IMPROVED SOCIALLY SELFISH ROUTING FOR DELAY TOLERANT NETWORKS

by

YAN LI

A thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfillment of

the requirements for the degree of

MASTER OF COMPUTER SCIENCE

School of Computer Science

at

CARLETON UNIVERSITY

Ottawa, Ontario

December 15, 2011

Your file Votre référence
ISBN: 978-0-494-91497-7

Our file Notre référence
ISBN: 978-0-494-91497-7

To Mom and Dad, my wife Qi

ACKNOWLEDGEMENTS

I am grateful to my supervisor, Professor Evangelos Kranakis for his support in the completion of my thesis. He gave me freedom to think about the problem, shared his research insight and provided encouragement. The completion of this work would have not been possible without his enthusiasm, guidance and feedback. Finally, I must thank my wife, Qi, who endured these years of graduate studies and contributed to my work.

ABSTRACT

A delay tolerant network (DTN) is a network overlay on top of regional networks. It is constructed based on an abstraction of message switching capabilities with limited expectations of end-to-end connectivity and node resources. In the real world, social selfishness will definitely affect the nodes preferences according to the properties of social ties in the DTNs since most people are selfish. The purpose of this study is to propose a new forwarding routing architecture to solve three problems: a) efficiently select a forwarding object, b) understand next hop's forwarding strategy and c) intelligently maintain the message's priority. This design is to optimize the forwarding strategy in order to achieve best performance and also maintain reasonable social selfishness.

- We construct our architecture with reasonable assumptions such as a bidirectional model which completely takes the next hop's social selfishness into consideration and then avoids blind forwarding.
- We propose new algorithms finding a balance point between the achievement of maximum system performance and social selfishness.
- Our simulation results demonstrate that our routing achieves better performance and more efficient usage of system resources even for higher workload.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BM	Buffer Manager
CDP	Counter Dropping Probability
DPE	Delivery Probability Estimator
DTN	Delay Tolerant Network
EDP	Expiration Dropping Probability
FSM	Forwarding Set Manager
ICMN	Intermittently Connected Mobile Network
ISSR	Improved Socially Selfish Routing
MANETs	Mobile Ad hoc Networks
MKPAR	Multiple-knapsack problem
MNs	Mobile Nodes
MSC	Mobile Switching Center
OCP	Open-closed Principle
OSM	Object Selecting Manager
PC	PeopleNet Coordinator
PDM	Packet Dropping Manager
PPM	Packet Priority Manager

PSN	Pocket Switched Network
SPR	Single Responsibility Principle
$SSAR$	Social Selfishness Aware Routing
$TFIDF$	Term Frequency Inverse Document Frequency
TTL	Time to Live

CHAPTER 1

INTRODUCTION

In the last decade, network architectures with characteristics such as the existence of end-to-end data paths, inexpensive round-trip time, reliable data transmission, and small packet drop probability no longer hold for communicating outside of the internet – where power limited nodes are mobile and connectivity among them changes quickly. This means if a graph is constructed by these nodes according to their radio detecting ranges as well as their time-based activities, there is high likelihood that each node behaves as a highly complex entity. Obviously, traditional routing protocols cannot be directly implemented in such a network. Delay tolerant network (DTN), sometimes also called Intermittently-Connected Mobile Network (ICMN), is an overlay on top of regional networks [1]. The primary goal in such a network is to achieve operations in the environments with properties of long delay paths and frequent network partitions.

A wide variety of networks fall under this category, revealing many challenges that are not present in traditional networks [2] [3] [4]:

- Routing: Routing is no doubt the most challenging issue within a DTN. Restrictions of end-to-end path in DTN determine that the potential available links could not

be predicted from any previous knowledge and the destination node may never be reachable immediately.

- Resources: Unstable routing results in messages stored in the device buffer for long periods of time. Accumulated messages require huge buffer space in order to take advantage of future communication opportunities.

- Power: Nodes in DTNs consume energy for movement, message storage and delivery as well as some computations. Routing strategies should consider devices capabilities for saving and optimizing power when making decisions.

- Security: Conventional methods are inefficient for user authentication, forwarding permission authorization and message confidentiality in DTNs due to the long round trip time on the available path.

For a sparse DTN, mobility-assisted routing is based on the store-carry-and-forward method [1]. Several routing algorithms have been introduced according to different techniques [5] [6]. Some other researches focus on the problem of power [7].

1.1 Motivation

Previously mentioned implementations assume that nodes are willing to forward packets for others, which may not reflect status in the real world. People in society may have different behaviours according to their social ties with different objects. People are usually willing to help others, if they got help from them in the past or will probably get help from them. In recent studies, socially selfish behaviour of nodes on routing is taken into consideration [8] [9]. However the study of selfishness affecting routing strategy in DTNs is still in its infancy. Some important factors such as message priority maintenance are not considered. The purpose of this thesis is to design a routing applied in social selfishness aware DTNs for achieving better performance and also maintaining social selfishness at a reasonable level. Our routing solves the following three questions:

1. Efficient selection of forwarding object (node): Since nodes are mobile and their activities are time-based, there is a low possibility that a node could communicate with several nodes one by one if it detects these nodes within its radio range. This leads to a question for the node how to find a most suitable forwarding object.

2. Understanding of the next hop's forwarding strategy: In previous studies, a node determines the delivery list of messages according to the relationship between next hop and itself. The relationship between next hop and destination is never considered. But in real world, the intermediate node may drop the message or deliberately delay delivery if it has weak social ties with the corresponding destination. To avoid such a result, next hop's forwarding strategy must be carefully considered.

3. Intelligent maintenance of the message's priority: The priority of a message will be decreased when this message traverses several hops. However, the rate of priority decrease should be different for different messages. If a message has high importance, its priority decrease rate is small. On the other hand, if a message has low priority its decrease rate is large. This manifests the real world situation how people deal with imperative messages and maintain importance of messages by different ways.

Each question will be deeply discussed in Section 5.1. In our work, we propose an improved routing strategy taking three issues into consideration in order to achieve better routing performance with higher usage of system resources and also behave well under high workload. Since our routing is designed for social selfishness aware DTNs, we name it improved socially selfish routing.

1.2 Contributions

In our work, we study the importance of social ties and social selfishness in DTNs as well as some forwarding strategies implemented for DTNs. We state three key problems which are

not considered in other routings. We construct our model based on reasonable assumptions (in Chapter 5) and make the following contributions to these problems:

1. Our strategy finds a balance point between the achievement of maximum system performance and social selfishness when a node selects its forwarding object.

2. Our bidirectional model takes the next hop's willingness to forward a message to its destination into consideration. This will avoid undesirable forwarding and increase the message delivery ratio.

3. We rebuild the formula for message's priority calculation for the purpose of intelligent maintenance of messages' priority.

The performance of our routing is evaluated and the results demonstrate that our routing achieves better performance with different messages' TTL (time to live), and has higher usage of system resources. It also behaves well under high workload.

1.3 Organization of Thesis

Chapter 2 presents the difference between conventional networks and DTNs in more detail, the basic concepts of strength of social ties and social selfishness in DTNs. Chapter 3 outlines two existing models acting as DTNs, including PeopleNet and Pocket Switched Networks. Chapter 4 reviews routings implementing data delivery in DTNs; classifies them into several groups and compares the performance difference between social selfish algorithm and social selfless algorithm. Chapter 5 details the architecture of our improved routing strategy and how it could solve the listed problems. Chapter 6 describes the simulation environments and implementations. Chapter 7 discusses the simulation result and evaluates the performance of our new routing. Chapter 8 concludes the thesis and illustrates possible directions for future work.

CHAPTER 2

BACKGROUND

In this chapter, we discuss the difference between DTNs and traditional networks in detail; introduce the basic idea of social ties and methods measuring the strength of social ties, and finally present the importance of social selfishness in DTNs.

2.1 Delay Tolerant Network

The internet has been a great success at interconnecting communication devices across the globe. TCP/IP is the mostly used protocol suite for internet applications. It provides a principle of end-to-end inter-process connection. To achieve the overall performance for smooth data switch, some key assumptions are made [10] [11]:

- There is an end-to-end data path between source node and target node
- The maximum round-trip time between any node pairs in the network is not excessive
- The end-to-end packet drop probability is small
- Communication links have relatively symmetric bidirectional data rates

However, there are a wide variety of networks where an end-to-end connection between a given source and destination pair may never be present. The students' movement in university campus [12] and buses traveling among five colleges [13] act as relays to assist in the data delivery and collection. They are representative examples of DTNs. A wide variety of networks fall under DTNs range from terrestrial mobile networks, Satellite-based Networks, sensor/actuator networks to military Ad-Hoc networks [3]:

- Terrestrial Mobile Networks: are partitioned due to node mobility and may never have an end-to-end path.
- Satellite-based Networks: usually have very long-distance links with high latencies (such as near-Earth satellite communications) by predicable interruption and suffer outages due to environmental conditions.
- Sensor/Actuator Networks: have limited capabilities such as end-node power or memory.
- Military Ad-Hoc Networks: are operated in hostile environments in which disconnection happens because of intentional jamming or other factors.

In DTNs, a network is partitioned into different regions running various network architectures and the connectivity may not be available at any given time. Data delivery is based on an abstraction of message switching capabilities with limited expectations of end-to-end path and node resources. Nodes in a network send data from source to the destination by existing nodes in the same region who relay the data in one or more hops. Each node along the path receives the data from the previous node, stores them in its buffer for a while and forwards to next hop in next contact until the data are finally accepted by the destination.

To accomplish the connection between any adjacent regions, DTNs construct a gateway providing a function of storing messages for reliable delivery. The gateway spans above their corresponding transport protocols [4] [14] [15]. For example there are four regions

named as A, B, C and D in Figure 2.1 (reproduced from [3]). There is a DTN gateway R4 on a bus in region B, by which DTN gateways GW-R3 and GW-R5 are connected. In region D, there is a low-earth-orbiting satellite link. As a substitute for end-to-end network, data could be relayed not only within an intranet network (region C) but also among remote regions (A to D) via temporary connections provided by DTNs.

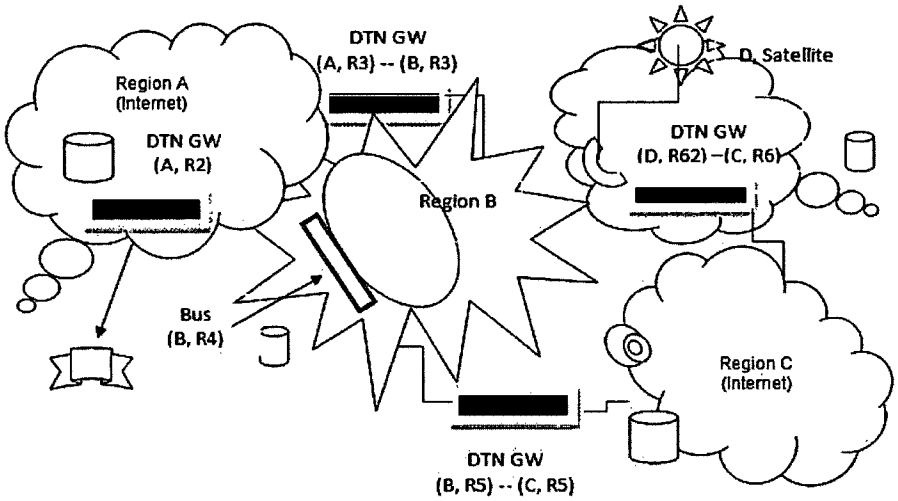

Figure 2.1: The DTN architecture includes the concepts of regions, DTN gateways and dissimilar protocol stacks. Messages could be transmitted in corresponding region, mapped to region-local names as required by the adjacent regions' delivery semantics and then switched via DNT gateway.

2.2 The Strength of Social Ties

Social media make relationships that rage from trusted friend to complete stranger, and the degree of two individuals' relationship varies directly with the strength of ties between them. The concept of the strength of social ties was introduced by Mark Granovetter in 1973. It is a combination of the amount of time, the emotional intensity, the intimacy

(mutual confiding) and the reciprocal services characterizing the ties [16]. Social ties can be characterized into two types of ties: strong and weak. The strong social ties connect individuals who are mostly familiar and weak social ties connect people who are merely acquainted.

