobile Computing (WCMC), Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp 483-502, 2002.

[41] A. P. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri, *Toward realistic mobility models for mobile ad hoc networks*, in Proceedings of ACM MOBICOM, pp. 217229, San Diego, CA, Sep. 2003.

[42] A. Jindal, K. Psounis, *Fundamental Mobility Properties for Realistic Performance Analysis of Intermittently Connected Mobile Networks*, PerCom Workshops, 2007.

[43] J. W. Byers, M. Luby and M. Mitzenmacher, *Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads*, in Proceedings of INFOCOM, 1999.

[44] T. Spyropoulos, K. Psounis,C. S. Raghavendra, *Performance Analysis of Mobility-assisted Routing*, MobiHoc, 2006.

[45] P. U. Tournoux, J. Leguay, F. Benbadis, V. Conan, M. Amorim, J. Whitbeck, *The Accordion Phenomenon: Analysis, Characterization, and Impact on DTN Routing*, in Proceedings of Infocom, 2009.

[46] T. Spyropoulos, K. Psounis,C. S. Raghavendra, *Efficient routing in intermittently connected mobile networks: The single-copy case*, IEEE/ACM Transactions on Networking, vol. 16, no. 1, Feb. 2008

[47] T. Spyropoulos, K. Psounis,C. S. Raghavendra, *Efficient Routing in Intermittently Connected Mobile Networks: The Multiple-copy Case*, IEEE/ACM Transactions on Networking, 2008.

[48] P. Hui, J. Crowcroft, and E. Yoneki, *BUBBLE Rap: Social Based Forwarding in Delay Tolerant Networks*, 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), HongKong, May, 2008.

[49] E. Yoneki, P. Hui, S. Chan and J. Crowcroft, *A Socio-Aware Overlay for Publish/Subscribe Communication in Delay Tolerant Networks*, 10th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Crete, Greece, October, 2007.

[50] E. Bulut, Z. Wang, B. Szymanski *Time Dependent Message Spraying for Routing in Intermittently Connected Networks*, in Proceedings of Globecom 08, New Orleans, November 2008.

[51] E. Bulut, Z. Wang, B. Szymanski, *Cost Effective Multi-Period Spraying for Routing in Delay Tolerant Networks*, in IEEE/ACM Transactions on Networking, vol. 18, 2010.

[52] E. Bulut, Z. Wang, B. Szymanski, *Minimizing Average Spraying Cost for Routing in Delay Tolerant Networks*, in Proceedings of Second Annual Conference of International Technology Alliance (ITA), ACITA, Imperial College London, September 2008.

[53] E. Bulut, Z. Wang, B. Szymanski *Cost Efficient Erasure Coding based Routing in Delay Tolerant Networks*, in Proceedings of ICC 2010, South Africa.

[54] S. C. Geyik, E. Bulut, B. Szymanski, *PCFG based Synthetic Mobility Trace Generation*, in Proceedings of Globecom 2010, Miami, December 2010.

[55] Z. Wang, E. Bulut, B. Szymanski, *Service Discovery for Delay Tolerant Networks*, in workshop on Heterogeneous, Multi-Hop, Wireless and Mobile Networks (HeterWMN), in conjunction with Globecom 2010, Miami, December 2010.

[56] E. Daly and M. Haahr, *Social network analysis for routing in disconnected delay-tolerant manets*, In Proceedings of ACM MobiHoc, 2007.

[57] S. C. Geyik and B. K. Szymanski, *Event Recognition in Sensor Networks by Means of Grammatical Inference*, In Proceedings of INFOCOM 2009, April, Brazil, 2009.

[58] D. Bertsekas and R. Gallager, *Data networks (2nd ed.)*, 1992.

[59] S. Srinivasa and S. Krishnamurthy, *CREST: An Opportunistic Forwarding Protocol Based on Conditional Residual Time*, in Proceedings of SECON, 2009.

[60] I. Psaras, L. Wood and R. Tafazolli, *Delay-/Disruption-Tolerant Networking: State of the Art and Future Challenges*, Technical Report, University of Surrey, UK.

[61] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, *Pocket Switched Networks and human mobility in conference environments*, in Proceedings of the ACM SIGCOMM 2005 Workshop on Delay-Tolerant Networking (W-DTN05), 2005.

[62] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, *Impact of Human Mobility on the Design of Opportunistic Forwarding Algorithms*, in Proceedings of INFOCOM, 2006.

[63] T. Karagiannis, J. Boudec, and M. Vojnovic, *Power Law and Exponential Decay of Inter Contact Times Between Mobile Devices*, in Proceedings of MobiCom, 2007.

[64] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli, *Age Matters: Efficient Route Discovery in Mobile Ad Hoc Networks Using Encounter Ages*, In Proceedings of ACM MobiHoc, 2003.

[65] M. Grossglauser and M. Vetterli, *Locating Nodes with Ease: Last Encounter Routing in Ad Hoc Networks through Mobility Diffusion*, In Proceedings of IEEE INFOCOM, 2003.

[66] T. Spyropoulos, K. Psounis, and C. Raghavendra, *Spray and Focus: Efficient Mobility-Assisted Routing for Heterogeneous and Correlated Mobility*, In Proceedings of IEEE PerCom, 2007.

[67] T. Spyropoulos, T. Turletti, and K. Obrazcka, *Routing in Delay Tolerant Networks Comprising Heterogeneous Populations of Nodes*, IEEE Transaction on Mobile Computing, Vol. 8, No. 8, Aug. 2009.

[68] E. Daly and M. Haahr, *Social network analysis for routing in disconnected delay-tolerant manets*, In Proceedings of ACM MobiHoc, 2007.

[69] J. Link, N. Viol, A. Goliath and K. Wehrle, *SimBetAge: utilizing temporal changes in social networks for pocket switched networks*, Proc ACM Workshop on User-provided Networking, Rome, Italy, 2009.

[70] E. Bulut, B. Szymanski, *Friendship based Routing in Delay Tolerant Mobile Social Networks*, in Proceedings of Globecom 2010, Miami, December 2010.

[71] E. Bulut, B. Szymanski, *Exploiting Friendship Relations for Efficient Routing in Delay Tolerant Mobile Social Networks*, submitted to Elsevier Ad hoc Networks Journal - Special Issue on Social-Based Routing in Mobile and Delay-Tolerant Networks, 2010.

[72] E. Bulut, S. C. Geyik, B. Szymanski, *Conditional Shortest Path Routing in Delay Tolerant Networks*, in Proceedings of WoWMoM 2010.

[73] E. Bulut, S. Geyik and B. Szymanski, *Efficient Routing in Delay Tolerant Networks with Correlated Node Mobility*, in Proceedings of 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), Nov, 2010.

[74] E. Bulut, S. Geyik, B. Szymanski, *Efficient Routing in DTNs with Correlated Node Mobility*, submitted to IEEE Transactions on Mobile Computing (TMC), 2010.

[75] E. Bulut, Z. Wang, B. Szymanski, *Impact of Social Networks on Delay Tolerant Routing*, in Proceedings of IEEE Global Telecommunications Conference (GLOBECOM), pp.1-6, Nov. 30-Dec. 4, Honolulu, HI, 2009.

[76] F. Li and J. Wu, *LocalCom: a community-based epidemic forwarding scheme in disruption-tolerant networks*, in Proceedings of SECON, p.574-582, June 22-26, 2009.

[77] T. Zhou, R. R. Choudhury, K. Chakrabarty, *Diverse Routing: Exploiting Social Behavior for Routing in Delay-Tolerant Networks*, Pro. Conf. Computational Science and Engineering, Canada, 2009.

[78] Q. Li, S. Zhu and G. Cao, *Routing in Selfish Delay Tolerant Networks*, Proc. IEEE Infocom 2010.

[79] P. Hui and J. Crowcroft, *Predictability of Human Mobility and Its Impact on Forwarding*, Communications and Networking in China, 2008.

[80] N. Eagle, A. Pentland, and D. Lazer, *Inferring Social Network Structure using Mobile Phone Data*, Proc. National Academy of Sciences, 106(36), pp. 15274-15278, 2009.

[81] C. liu, J. Wu and I. Cardei *Message Forwarding in Cyclic Mobispace: the Multi-copy case*, In Proceedings of MASS, 2009.

[82] S. Jain, K. Fall, and R. Patra, *Routing in a delay tolerant network*, in Proceedings of ACM SIGCOMM, Aug. 2004.