The strength of social ties is considered as a tool for analysis of individuals and organizations. A study suggests that mental health could be improved by social support with strong ties [17]. The social framework of strong ties between firms and banks is able to help firms receive lower interest rate on loans [18].

The strength of social ties has four dimensions, and subsequent researches expand the lists into at least seven major dimensions and many manifestations [16] [19]. These include: intensity, intimacy, duration, reciprocal services, structure, emotional support and social distance. To study the question of whether these dimensions could predict the strength of social ties or not, the Firefox extension Greasemonkey is used to guide participants through a randomly selected subset of their Facebook friends [20]. Over seventy variables are distributed in seven dimensions and used in strength-related questions completed by participants. Some important variables are listed:

- Intensity variables: include the total number of words used between friends via a public communication channel (wall), whole messages through a private communication channel (inbox) and the number of individual inbox messages between pairs.

- Intimacy variables: include wall intimacy words or inbox intimacy words which are related to the phenomenon that people with different ties strength use different types of words for communications.

- Duration variables: is the day since first communication (the length of the friendship).

- Reciprocal services variables: comprise information (such as URLs) passed between friends, social or economic goods shared among the friends.

- Structural variables: refer to the groups to which friends belong. These groups are organized in common such as specific topics or interests.
- Emotional support variables: are applied to analyze positive and negative emotion words used among the friends.
- Social distance variables: contain age difference, educational difference and political divergence, etc.

According to the lists, the social media can predict the strength of social ties. Although some findings have disparities with previous conclusions in [21] [22], the intimacy dimension makes the greatest contribution to the strength of social ties.

2.3 Social Selfishness in DTNs

The problem of routing in a DTN has been proposed in [2] [3]. Since the end-to-end data path in a DTN is not available, many assumptions in traditional network routing protocols will not satisfy the situations in DTNs. More restrictions have to be taken into account and the solution designed for DTNs must consider availability of intermediate nodes and bandwidth capacity [23]. Routing model/algorithm needs to know:

- When to send/forward a message
- Where to send a message
- Which message to send/forward
- Which message to delete
- If it follows a single- or multi-copy strategy
- If it acts in a pro- or reactive manner

PeopleNet [24] and Pocket Switched Network [25] are representative models acting as civilian DTNs [26]. In civilian DTNs, where the nodes are human/car/train, a given node may visit some locations which often exemplify spatial correlation of movement. Several algorithms are designed to implement these models, and these solutions assume that all nodes are willing to forward packets to others without considering users' willingness.

However, in the real world, most people are selfish. There is a possibility that a node may not be willing to forward packets to other nodes. This means the node has different preferences according to the properties of social ties [16] [19]. For example, a node is willing to accept packets from someone with whom it has strong social ties such as relatives or friendship, and then deliver packets to the destination. On the other hand, it could perform a different way if it has weaker social ties with others when it has limited resources. Moreover, if a node has no social ties with anyone; its social selfishness becomes individual selfishness. In fact, social selfishness will definitely affect the nodes' behaviours in DTNs. Thus, any routing or model designed and algorithm implemented in DTNs should carefully consider the social selfishness as an important factor.

2.4 Summary

In chapter 2, we presented the basic properties of DTNs as well as the importance of social ties and perception of selfishness in DTNs. We will discuss structure and operation of two models acting as DTNs in next chapter.

CHAPTER 3

MODELS ACTING AS DTNS

In this chapter, we review two models designed for DTNs: PeopleNet and Pocket Switched Networks. We will discuss their architectures and operations in detail with some scenarios.

3.1 PeopleNet

Specific information is not always easily found from internet or libraries. To obtain such information, people would like to ask other people. This scenario whereby people seek information in their social circles could be simulated as a wireless virtual social network [24].

3.1.1 Architecture

PeopleNet architecture, a simple and efficient mechanism, is designed for meeting the needs of these information seekers. This query matching system exploits the natural behaviors of social networking and social mobility, and utilizes the pervasiveness of mobile phones capabilities. When people seek some desirable information, they can input a request query

into their mobile devices which propagate messages into a specific geographic region called a bazaar. Obviously, different bazaars handle different specific types of queries such as a video game related query going to a game bazaar. Queries could be spread by any users in the bazaar since users are mobiles, and then users are able to seek response queries from others, which have some possibilities to match their submitted queries. This is a simple, low-cost and scalable mechanism for dynamic information sorting, storage and access.

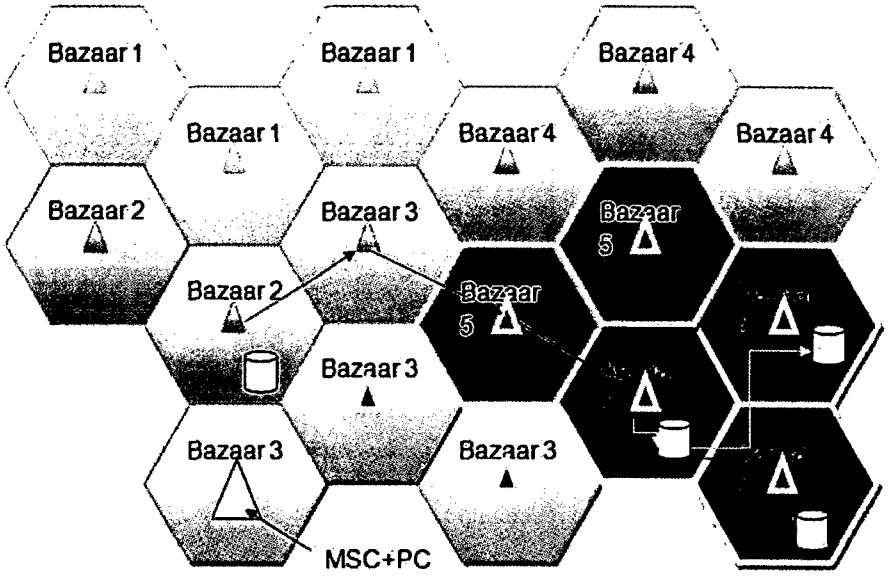

Figure 3.1: Depiction of the PeopleNet Bazaars as an overlay to an existing cellular infrastructure. The hexagonal cells represent the existing cells. Several cells are clustered to form a bazaar, which are represented by different color.

A given area is divided into non-overlapping regions called bazaars handling different types of queries (Figure 3.1 [24]). A bazaar consists of several base stations controlled by a Mobile Switching Center (MSC), and a PeopleNet Coordinator (PC) would be located at the MSC to provide features for PeopleNet. A user needs not be physically located in the specific bazaar to ask a corresponding question, since any query is propagated via the

network infrastructure to some randomly selected users in the associated bazaar. These selected users will spread queries via the peer-to-peer model. Information in a query is organized in a hierarchal structure. Request and response match in case that all specified levels in structure are identical. When the match occurs in a single device, the corresponding users will be automatically notified via the cellular infrastructure.

3.1.2 Operation

There are two methods used for propagating queries in the peer-to-peer model. In random spread method, nodes A and B randomly exchange a copy of queries from their buffer in order for a query to find a match quickly. Since nodes replicate queries and spread them to as many nodes as possible, their buffer capability decreases sharply. This causes some queries to be dropped in order to avoid buffer overflow and provide room for next arriving query. In random swap method, nodes swap the queries without retaining copies of the queries they transmit. By increasing the lifetime in the system, it is obvious that each query has more possibilities to find a match compared to the random spread model. It is evident that the random swap method outperforms the random spread method for the PeopleNet architecture.

Moreover, to determine how many and which queries to swap, the nodes could employ some intelligence in swapping queries by giving higher priority to certain queries and less to other queries. A variety of factors, including the query lifetime or some features predicted benefit determines which methods are implemented to assign the priorities to the queries. For example, there are three identical requests for a video game and one response for a movie in a node buffer. The information can be displayed in the following format: $[1Q, 1Q, 1Q, 2A]$, where the number is the type of a request (Q) or a response (A). A response matches a request if they have the same type. The average number of distinct queries can be improved by using meta-information: $[(1Q, 3), (2A, 1)]$. When two nodes meet, they can apply some algorithms to swap their meta-information in order to maximize matching. Regular greedy approach only takes into account query's count and chooses one

with highest value to make blindly a match. This solution will result into matched queries being exchanged again. Intelligent solution should avoid such an issue by using a thought experiment. It chooses the largest query type first, then reorders the meta-information of peers and finds new largest query type count.

Since there is no notion of reliable communication between specific source-destination pairs in PeopleNet, the methodology used in [27] could be adopted to build an experimental DTN based on the dynamics of PeopleNet.

3.1.3 Extensions

To construct PeopleNet, the following factors should be taken into account:

- Mobility patterns: since random walk mobility model is not ideal for mobile users, more accurate mobility model should be found and applied.
- Selfishness: why should a user be bothered? And what benefits does a relaying mobile phone user have?
- System security and buffer management

3.2 Pocket Switched Networks

As mentioned before, the connection between any two nodes is not always available in DTNs since location of a node changes over time. End-to-end connectivity becomes expensive or unavailable in sparse mobile wireless networks. On the other hand, mobility based on the human movement has potential to increase local connection opportunities. Pocket Switched Network (PSN) falls under the DTN space and aims at providing services adapting to human mobility and exploiting opportunistic network with local/global connectivity. In PSN, data exchange between mobile users' devices makes use of the store-and-forward message strategy [25].

3.2.1 Architecture

In Figure 3.2 [25], the scenario describes when nodes run into each other, they take an opportunity in local network connectivity (using wireless or otherwise) and then determine what to transfer in each direction. In addition, global connection opportunities at some time and in some places are available such as via Wi-Fi access infrastructure or GPRS. PSN could provide highly-robust networking by using opportunities, as it can transparently switch between local and global connectivity. Obviously, human mobility creates the challenge for communication with other users because of unstable paths or unreachable devices. In PSN, human mobility has another important role: the mobility patterns determine the next communication opportunities. Human mobility is so important that it is worth spending time to study.

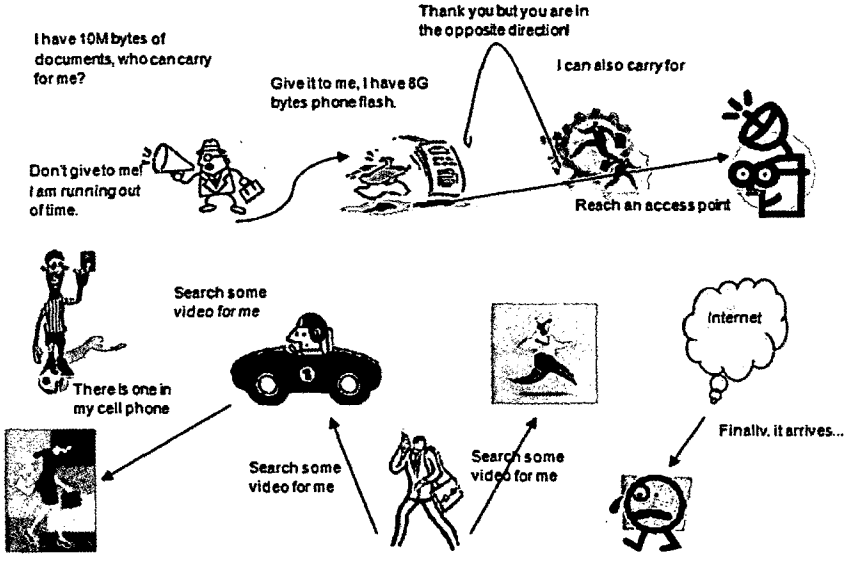

Figure 3.2: The architecture of PSN implements a scenario of data switch using local and global connectivity.

3.2.2 Mobility Patterns

Mobility provides a high possibility for a large bunch of data flooding from one place to another, since mobility increases the capacity of a dense network at some point. However, mobility at the same time also creates the challenge for communication at another place where it creates unstable connectivity and makes end to end communications less possible.

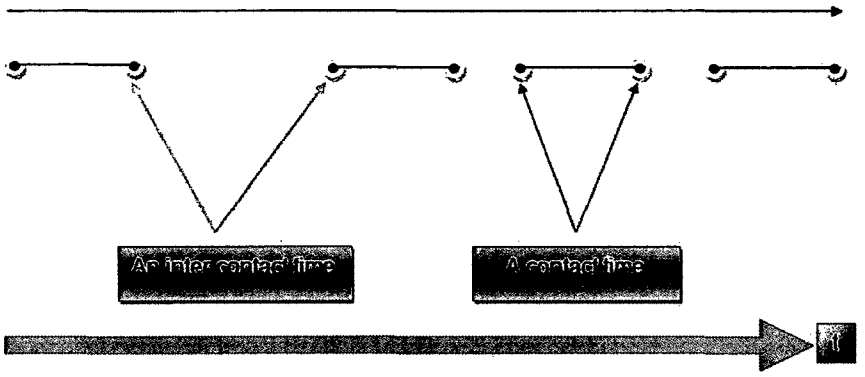

Figure 3.3: The timeline: contact times and inter-contact times. The contact times are two nodes in range of one another. Inter-contact times are simply the times between the contact times.

For a given pair of nodes A and B, the timeline can be divided into two regions: contact times and inter-contact times (Figure 3.3 [25]). It is obvious that two persons may meet each other, go away and then meet again at some other time in the daily life. When two nodes are within a contact range, the duration is called contact time. In addition to the bandwidth of transmission, how much data two devices can transfer between each other is also heavily dependent on the contact time. Inter-contact time is the time between two contacts, which determines how often a communication is possible. According to the study of human mobility in 2005 [25], the distribution of the inter-contact time between two nodes in an opportunistic networking environment follows an approximate power law over a large range, and the power law coefficient is less than one. The power law coefficient affects

the forwarding algorithm: normally, the larger the coefficient, the smaller the delay. This illustrates inter-contact time is an important parameter in forwarding algorithm design:

- Shorter inter-contact time: two people contact each other more often and they can exchange the data directly.
- Longer inter-contact time: there is a high possibility that other media should be used to transfer the data instead of waiting forever.

3.3 Summary

In this chapter, two models acting as DTNs were presented. To implement PSN, some routings will be introduced in next chapter. We will classify them into several groups. We also discuss SimBet routing and social selfishness aware routing in detail and compare their performance.

CHAPTER 4

INTRODUCTION TO ROUTING

In this chapter, we classify routing schemes for DTNs into several categories based on their designs and characteristics. We also discuss two approaches (with or without consideration of selfishness) to solve information diffusion challenges in DTNs, and compare their performance.

4.1 Data Delivery Mechanisms

Probability functions or metrics are used in routing algorithms for making decisions whether packets should be forwarded or not. The successful delivery is defined as the percentage of the sent messages that are actually delivered to destinations before the messages' time out. The delay is measured as the average time of delivery of these messages. The difference among these measurements is due to the values and properties they have taken into account. The algorithms are classified based on their sources of values and properties.

4.1.1 Stateless Algorithms

The reason algorithms are grouped into this category is that they do not maintain history data or attempt to predict performance in the future. Generally, they can be characterized as stateless. Each node takes advantage of contact opportunities to forward as many packets as possible while in contact with another node.

Epidemic routing [28] is one of the simplest protocols proposed for data delivery in DTNs. Epidemic forwarding is based on flooding. The idea is if two nodes meet they will exchange all the messages in their buffer except duplication. At the end of this exchange, they will have the same set of messages. Eventually, every node will be able to send information to other nodes so that packets are flooded through the network which looks like the spread of a viral epidemic. This algorithm does not consider storage and bandwidth constrains as well as the properties of networks and nodes. It results in some efficiency issues such as redundant messages existing in networks even after they have been received by destinations.

Further work has been done in designing efficient and better performing forwarding protocols. In [29], an improved epidemic scheme restricts the buffer size and applies dropping policies. Another algorithm [30] reduces the resource consumption by killing copies of a message in the network, which has been delivered to the destination. Spray and Wait [31] controls the replications to a specified a-priori number and performs well in low traffic network.

4.1.2 Location-based Algorithms

Currently, a node could obtain its location information by a GPS device attached to it or a virtual coordinate space. When two nodes meet, a node forwards its data only if another node is closer to the destination. The algorithms in this group are based on their analysis of how location data can be used to determine an optimal route for a message. Location-based Algorithms avoid maintaining all routing tables and exchanging any unnecessary

information between the nodes.

MoVe [32] is a representative location-based algorithm using GPS. This scheme learns structure in the movement patterns of network participants and uses it to enable informed message passing. For the neighbour nodes in the lists predicted to move to the destination, a node N calculates distance between neighbour nodes and destination according to their location information. Then N forwards its data to a neighbour M if and only if M is the node closest to the destination in the list and M is closer to destination than itself.

MobySpace [33] is the routing proposed for DTNs with the help of a multi-dimensional Euclidean virtual space in which each node is represented by a point. In MobySpace, when a message is forwarded between two nodes, they have the property that the mobility pattern of next hop is more similar to the destination than that of previous one [34].

4.1.3 Knowledge Algorithms

The algorithm for routing in DTNs presented in [23] is based on the presence of knowledge oracles. The DTN graph is a directed multi-graph in which the link capacities are time-dependent. The various knowledge oracles include average waiting time for next contact, duration of a contact, available buffer size in each node, and the present or future traffic demand. Obviously, complete knowledge of input variables facilitates the computation of optimal routes while zero knowledge results in poor performance. Based on the various knowledge oracles, link capacities are available and then Dijkstra's shortest path algorithm can be applied to calculate shortest paths. Moreover, a linear programming formulation uses all the oracles to determine the optimal routing for minimizing average delay in the network.

It is evident that if an algorithm has more knowledge oracles, it will provide better performance. However the global information and future traffic demand are not always available, algorithms making per-hop decisions based on local knowledge (such as probability of delivery) may also behave well. For example, the idea of data carriers (termed as zebroids) [35] is to utilize knowledge of the contacts between the vehicles in the near

future. The forwarding path is determined based on the knowledge of contacts.

4.1.4 Other Categories

Algorithms (such as Bubble [36]) forwarding messages based on techniques that exploit social aspects of node movement could be classified into another group. Other categories include network coding based schemes [6], gradient-based schemes [37] and so on. Normally, many algorithms combine two or more approaches but focus on one specific topic to optimize. For the purpose of presenting a general overview of data delivery mechanisms, others might choose different classification criteria depending on what is intended to be shown. In the next section, we will review two algorithms in more detail and compare their performance.

4.2 Metrics of Interest

The major purpose of routing algorithms for DTNs is to tune system performance. The following are important metrics affecting the optimization.

4.2.1 Delivery Ratio

The number of packets (or messages) created by different nodes is received by their intended destinations in the network setting under consideration. If a message is dropped by any reason it is considered as a failed delivery. Delivery ratio is defined as the proportion of messages that are delivered to their destinations among the total messages generated.

4.2.2 Delay

According to the properties of DTNs, the applications should tolerate larger delays as long as packets traverse nodes and reach the destinations. The optimization is to keep it as short

as possible subject to resource constraints. A modified definition of the delay is average cost: the average number of forwards done per successfully delivered message.

4.2.3 Number of Replicas

Usually, some algorithms improve their performance by generating additional copies of a message and transmitting them via different relays. The accompanying issue is how to control the number of replicas (buffer management). Moreover, when a message is received by its destination, the elimination of redundant copies also should be considered.

4.2.4 Energy/Power

The energy used in DTNs includes computation, movement, delivery and waiting. Normally, this metric is difficult to quantify and it is not a main issue of study.

4.3 SimBet Routing Vs. SSAR

Both SimBet Routing and Social Selfishness Aware Routing (SSAR) exploit the probability of delivery of nodes. The difference is that SSAR takes social selfishness into consideration but SimBet Routing does not.

4.3.1 SimBet Routing

In [38], the use of social network analysis techniques is proposed for solving the issue of message delivery in disconnected delay-tolerant sparse Mobile Ad hoc Networks (MANETs). When nodes in a disconnected network move freely, the main challenge is to establish a route with high performance and low delay to forward packets between source and destination. The algorithm SimBet Routing makes use of the consideration of small world properties [39]. Due to the complexity of network, the whole network might behave as cliques in which only local information is available. Some bridge nodes are identified based

on their centrality characteristics such as the capability to switch information among disconnected nodes. In Figure 4.1 [38], there are three disconnected clusters. Source node S wishes to send a message to destination node D, but none of the nodes in the left upper cluster have a path to connect to destination node D. It results in the difficulty of message delivery. In this case, some bridge nodes exist in three clusters linked by ties from N_1 to N_2 and from N_3 to N_4. Due to the bridge nodes, there is a path among clusters if these four bridge nodes are considered as intermediates.

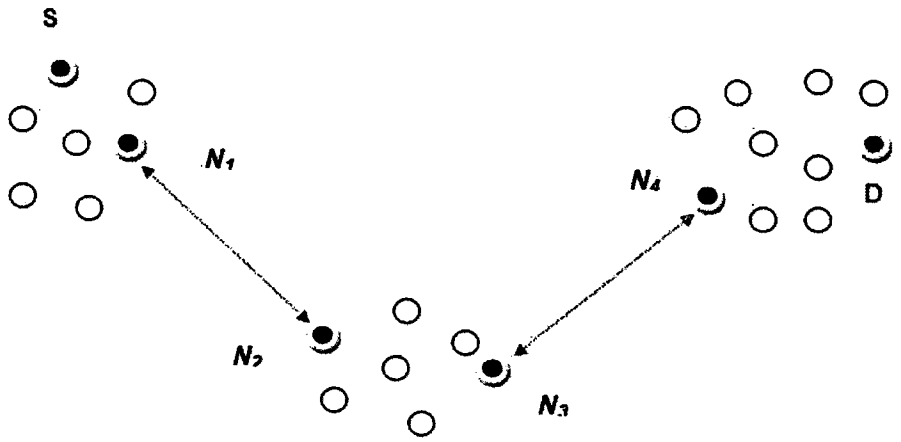

Figure 4.1: Forwarding packets in disconnected Clusters.

Bridge nodes have important structures. There are two measures used to determine the structural importance of the node. One is ego-centric centrality and the other is social similarity. Bridge nodes are identified by the estimation of nodes' centrality in the network. Freeman degree, closeness, and betweenness are most widely used centrality measures [40] [41]. Usually, a central node has a stronger capability of connecting other network members. However in networks with a large node population, it is very difficult to find the central nodes since the complete knowledge of the network topology is not always available. To get around this issue, the concept of Marsden's 'ego networks' was introduced in this algorithm in order to measure the centrality of a node in a network [42]. Ego networks can be defined

as networks consisting of a single actor (ego) together with the actors they are connected to (alters) and all the links among those alters [43]. Egocentric centrality performs much better compared to the traditional Freeman's centrality metrics. Ego networks take betweenness centrality into consideration for nodes with which the ego node has come into contact. Node contacts (Figure 4.2-a [43]) could be represented by an adjacency $n \times n$ symmetric matrix A (Figure 4.2-b [43]). If there is a connection between two nodes, the value is set as 1. Geodesics in the network are either of length 1 or 2. The geodesic length of any pair of non-adjacent alters is 2, since it passes through an ego. A (Figure 4.2-b) is the adjacency matrix for a graph G (Figure 4.2-a), and $A^2_{i,j}$ contains the number of walks of length 2 connecting i and j. If $i \neq j$, it must be a path. To count the number of paths (geodesics with length 2) for non-adjacent pairs of actors, it follows the formula $A^2[1 - A]_{i,j}$. The ego betweenness is the sum of the reciprocals of the entries of $A^2[1 - A]_{i,j}$(Figure 4.2-c [43]). Since the matrix is symmetric only the non-zero entries above the leading diagonal are considered. The betweenness of ego is simply the sum of the reciprocals of the entries, which is 0.25.