[83] E. P. C. Jones, L. Li, and P. A. S. Ward, *Practical routing in delay tolerant networks*, in Proceedings of ACM SIGCOMM workshop on Delay Tolerant Networking (WDTN), 2005.

[84] J. Leguay, T. Friedman, and V. Conan, *DTN Routing in a Mobility Pattern Space*, In Proceedings of ACM WDTN, 2005.

[85] C. Liu and J. Wu, *An Optimal Probabilistically Forwarding Protocol in Delay Tolerant Networks*, in Proceedings of MobiHoc, 2009.

[86] J. Leguay, A. Lindgren, J. Scott, T. Friedman, J. Crowcroft and P. Hui, *CRAWDAD data set upmc/content (v. 2006-11-17)*, downloaded from http://crawdad.cs.dartmouth.edu/upmc/content, 2006.

[87] CRAWDAD data set, *http://crawdad.cs.dartmouth.edu.*.

[88] Y. Wang, P. Zhang, T. Liu, C. Sadler and M. Martonosi, *http://crawdad.cs.dartmouth.edu/princeton/zebranet*, CRAWDAD data set princeton/zebranet (v. 2007-02-14), 2007.

[89] A European Union funded project in Situated and Autonomic Communications, *www.haggleproject.org*.

[90] J. M. Pujol, A. L. Toledo, and P. Rodriguez, *Fair routing in delay tolerant networks*, Proc. IEEE INFOCOM, 2009.

[91] C. Mascolo and M. Musolesi, *CAR: Context-aware Adaptive Routing for Delay Tolerant Mobile Networks*, in IEEE Transactions on Mobile Computing. Vol. 8(2). pp. 246-260. February 2009.

[92] A. Balasubramanian, B. N. Levine, A. Venkataramani, *Replication Routing in DTNs: A Resource Allocation Approach*, IEEE Transactions on Networking, Vol. 18, No. 2. (April 2010), pp. 596-609.

[93] Y. Wang and H. Wu, *Delay/Fault-Tolerant Mobile Sensor Network (DFT-MSN): A New Paradigm for Pervasive Information Gathering*, IEEE Transactions on Mobile Computing, vol. 6, no. 9, pp. 10211034, 2007.

[94] C. Chen and Z. Chen, *Exploiting Contact Spatial Dependency for Opportunistic Message Forwarding*, in IEEE Transactions on Mobile Computing, vol. 8, no.10, October 2009.

[95] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot, *Delegation forwarding*, in Proceedings of ACM MobiHoc, pp. 251260, 2008.

[96] Y. Wang, X. Li, and J. Wu, *Multicasting in delay tolerant networks: delegation forwarding*, in Proceedings of IEEE Global Communications Conference (GLOBECOM), 2010.

[97] W. Gao, Q. Li, B. Zhao, and G. Cao, *Multicasting in delay tolerant networks: a social network perspective*, in Proceedings of ACM MobiHoc, pp. 299308, 2009.

[98] R. Lu, X. Lin, and X. Shen, *SPRING: A Social-based Privacy-preserving Packet Forwarding Protocol for Vehicular Delay Tolerant Networks*, The 29th IEEE International Conference on Computer Communications (INFOCOM 2010), San Diego, California, USA, March 14- 19, 2010.

[99] Y. Ren, M. C. Chuah, J. Yang, Y. Chen, *Detecting Wormhole Attacks in Delay Tolerant Networks*, in IEEE Wireless Communications Magazine, special issue on Security & Privacy, October 2010.

[100] N. Li and S. K. Das, *RADON: Reputation-Assisted Data forwarding in Opportunistic Network*, in Proceedings of the Second International Workshop on Mobile Opportunistic Networking (Mobiopp 2010), pp 8-14, Feb 22-23, 2010.

[101] S. C. Nelson, M. Bakht and R. Kravets, *Encounter-based Routing in DTNs*, in Proceedings of IEEE Infocom, Rio De Janeiro, Brazil, pp.846-854, Apr. 2009.

[102] U. Shevade, H. Song, L. Qiu, and Y. Zhang, *Incentive-Aware Routing in DTNs*, in Proceedings of IEEE International Conference on Network Protocols, Orlando, FL, USA, Oct. 2008.