Social similarity is relative to the network topology. The degree of separation can be measured by the ratio of common neighbours between individuals in social networks. When the degree of separation is high, it takes longer time to diffuse messages among the neighbours of nodes. Therefore nodes with a low degree of separation are good for routing. A node has high similarity if it is the common neighbour among other nodes. In other words, this node has low degree of separation. Node's direct similarity is calculated using the same $n \times n$ symmetric matrix. Node's indirect similarity is calculated by an additional $n \times m$ matrix. Figure 4.2-d [38] shows that node 3 and node 5 share node 4 as a common neighbour; therefore there has a similarity 1.

The purpose of SimBet utility calculation is to compute the forwarding quality of a possible relay node. The SimBet utility is evaluated based on ego-centric centrality of nodes and their social similarity. The value of SimBet utility is between 0 and 1. The SimBet routing represents the communication between nodes N and M. When two nodes meet,

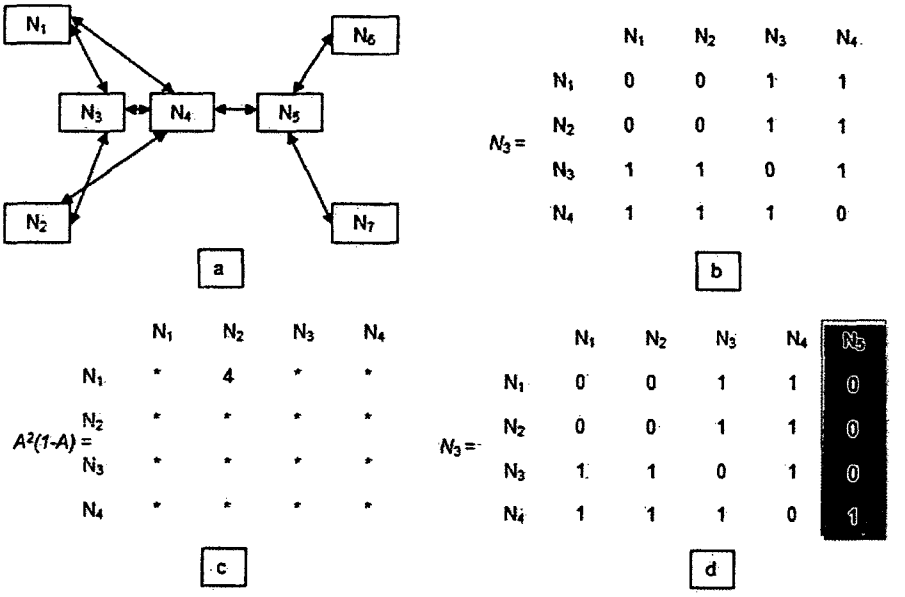

Figure 4.2: An example of egocentric centrality and social similarity calculation.

they exchange the lists of destinations for messages as well as their locally determined betweenness value and similarity value for these destinations. For every destination, each node updates betweenness and similarity values and then calculates SimBet utilities. According to the result, if node N has lower SimBet utility on the destination, it will forward corresponding message to the node M and remove it from its queue. Finally, two nodes swap all messages if necessary. It means the node with high SimBet utility will carry more messages for distribution.

4.3.2 Social Selfishness Aware Routing

In SimBet routing (and other previous routings), nodes are always willing to forward packets for others. It does not match to conditions in the real world, since most people are socially selfish. Social selfishness is a kind of user demand and affects nodes' behaviours. Two observations could be obtained from social perspective:

- If a selfish user gets help from a person in the past or could get help from this person in the future, he is usually willing to help this one.
- The willingness varies according to the strength of the social ties.

Based on the observations, SSAR [8] is proposed. It allows user selfishness and provides better routing performance in an efficient way. The socially selfish network is considered as a fully-connected weighted directed graph where edges are the social links between nodes. The weight of a link is the value of willingness, a real number between zero and one. The stronger the social ties are, the larger the social willingness is. In a social selfishness system, friendship between two nodes could be ranked [20].

The architecture of SSAR is constructed with four components. Each component has different responsibility respectively. The following description illustrates how SSAR works. When node N runs into node M, N and M deliver packets destined to each other. During the packet delivery, they also swap a list of information about packets in their buffer, including the destination ID, expiration time and priority. According to the list, packet priority manager (PPM) calculates a new priority for each buffered packet based on its willingness. PPM calculates the new priority with following formula $p_i = p_{i-1} \times \omega$ where p_i is the packet's priority in its i^{th} hop and ω is the i^{th} hop's willingness to forward the packets from the $(i-1)^{th}$ hop. For example, node A has a packet with priority 0.8 and node B (with willingness value 0.7) will help A to forward this packet to destination. After A forwards it to B, the new priority for such a packet is 0.56.

Based on the new priority and other information such as expiration time, each node calculates its delivery probability using delivery probability estimator. Delivery probability estimator (DPE) quantifies the node's forwarding capability for the packet. The overall delivery probability is about the possibility that a packet is delivered to its destination. The complement of delivery probability describes the reasons that packet will be dropped. It is determined by two independent droppings. The first is if a packet does not have enough priority it could be dropped when buffer overflow occurs. The second is if it is not received

by its destination before the expiration time it will be dropped.

With the approach mentioned in [44], the expiration dropping probability represents how to avoid dropping a packet before it is received by the destination. If nodes have a lower average inter-contact time with destination, they will have high contact frequency and then a packet traverses these nodes with a lower expiration dropping probability.

The factors affecting buffer overflow dropping probability include priority value p, the current empty buffer size L and residual time before expiration t. According to data mining technology, the probability a packet will be dropped is similar to some historical packets which have similar feature values when they enter a node's buffer. Suppose a packet matches to a set S of similar packets and its dropped subset is S_{drop}, then buffer overflow dropping probability is $|S_{drop}|/|S|$.

After delivery probability is calculated, available buffer size will be determined by buffer manager (BM). In brief, a packet with low priority would be dropped if buffer overflows, or replaced by new coming packets with high priority. According to the available buffer size of node M, node N has determined a candidate packet set C with two principles:

- M will not accept the packet from N if it does not have enough space.
- N tries to maximize its selfish gain through this contact.

In the first principle, M's available buffer size for the incoming packets is defined as the total space of current empty buffer size and potential buffer size in which the priority of a packet in M's buffer is smaller than that of a packet in delivery list of N. In the second principle, the selfish gain is defined as the product of packet priority in N and the increment of delivery probability. It could be achieved by forwarding packets from node N to node M. Both factors in the definition are related to selfishness. Packet priority means how socially important the packet is. The increment of delivery probability means how much this forwarding can increase the packet's probability to be delivered. So their product is a natural representation of the gained selfishness.

Two nodes calculate delivery probability and available buffer size, and swap them again. Each node determines a candidate set of packets for delivery. Finally, forwarding set manager (FSM) decides which packets to transmit. This is a multiple-knapsack problem (MKPAR). The original buffer is divided into $|C| + 1$ knapsacks. Each knapsack has predefined size. So the packet i can only be packed into Knapsacks indexed not larger than i. Since this problem has been proved as NP-hard [45], a greedy algorithm is applied to optimize the running time. All candidate packets are ranked in the decreasing order of selfish gain weighted by packet size. If available buffer size L is larger than current packet size of i, the packet i will be forwarded and available buffer size L for next packet will be reduced to $L - l_i$, where l_i is the size of packet i in the candidate forwarding list. Otherwise the algorithm skips packet i and keeps on checking the next one. It continues one by one until no more packets can be forwarded. Forwarded packets in original buffer will be deleted after being forwarded, so there is only one copy for each packet.

4.3.3 Performance Comparison

Two types of social graphs are constructed for evaluating performance of SSAR and SimBet Routing. The first one is probabilistically contact dependent social graph with following heuristic: the stronger ties two individuals have, the more likely they contact frequently [16]. According to the study in [46], the power-law distributed degrees are assigned to nodes in the decreasing order of f_N/f_*, where f_* is overall contact frequency in the system and f_N is the node N's overall contact frequency. The weight of social ties of each node is evaluated between zero and one [20]. The weight of edge of $\overrightarrow{NM}$ is determined by f_{NM}/f_N, where f_{NM} is the contact frequency between N and M. The second type is random social graph that is constructed with random assignment of degrees for nodes and social ties between any two nodes.

Since SimBet Routing does not consider social selfishness, some modifications are applied before comparison. In revised SimBet Routing, nodes do not forward packets to others which are not willing to forward for them in order to avoid immediate droppings

caused by selfishness. The dropping policy is optimized and the transmitting packets order follows the method described in [47].

By increasing time to live (TTL) in two types of graphs, more packets could be delivered to destinations. But due to the forwarding capacity of the network, the delivery ratio will not increase after the TTL reaches a certain value. Social ties are more likely to be added to nodes with more frequent contacts when the average numbers of social ties per node increases. It means the network's contact opportunity does not increase too much. The test results are listed in Table 4.1 [8].

Table 4.1: Performance Comparison between SSAR and SimBet.

	Contact Dependent Social Graph		Random Social Graph	
Items	*Delivery Ratio*	*Transmissions*	*Delivery Ratio*	*Transmissions*
SSAR	0.4	2000	0.42	1500
SimBet	0.2	3200	0.28	2400

SSAR has higher packet delivery ratio than SimBet Routing, because it avoids contacting low-willingness nodes and forwarding a packet to the receiver whose buffer is insufficient. For transmissions test (the lower the better), the performance of algorithm SSAR is still better than that of SimBet Routing. Moreover, SSAR allows better selfishness than SimBet Routing, since SSAR implements buffer management policy to achieve social selfishness, delivers packets taking selfishness information into consideration without purely relying on contact opportunities and it also applies smart forwarding algorithm. SimBet has good performance in the system when the movement of the nodes is predictable and not highly dynamic. Otherwise, the need of maintenance and updated routing tables will significantly slow down its performance.

4.4 Summary

In this chapter, we have classified routing schemes according to their data delivery mechanisms. We have also discussed two approaches and compared their performances. In the next chapter, we will resolve three problems: a) efficiently select a forwarding object, b) understand next hop's forwarding strategy and c) intelligently maintain the message's priority, and propose an improved socially selfish routing.

CHAPTER 5

IMPROVED SOCIALLY SELFISH ROUTING

In this chapter, we will discuss three problems why they are important in socially selfish DTNs, propose an improved model, and illustrate how a routing architecture could solve these problems.

5.1 Problems Statement

As mentioned before, the communication among mobile nodes in DTN has following characteristics: end-to-end path not always available (frequent disconnection), limited resources and unreliable wireless transmission. For a sparse DTN investigated in recent studies, several mobility-assisted routings based on store-carry-and-forward have taken social selfishness into account. In [8], a node is selected as a relay if it has high willingness to help others. In [9], routing considers honesty and unselfishness for social trust to account for node trustworthiness for message delivery. In [48], forwarding strategy makes the delivery

decisions of messages based on the differentiated friendships. In these routings, willingness will give the nodes an incentive to behave in the aforementioned ways to satisfy their selfishness. However, no considerations were given to solve the following questions: a) how to efficiently select a node as the most suitable forwarding object, b) how to understand next hop's forwarding strategy, and c) how to intelligently maintain the message's priority. With these in mind, we need to address the following problems in this thesis.

5.1.1 Problem 1: Efficiently select a forwarding object

In [48], it never mentions the condition that a node may be in contact with multiple neighbours at the same time. In [8], when a node detects some other nodes within its contact range the node contacts its neighbours one by one. This will cause some undesirable results, and lead to reducing packet delivery ratio and increasing transmission cost.

Example 1 *This is a scenario that a node inefficiently selects forwarding objects (Figure 5.1).*

Node N_1 discovers three nodes N_2, N_3 and N_4 within its contact range. According to the order of willingness of a relay or a random order, it may communicate with them in sequence of N_3, N_2 and N_4. If N_1 firstly contacts N_3, a packet P_2 (whose destination is N_2) is delivered to N_3. In next turn, N_1 talks with N_2, and simultaneously N_3 forwards packet P_2 to N_4. Finally N_1 and N_4 swap their packets, and then P_2 is returned to N_1. Actually packet P_2 still has the possibility to be received by N_2 in the future, but the result reduces the routing performance with high transmission cost. In the worst case, P_2 will be dropped because it exists in the system too long and it misses the only opportunity.

Since nodes are mobile and their activities are time based, there is a low possibility that a node could communicate with several nodes one by one if it detects these nodes within its radio range. How can nodes efficiently filter the multiple neighbours so that packets could be delivered to their destinations with low transmission cost and dropping rate? On the other hand, a node with higher willingness is always the first choice. This leads to the

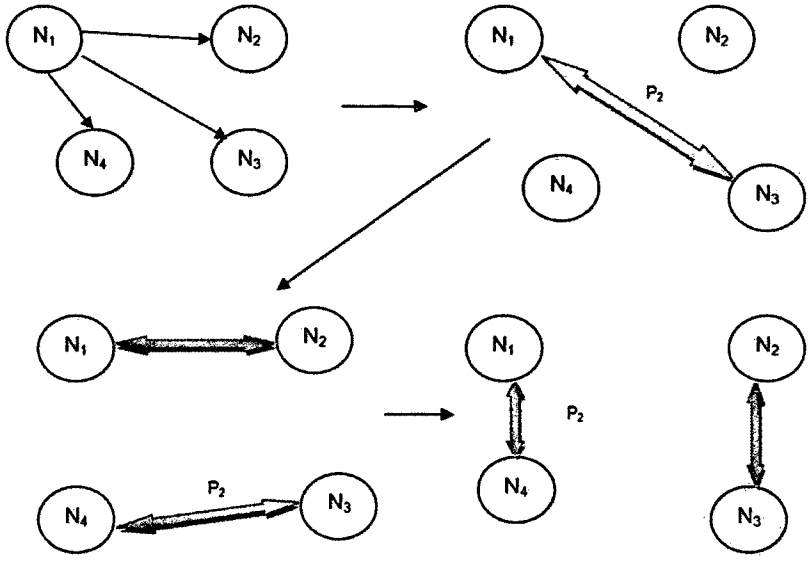

Figure 5.1: A scenario that a packet could not be received by its destination because N_1 inefficiently selects forwarding object.

smart selecting strategy taking both packet's destination and next hop's willingness into consideration in order to satisfy node's selfishness and also maintain system performance.

5.1.2 Problem 2: Understand next hop's forwarding strategy

When a node has determined the forwarding object, a communication channel is established between two nodes. Normally, either the willingness between current node and next hop [8] or the friendship between next hop and destination [48] is the most important factor affecting a node to make a decision on forwarding candidates. The former option ignores the next hop's attitude to the destination (Example 2), which results in a packet possibly not being forwarded to its suitable relay. The latter option overlooks the willingness between the current node and the next hop, which leads to packet dropping (Example 3).

Example 2 *This is a scenario that a packet may not be received by the destination (Figure*

5.2).

Suppose node S has two packets P_1 with destination D and P_2 with destination Z respectively. In this case, S has no packet destined to node A and node B. When S detects A and B within its contact range, it selects the next hop purely due to the node having higher willingness to help it although it has no packets for A or B. S forwards packet P_1 to A after recalculating packet priority based on the willingness (value of A to S is 0.6 larger than that of B to S). In the following step, node A meets node C and D. A has each packet destined to C and D respectively. A will consider node C as the next forwarding object since the willingness from C to A is higher. The delivering chain from S to C may not be suitable for packet P_1.

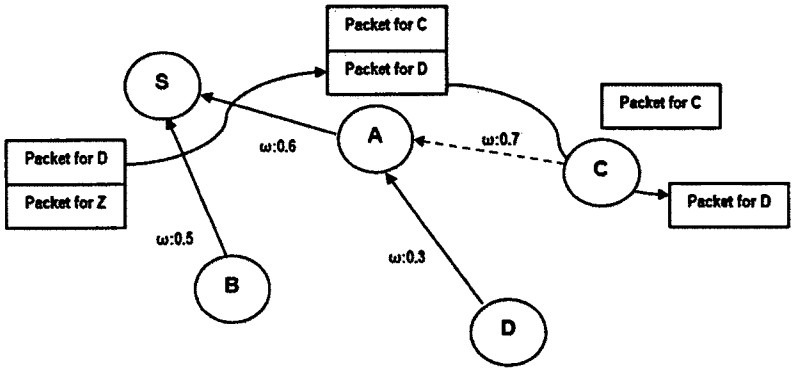

Figure 5.2: The packet cannot reach its destination since the source node S ignores the next hop's attitude to the destination when S forwards packets to next hop.

Example 3 *This is a scenario that a packet is dropped since only friendship between next hop and destination is considered (Figure 5.3).*

When node A encounters node B and finds friendship between node B and destination D is 0.8 much higher than that between D and itself, it forwards packets to B without considering their lower friendship. Node B will drop the packet since it is unwilling to help A (the friendship between B and A is 0.01).

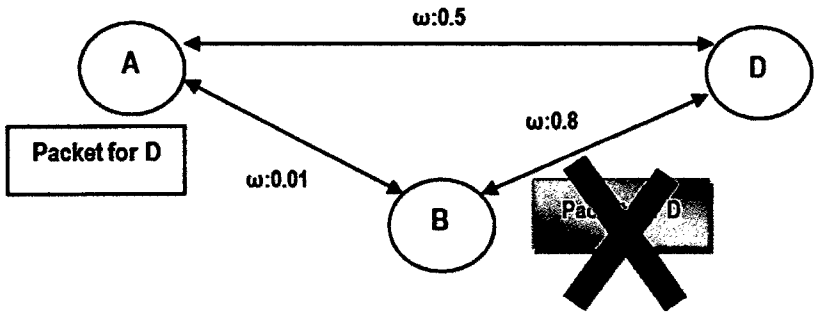

Figure 5.3: The packet will be dropped since source node overlooks willingness between next hop and itself.

Two examples illustrate the problem that in the real world, the intermediate node may drop the message or deliberately delay delivery if it has weak social ties with either predecessor or destination. To avoid such phenomena, the problem should be addressed from a different point of view – next hop's forwarding strategy must be carefully considered including the willingness of next hop for predecessor and the relationship between next hop and destination.

5.1.3 Problem 3: Intelligently maintain the message's priority

The basic idea of packet priority estimation is from semantic constraints for trust transitivity [49]. The calculation for each buffered packet is based on the willingness of nodes that the packet has traversed. The process is determined in a chained way. The basic formula for priority calculation is: $p_i = p_{i-1} \times \omega$ where p_i is the packet's priority in its i^{th} hop and ω is the i^{th} hop's willingness to forward the packets from the $(i-1)^{th}$ hop [8]. However, this formula will not actually reflect real world phenomenon where people have different attitudes for the messages with different priorities. It causes some accidents in packets' delivery.

Example 4 *An accident occurs in a system because of no intelligent maintenance for message's priority.*

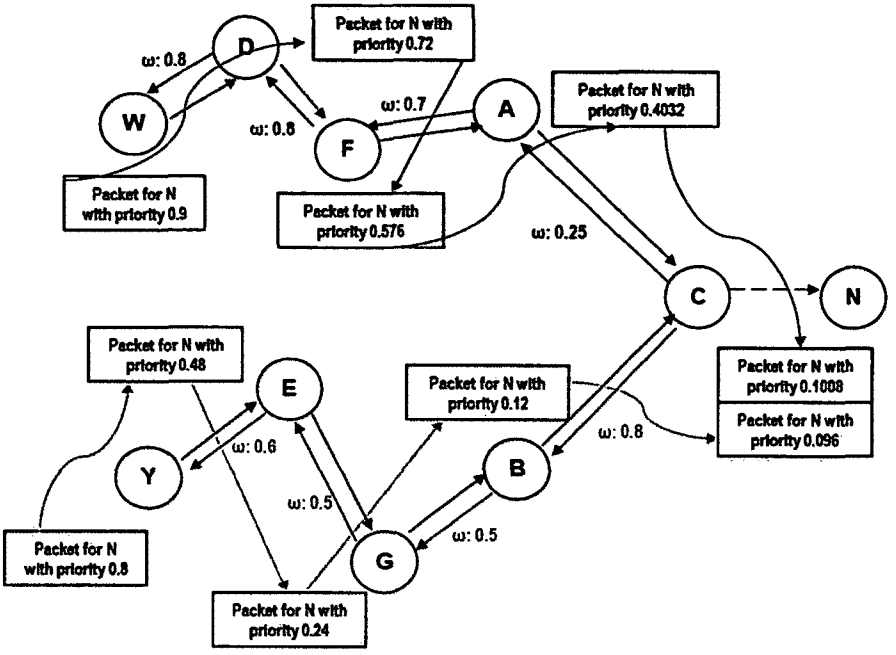

Figure 5.4: One packet traverses from node W to node C (in blue chain) with variation of priority (in red chain) and willingness (in black chain). Another packet is from node Y to node C (in green chain) with variation of priority (in orange chain) and willingness (in purple chain).

In the following scenario (Figure 5.4), suppose there is a packet with destination N from node W to node A, the result of sequential priorities is $0.9 \times 0.8 \times 0.8 \times 0.7 = 0.4032$. *When node C meets node A, C gets this packet from A and assigns a new priority, it puts* $p_{CW} = p_{AW} \times \omega_{CA} = 0.4032 \times 0.25 = 0.1008$ *where p_{AW} is the priority that A sets for packet from source W via node D and F, ω_{CA} is C's willingness for A and p_{CW} is new priority for this packet due to the formula. And there is another packet with destination N from node Y to node B with the sequence of priorities* $0.8 \times 0.6 \times 0.5 \times 0.5 = 0.12$. *When C meets node B*

and gets this packet, it will assign a with new priority $p_{CY} = p_{BY} \times \omega_{CB} = 0.12 \times 0.8 = 0.096$ *where* p_{BY} *is the priority that* B *sets for packet from source* Y *via node* E *and* G, ω_{CB} *is* C*'s willingness for* B *and* p_{CY} *is new priority for this packet due to the formula.*

Normally, if time is limited and only one packet can be delivered from node C *to node* N, C *will forward the packet from node* A *when* N *is available. The reason is that the packet priority is higher than that from node* B*. But in real world,* N *can remember where the packet comes from, check its priority, compare willingness among previous nodes and then make a decision which packet should be forwarded. In this scenario, the willingness value of* ω_{CB} *is almost* 3.2 *times higher than that of* ω_{CA}*, which means there is a much stronger tie between node* C *and* B*. The ratio of two packets' priorities is* 1.04*; the difference is so small that it could be ignored.* C *would still like to forward the packet from* B *instead of that from* A*.*

This phenomenon illustrates the selfishness that node C values its friendship higher than the forwarding rules and therefore it chooses the higher value. Can we have a more intelligent approach to calculate packets' priorities so that there is significant difference between two packets' priorities in Example 4? In other words, node C may abide the forwarding rules if two packets' priorities have an obvious difference. The purpose is not to completely eliminate selfishness but to find a smart strategy to avoid a node in a dilemma and prevent a larger accident.

Before the approaches are presented to solve the listed problems, we will introduce new models, make some reasonable assumptions, provide more detailed explanations, propose new algorithms and address these challenges. Finally, an improved socially selfish routing will be constructed.

5.2 Models and Assumptions

5.2.1 Value of Willingness

The socially selfish network is modeled as a fully-connected weighted directed graph, where the vertex set V consists of all the nodes and the edge set E consists of the social links between nodes. The willingness of node B forwarding packets to node A is the weight of $\overrightarrow{BA}$, which is represented as $W(\overrightarrow{BA})$. The value of $W(\overrightarrow{BA})$ and $W(\overrightarrow{AB})$ may differ. The value of willingness can be easily mapped into a real number between zero and one. When value is very small or zero, it means node B will be disinclined to give assistance for node A. If the willingness has high value (such as one), B is very enthusiastic to forward packets to A. Each node knows its willingness to forward for others by quantitatively rating its friendships [20].

In the real world, a selfish user is usually willing to help others with whom he has strong social ties, because he got help from them in the past or will probably get help from them in the future. This relationship is based on the reciprocal respect and privileges.

Assumption 1 *The value of $W(\overrightarrow{BA})$ and $W(\overrightarrow{AB})$ may be different, but there should be no big difference between them: $|W(\overrightarrow{BA}) - W(\overrightarrow{BA})| \leq \varepsilon$ for some constant ε. For example ε is less than 0.1, which reflects the relationship between two nodes is based on reciprocal respect.*

In the simulation environment, the system initializes the relationship between nodes. The complete willingness information is available for the system, but each node only needs to know its willingness to forward for others. A node could estimate another node's willingness to itself according to Assumption 1. Moreover, to apply a smart forwarding strategy, node should know next hop's willingness to the packets' destinations.

Assumption 2 *When a communication channel is built between two nodes, each node could acquire the other node's willingness (next hop's willingness) to the packets' destinations. This is achieved by request and honest response between two nodes.*

5.2.2 Bidirectional Model

In a social network, if A is the closest friend of B, A may understand B's habits and thinking. B might share extremely strong interpersonal ties with A, but there is no guarantee that A could know everything else of B. There is a very high possibility that A has no idea how B actually evaluates his/her relationship with C. This could explain the reason why the issue (undesirable forwarding) happens in Figure 5.2. Packet's new priority is completely based on the node A's willingness and source S forwards its packet to node A without taking the relationship between A and destination D into consideration. It is called one-way-direction issue.

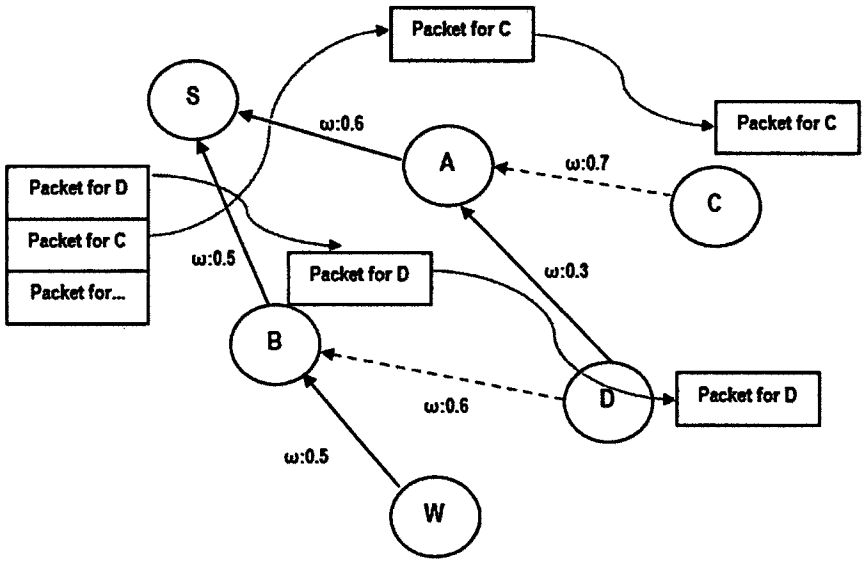

Figure 5.5: Node S makes its decisions for the forwarding list according to bidirectional model.

In bidirectional mode (Figure 5.5), source S assesses the willingness in combination with $W(\overrightarrow{AS})$ and $W(\overrightarrow{DA})$, before it forwards the packet to next hop A. Although the social ties between S and A is stronger than that between S and B, the willingness value of D

to A is so small that the packet is not suitable to be delivered to A. On the other hand, S should forward packet (with destination C) to A because of the bidirectional willingness value: the combination of $W(\overrightarrow{CA})$ and $W(\overrightarrow{AS})$; dispatch packet (with destination D) to B according to the bidirectional willingness value: the combination of $W(\overrightarrow{BS})$ and $W(\overrightarrow{DB})$ respectively.

Although there are some other factors (such as delivery possibility) affecting S's decision which packets should be dispatched in this turn, bidirectional model provides big benefits for the packets traversing the delivery chain. According to assumption 1 and 2, the value of $W(\overrightarrow{CA})$ and $W(\overrightarrow{AC})$ should be at the same level. S might not know the value of $W(\overrightarrow{CA})$, but it could estimate it based on the value of $W(\overrightarrow{AC})$ (obtained from A). This helps S to determine the forwarding list and avoids the mentioned issue. Obviously, the benefit is available in the whole process, even if node B does not meet node D in the next step. B could still find the next hop with high willingness to forward this packet until destination D receives it.

5.2.3 Reliable Model

Normally in the real world, when A asks B to help forwarding a packet, B may have a request for the original owner. In some cases, if B does not like the owner, B would refuse this task no matter what relationship is between A and itself. On the other case, B does not care whom the packet belongs to. This factor will make the system very complex if it is introduced into forwarding routing. As a simple solution, the intermediate nodes should not know the actual packet source. This could be achieved by encrypting the corresponding information so that only the destination can decrypt. Then nodes provide data forwarding service only based on the previous hop information and the destination information.

Assumption 3 *The source of a packet is anonymous to intermediate nodes.*

Moreover, the nodes in this system are not able to impersonate others to obtain forwarding services. They could not drop packets without any reasonable justification. They

also should not seek any interests by breaking the delivery chain and encroaching upon others' packets. For example, node S asks for A's willingness to C in Figure 5.5 and the feedback from A must be honest.

Assumption 4 *The nodes in the system are trustable.*

5.2.4 Network Model

In DTNs, when two nodes meet, they would swap information, reassign packets priority, determine which messages should be delivered, and then forward these messages. The whole process must be completed in a limited time. Since the entire communication time is limited, nodes should have strong computational capability so that a large proportion of total time could be left for forwarding packets. In addition, the bandwidth between two devices for packet delivery is also limited and the big size data could not be forwareded in the allocated time. According to such conditions, each packet should have small enough size in order to achieve the task in each round. So it is possible to forward as many packets as possible.

Compared to limited packet size, each node has unlimited buffer. This is because large storage capability is available on currently used mobiles such as a gigabyte SD memory card whose size is much bigger than forwarded messages. This assumption promises that any packet will not be dropped due to storage size issue. But a packet would be dropped due to its expiration time. The longer time the packet exists in the system, the less importance it has. Outdated packets should be dropped.

Assumption 5 *In the limited time (duration of contact time), the forwarded packets size should be restricted. The storage buffer for each node is unlimited.*

Assumption 6 *Each packet has a given lifetime.*

5.2.5 Adversary and Fault Tolerant Model

In this thesis, attention concerns deep understanding of social network, scenarios of selfish behaviours, and improvement on forwarding performance and transmission cost. The defence of malicious attacks and free-riding behaviours are not the focus of this study. Such researches can be found in [49] [50].

Assumption 7 *There is no malicious attack in the system.*

During the forwarding process, a packet might be corrupted for some reason. Fault tolerance in DTNs deserves separate studies and it will be discussed in the section of future work (Section 8.2).

5.3 Packet Structure

The study of packet structure (Figure 5.6) helps to understand the system architecture and principle of protocol (algorithm). The packet structure has the following five components:

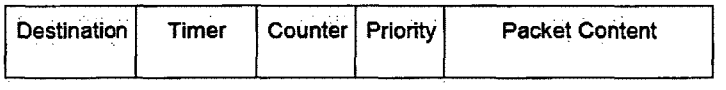

Figure 5.6: The structure of packet has five components: destination, timer, count, priority, and packet content.

Destination: The intermediate node N should know the destination (D) of the delivered packet, and then estimates the willingness value $W(\overrightarrow{DN})$ based on $W(\overrightarrow{ND})$. According to Assumption 1, there is no significant difference between two values (such as the difference between $W(\overrightarrow{DN})$ and $W(\overrightarrow{ND})$ is a real number smaller than 0.1). Finally, N could determine the packet's priority by bidirectional value: the combination of $W(\overrightarrow{DN})$ and $W(\overrightarrow{NS})$ (S is the predecessor node).

Timer: In this part, a timer is set up for recording elapsted time the packet exists in the DTNs. As mentioned above (Assumption 6), each packet has a given lifetime. A packet's priority decreases over deliveries. When this important value is greater than a predefined threshold, there is no opportunity for this packet to be forwarded to another node. Moreover, if the current node cannot contact the destination, this packet must be locked in the node buffer. This phenomenon increases a node's computational time (i.e., sorting buffer list) and data traffic (such as swapping buffer information). The packet should be dropped if the value of its timer is larger than its expiration time (such as TTL).

Counter: The counter is used to record how many nodes the packet has traversed. Due to small world properties [51], the average path length for social networks is much shorter than expected. It means that the packet should be passed though a limited number of intermediate nodes and then arrives at the destination. If the counter value is larger than predefined threshold, the packet should be dropped since the packet is no longer important. The reasons are obvious: Either the destination node D is so isolated or inactive that no node is able to find and forward packets to it or D is out of range (such as D leaves DTNs). To reduce the data traffic and increase node's computational capability, such a packet should be dropped.

Priority: Each packet should have its own priority. When two nodes contact, each one will swap this value as well as destination. It is an important factor affecting the current node which determines whether it should be in forwarding candidate list.

Packet Content: It contains the information from source node S to destination node D. Usually it should be encrypted by S and not available for intermediate nodes. Only D could decrypt it.

5.4 Routing Architecture

The architecture of improved socially selfish routing (ISSR) has five parts. These parts are implemented in all nodes moving in the system. In this section, we will discuss functions

of each part, address three problems given above, and finally give the entire architecture.

5.4.1 Packet Dropping Manager

As mentioned in Section 5.3 (packet structure), if a packet counter value is larger than predefined number or it exists in system too long, the current carrier should drop such a packet in order to reduce data traffic and save computational time. This work could take place during the travel time by packet dropping manager (PDM).

After each turn of packet delivery, a node has some new packets in the buffer. It could sort all packets according to their timer in increasing order and this process takes $O(n \log n)$ time (suppose there are n packets in the buffer). After sorting, the node checks each packet's timer and counter one by one. A packet will be dropped if its counter value or timer values is larger than predefined baseline, and then the node verifies the next packet. The process could be stopped when the node confirms a packet's timer value is over expiration time. This packet and others with higher value would be dropped. In the worst case, it takes $O(n)$ time to check all packets in the list. The overall time used to complete the task including sorting and check is $O(n \log n)$. Obviously, sorting and counter check occur only at the beginning of each inter-contact time, and timer check happens at every time unit. Moreover, the PDM also records how many packets it dropped.

5.4.2 Object Selecting Manager

When a node N detects several nodes within its contact range, the idealized solution for the issue displayed in Figure 5.1 is to ask all nodes within this range to sit down and have a meeting to figure out how to exchange their packets. There are two scenarios. In the first one (Figure 5.7), node N finds other nodes A, B, C and D within its contact range, and other nodes cannot detect each other. N asks other nodes not to move, since it has some packets for them respectively. Then N dispatches packets for A, C and D. After this step, N continues the regular process with other nodes one by one. During each regular

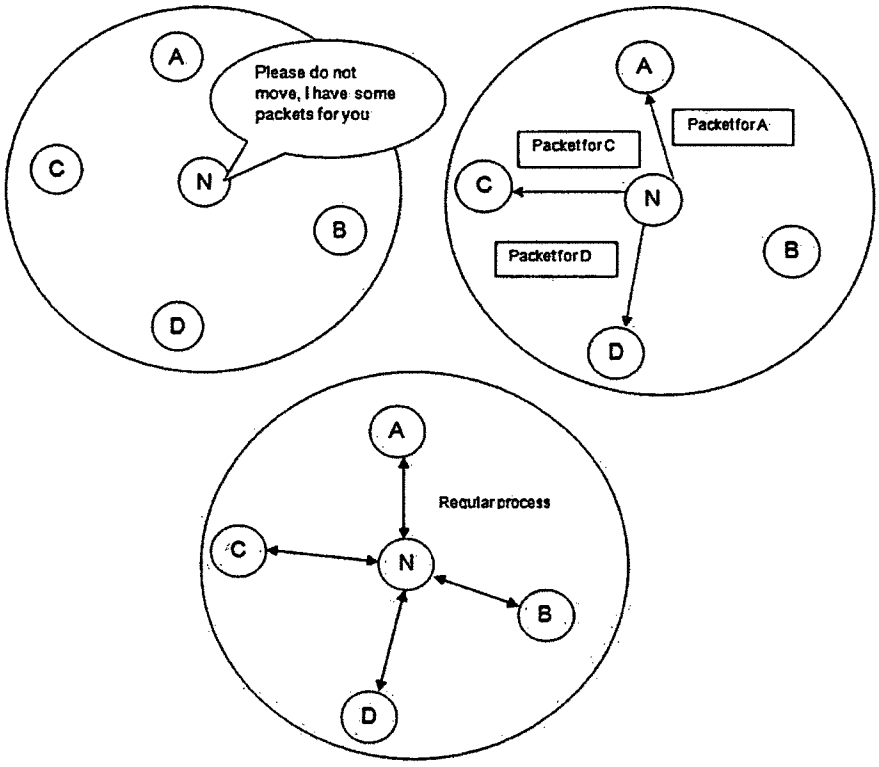

Figure 5.7: One solution solves Problem 1 with scenario that node N detects other nodes but others cannot detect each other.

process, N gets some packets with destination to it and exchanges some packets with A, B, C and D respectively. Since other nodes cannot move until N completes its tasks, the packets from A, B, C or D for N and from N for A, B, C or D will not be dispatched to undesirable receivers.

In scenario 2 (Figure 5.8), node N, A, B, C and D detect each other within some range. N holds a token and asks other nodes whether they have some packets with destination to N. Other nodes can dispatch these packets to N if they have. After that, the token passes to next node D. Then D has the same request and so on. When token goes through all nodes, each node gets its own packets. After that, N can swap packets with other nodes

one by one. Simultaneously, other nodes also do the same procedure.

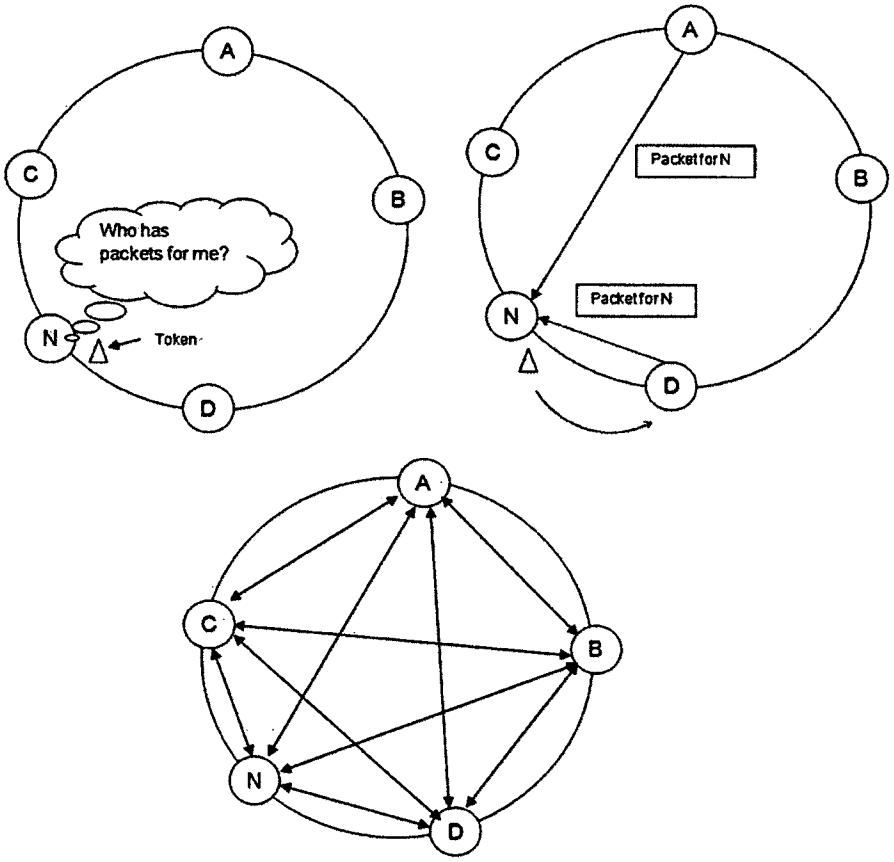

Figure 5.8: Another solution solves Problem 1 with another scenario two that nodes can detect each other.

In a more complex case that some nodes can detect each other (for example N, A and B) and other nodes (such as C and D) can only be detected by N, N should work as a bridge between two parts. However, these solutions seem Utopian. In the real world, some people do not stop in order to avoid answering strangers or others with whom they have weak social ties. Others might have some urgent things so that they cannot wait and leave early. To simplify this problem, it is assumed that any node can contact only one node

in its detecting list. For example, if N finds some nodes A, B and C in its detecting list, it can come into contact with only one of them (e.g., A). During the contact time, other nodes in N's detecting list may move to any other locations, contact each other if B and C can find each other, or do nothing but stay alone. After this round, all nodes (including N) will keep on moving.

The purpose of object selecting manager (OSM) in node N is to find a most suitable node in its detecting list. There are two factors affecting the selecting strategy:

- The number of packets for destination i in the detecting list is denoted by P_i where i is the i^{th} node in the detecting list and P_i is the number of packets for the destination i in the buffer.
- The willingness is denoted by ω_i where i is the i^{th} node in the detecting list and ω_i is the willingness of node i to N.

The first factor ensures the routing performance and second factor satisfies the social selfishness. The selecting strategy is to discover a balance point which obtains desirable routing performance and maintains reasonable social selfishness. A node N has a list based on the messages in its buffer. The list (S) records messages' destinations and the number of messages corresponding to the destinations. When N detects some nodes within its contact range, it sorts these nodes in list D according to the willingness. N calculates the weight of nodes in list D and then selects a node with largest weight as its forwarding object. Details are described in Algorithm 1. The weight of nodes in detecting list determines which node is selected as the forwarding object. The node with largest weight will be selected as the next forwarding object.

Definition 1 The weight of a node i (W_i) is the product of the number of packets (P_i) with a destination i in the detecting list and the willingness of node i to N (ω_i):

$$W_i = P_i \times \omega_i$$

Algorithm 1: Selecting a Forwarding Object

```
begin
    Sort packets by destinations into increasing order in list S
        with format: <Key: destination, Value: packet number >
    Estimate and sort (decreasing order) the willingness for nodes in detecting list D
        with format: <Key: node name, Value: willingness >
    Create a new element e with format: <Key: null, Value: 0 >
    for node i ← 1 to |D| do
        if there exists node j in S such that i.Key == j.Key then
            weight = i.Value × j.Value
            if e.Value < weight then
                e.Key = j.Key; e.Value = weight
        else
            continue
    if e.Key == null then
        return first element in D
    else
        return e.Key
```

In bidirectional model, a packet selected as forwarding candidate from node N to next hop M is mostly determined by M's willingness to N and D (the destination). If M's willingness to D is too small, this packet has low opportunity to be forwarded form N to M. This means for any two packets with different destination D_1 and D_2 in node buffer, there is no significant difference between the willingness from this node to D_1 and D_2. When we use forwarding object selection strategy (weight of a node in Definition 1) to select forwarding object, the bidirectional model helps to avoid some unreasonable case (such as $W_1 = 1 \times 0.9$ and $W_2 = 8 \times 0.1$) in which N selects D_1 as the forwarding object.

In another case ($W_1 = 2 \times 0.8$ and $W_2 = 4 \times 0.6$), D_2 as forwarding object seems more reasonable.

If there is no packet for any node in the detecting list, the node with highest willingness is set as the forwarding object. If weights of two or more nodes are equal, any one of them could be considered as the forwarding object. To sort list S and list D in algorithm 1, we totally use $O(|D|\log|D| + |S|\log|S|)$ time. The time used to calculate the weight of each node is $O(|D|)$. The time complexity of this algorithm is $O(|D|\log|D| + |S|\log|S|)$, where $|S|$ is the length of list in the first step of Algorithm 1 and $|D|$ is the length of list in the second step of Algorithm 1. It occupies a very small part of computing time due to the current system computational capability. In addition, OSM also records the detected nodes' names and their encountering times as well as the number of packets successfully received by their destinations.

5.4.3 Packet Priority Manager

After selecting the forwarding object (the next hop), node N and M deliver packets destined to each other. During the process, two nodes swap basic information of packets in the buffer including destination and priority. Then packet priority manager (PPM) calculates a priority for each buffered packet based on the willingness discussed in bidirectional model. This priority of a packet measures the social importance of the packet to the node.

A quick solution for Problem 3 is to allow intermediate node C to reassign priority if and only if the packet's priority is too low and C has strong social ties with predecessor. For example in Figure 5.4, the willingness from C to B is 0.8, much larger than 0.25 (C to A). To satisfy selfishness of C, it reassigns the priority (such as double the value) in order to make sure this packet has a high possibility to be received by its destination. The solution is simple, but it introduces new issues into the system:

- How to avoid abusing this strategy if it is allowed?
- How to reassign the priority in the reasonable and acceptable range?

Since selfishness is allowed in *DTNs*, any node should forward its favorite packets as much as possible. It could deliberately reassign some packets' priorities so that they occupy the front positions of a candidate list and then they are delivered to the next hop. In addition, node N also can return some falsified information to predecessor P so that N will get some desirable packets since P prepares forwarding candidate list according to the feedback from N. If the phenomenon is ignored, a lot of packets could not reach their destinations if everyone only delivers his own favorite packets.

Before answering these two questions, another scenario in Figure 5.9 would be discussed in advance. Normally, a very important packet has high forwarding possibility to the next hop, which means this packet would easily traverse the social network. It results in that the packet could either be received by the destination quickly or be blocked by some intermediate nodes because of its priority decreasing sharply via several nodes due to the formula $p_i = p_{i-1} \times \omega$ mentioned in problem 3.

Example 5 *Priority of an important packet decreases sharply via several nodes (Figure 5.9).*

A packet travels from node Y to H through node D, F, A, B and G. At the beginning, its priority is 0.9, when it arrives at node G its priority reduces to $0.8 \times 0.7 \times 0.6 \times 0.6 \times 0.9 \times 0.8 = 0.145$. *Another packet from Y to H is via node E and G, and its priority chain is* $0.4 \times 0.8 \times 0.5 = 0.16$. *If only one packet could be sent to destination, packet 2 dominates packet 1. In the real world, Y may inform next hop the packet's importance if the packet is really important. Usually, during the process of packet delivery, the notification may be handed on although packet's importance must decrease. Normally, an important packet has high privilege and its priority should not violently decrease even if it traverses several nodes.*

The solution is to apply a smart strategy to calculate next hop priority. Formally:

$$p_{ij} = \sqrt[\omega_{ij}]{p_{(i-1)j}}$$

where p_{ij} is the priority of packet j in i^{th} hop buffer and ω_{ij} is the i^{th} hop's final willingness

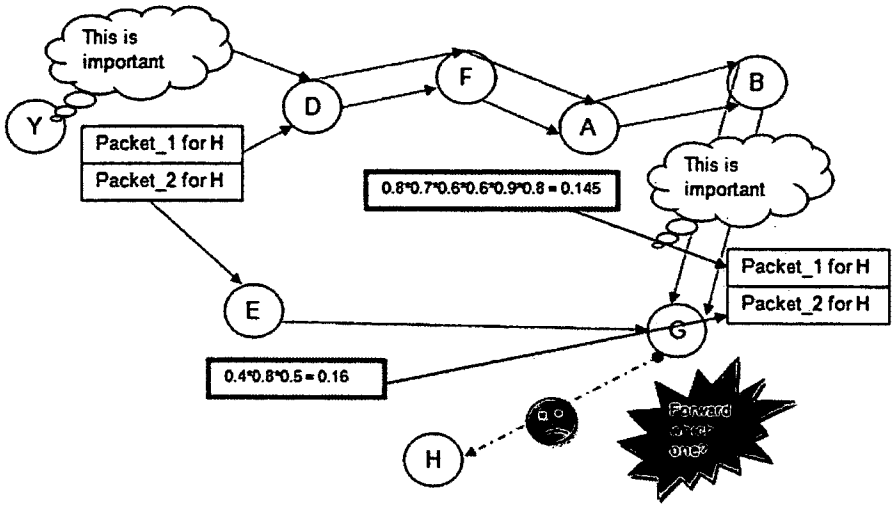

Figure 5.9: Priority of an important packet decreases sharply via several nodes.

to forward the packet j from the $(i-1)^{th}$ hop. We need to define final willingness ω used in formula.

Definition 2 Suppose the destination of packet j is D; willingness of hop i to $i-1$ is ω_1 and willingness of i to D is ω_2 respectively. The final willingness used in packet priority manager is defined as the sum of ω_1 and ω_2. Formally:

$$\omega_{ij} = \alpha \times \omega_1 + \beta \times \omega_2$$

where α and β are two numbers in $[0, 1]$ and $\alpha + \beta = 1$.

For Problem 3 in Figure 5.4, the priority chain recalculates with new formula, and results are 0.9^0.8^0.8^0.7^0.25 = 0.39035 (the packet from node W) and 0.8^0.6^0.5^0.5^0.8 = 0.155746 (packet from node Y). At the beginning, two packets have almost identical priorities. When they are received by node C, the difference is obvious. Although the former packet encounters a low willingness at the last step, it still keeps acceptable value since any other willingness on its priority chain is really high. The willingness of the latter packet on the chain is not remarkable, so its priority decreases faster despite the last step

willingness is much higher. Because of the difference, node C will not hesitate about the choice between the two packets. It avoids the simply solution mentioned above.

New formula prevents high priority from decreasing sharply after traversing long distance (scenario in Figure 5.9) and bidirectional model ensures that the case confronting low willingness is a rare occurrence (Problem 3 in Figure 5.4). In other words, there is no necessary to apply the simple solution. Since issue 1 about how to avoid abusing simple solution is eliminated, issue 2 about how to reassgin the priority is no longer a problem.

5.4.4 Delivery Probability Estimator

A packet selected into forwarding candidate list is determined by two factors. One is its priority (solved in Section 5.4.3) and the other is delivery probability. In this subsection, an estimator is used to quantify the node's forwarding capability, known as delivery probability for each packet.

The method for measuring the delivery probability of a node is based on the likelihood of its contacting with the destination. The chance could be estimated according to the detected nodes list. If a packet's destination is found in the historical record (OSM records the detected nodes' names and their encountering times), there is a high possibility for this node to meet such a destination another time. Otherwise, the contact opportunity should be low.

Suppose the next contact between node N and destination D happens at time T_D, N has to drop packet a due to counter value at time T_C or due to expiration time T_E. So the overall delivery probability should be:

$$P_{delivery} = (1 - P\{T_C \leq T_D\}) \times (1 - P\{T_E \leq T_D\})$$

Normally, a packet with high priority has more opportunities to be delivered. Every time a packet traverses a node, its counter value increases by one. The packet will be dropped when its counter value is larger than predefined threshold. There are two reasons for such a result. The first one is that the destination D is too silent during some periods. In

addition to inactivity, its location is so isolated that other nodes rarely reach that place. This scenario is very common in current society. Some persons disappear during some periods, and none of their friends is able to contact them even if popular communication tools are available. The other possible reason is that D is no longer in DTNs, which means it could not be found although it was spotted several times by other nodes sometime before. When $T_E \geq T_D$, $P\{T_C \leq T_D\} \leq P\{T_C \leq T_E\}$ because such a packet would be dropped before its time expired. Then:

$$P_{delivery} \geq (1 - P\{T_E \leq T_D\}) \times (1 - P\{T_C \leq T_E\})$$

$P_{delivery}$ is determined by two independent droppings. The first one is called expiration dropping probability (EDP) $P_{exp}\{T_E \leq T_D\}$ which means the packet expires before node N could contact with D. The second is $P_{cdp}\{T_C \leq T_E\}$ which means the counter value of packet is above threshold before its expiration, namely counter dropping probability (CDP).

Let random variable X denote the inter-contact time between N and the destination D. Assume that each inter-contact time is independent, and $\tilde{t}$ is the most recent contact time between N and D before the estimation time T:

$$P_{exp} = P\{T_E \leq T_D\} = P\{X > T_E - \tilde{t}\} \geq E(X)/(T_E - \tilde{t})$$

It means that nodes have a lower expiration dropping probability if it has a lower average inter-contact time with the destination. To reduce dropping rate, all nodes should exchange packets in reasonable limited time and also improve their contact frequency. The best encountering pattern in Figure 5.10 [48] is *c*. This is the macroscopic view for packet delivery probability. The next step is to simplify the delivery probability and quantify it.

Since OSM records all nodes detected in previous encounters, delivery probability estimator (DPE) will estimate the probability of a packet forwarded to its destination based on the factor whether the packet destination D could be found in the detecting list. If the node encounters D several times in its historical record, it should have high possibility to

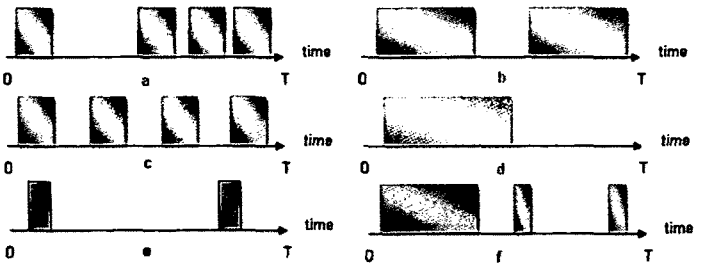

Figure 5.10: Various encountering patterns between node i and j.

meet D again and forward this packet to it. Otherwise, the node may not run into D in the future and then the packet will be dropped due to the expiration time. In the real world, some people always meet at some places since their regions have overlaps, but some others rarely or never encounter because they live in different communities, do not share the same hobbits and so on.

Definition 3 Suppose node N has a historical record H which lists the nodes it encountered before as well as the times it encountered for each node in the record: with format <Key: node name, Value: encounter times >. The probability (P_{NM}) that N runs into node M (if M in this record) next time is the times (E_{NM}) N encountered M before over the total encounter times of all nodes in the record (suppose there are K nodes including M in this record). For each node j in the record H, $j.key$ represents the name of node j and $E_{Nj.key}$ is the times N met j before. Formally:

$$P_{NM} = E_{NM} / \sum_{j \in H}^{k} E_{Nj.key}$$

For any node A not in the list, the probability that N comes across A is considered as $P_{NA} = 1/(2 \times \sum_{j \in H}^{k} E_{Nj.key})$.

Example 6 *The estimation of encounter probability.*

Node N has a historical record of encountering nodes: { <*Key: node A, Value: 2*>, <*Key: node B, Value: 3*>}. *The probability that N runs into A in next time is:* $2/(2+3) =$

40%. *If N comes across C in next time, new record is* { <*Key: node A, Value: 2* >, <*Key: node B, Value: 3* >, <*Key: node C, Value: 1* >}. *The probability that N encounters A is changed to:* $2/(2+3+1) = 33.3\%$. *Since node D is not listed in the record, the probability that N meets D is* $1/2 \times (2+3+1) = 8.3\%$.

This is an intuitive formula according to the explanation mentioned above that some people always meet but some never. In the initial status, since no node encounters any other node, the probability is set by system as $1/|L|$, where L is the list of all nodes in DTN and $|L|$ is the total number of nodes in the list. Encounter probability is a real number within $(0, 1]$.

People can quantitatively rate their friendships [20], which can be represented as the forwarding willingness used to set packet's priority. People also could evaluate their ability with several levels (such as strong, qualified, average, unsatisfied and none). This illustrates whether people have capability to accomplish their tasks. Normally, a packet in node N's buffer has four kinds of fates from the end of the last contact time to the beginning of the next contact time. The packets could be dropped by any reason, received by a destination, forwarded to the next hop or retained in the buffer. We can denote $|d|$ as the number of packets dropped by node N, $|s|$ as the number of packets N successfully delivered to the destinations, $|f|$ as the number of packets N forwarded to other nodes and $|r|$ as the number of packets remaining in the buffer respectively.

Definition 4 The capability of a node can be quantified as the percentage of successfully delivered packets. Formally:

$$Capability = \frac{|s|}{|d| + |s| + |f| + |n|}$$

It illustrates the idea that the node is more capable if it has a higher successful delivery rate. This could be achieved by OSM which records the number of packets successfully received by destination and by forwarding strategy maker which records how many packets are forwarded by contacting nodes. Forwarding strategy maker will be discussed in Section

5.4.5. DPE could fetch the data and figure out the node's capability. At the beginning of the stage, system will set initial capability value for all nodes as 1. Capability is a real number within $[0, 1]$, where 0 means the node never forwards any packet to its destination and 1 means the initial status.

After the concepts of encounter probability and capability are defined and discussed, a packet delivery probability can now be simplified. Any packet will be received by destination if and only if the following two conditions are satisfied:

1. Destination will be met in the next contact time.
2. Packet should not be dropped because of expiration time or its counter value and it could not be forwarded to any other nodes.

Definition 5 A packet i's delivery probability is quantified as the product of carrier encounter probability and capability:

$$P_{delivery-i} = P_{Ni} \times Capability$$

where P_{Ni} is the probability of node N encountering the destination of the i^{th} packet in the potential forwarding list and capability of N is the same value to all packets at present.

The value of a node's capability may fluctuate in the next turn of estimating delivery probability, since forwarding strategy maker (FSM) will increase total number of forwarded packets at the end of this contact, OSM will adjust the number of successfully forwarded packets at the beginning of the next contact and PDM also changes the number of packets it dropped.

After calculating capability, node N prepares its detecting record (K) according to the nodes and times it met before. This step is to estimate the probability that N meets a node in record K next time. For each packet in the candidate list, its delivery probability is determined by encounter probability (in Definition 3) and capability (in definition 4). The detail steps how to calculate delivery probability is summarized in Algorithm 2.

Algorithm 2: Estimating delivery probability

begin

calculate capability (Single step)

the historical encounter record K is ready with format:

<Key: destination, Value: encounter probability >

for *node* $i \leftarrow 1$ **to** $|L|$ *(L is the potential forwarding list)* **do**

let j as $i.destination$

if *find node name* j *in* $|K|$ **then**

$i.probability = P_{Nj} \times capability$

else

$i.probability = 1/(2 \times sum\_of\_K) \times capability$

return list of $i.probability$

This is a general description of estimating delivery probability. The initial cases of encounter number and capability could be implemented when the system starts to work. The detecting list K is pre-sorted by OSM in the initial stage. The insertion of a new detected node or modification of an existing value takes $O(\log |K|)$ time. To find a packet destination in this list also uses $O(\log |K|)$ time. The time complexity of this algorithm is $O(|L| \log |K|)$, where $|L|$ is the length of the potential forwarding list and $|K|$ is the length of the historical encounter record.

5.4.5 Forwarding Strategy Maker

In the real world, people cannot get help from some friends because of their abilities although they are very willing to support. On the other hand, people rarely ask for help from some reluctant people despite their competence in solving problems. People really hope to get help from reliable people. This means that the reliable people are willing to help you and also able to completely fulfill the task.

The reliability mapping into the routing architecture in DTNs is represented by packet priority and delivery probability. Priority illustrates the willingness of the next node to forward corresponding packets. Delivery probability demonstrates whether the next node could achieve expected results or not. After receiving the feedback from the contacting node (including destination, new priority, and delivery probability), the first step of forwarding strategy maker (FSM) is to calculate the next hop's reliability for each packet.

Definition 6 The reliability of next hop for packet i in current carrier buffer is the product of new priority of packet i and delivery probability of packet i:

$$R_i = p_i \times P_{delivery-i}$$

where R_i is the reliability of the next node for the i^{th} packet, p_i is priority of packet i and $P_{delivery-i}$ is its delivery probability.

According to the reliability, a node establishes a list of candidate packets that should be forwarded to a better relay. The next step of FSM is to greedily transmit them in the limited time. Obviously, bandwidth will be wasted if the transmitted packet is dropped due to its size and inadequate time. For example, only five seconds remain for transmission, the speed is $10KBps$ and the file size is $55KB$. The transmission process will be broken since the contact ends and two nodes leave for other places. To address this issue, the FSM should optimize the forwarding list.

Available forwarding size L_{size} is determined by contacting time and bandwidth. Suppose there are n packets in the set C; each item i in C has a value of reliability R_i and a packet size L_i. Let A_i denote whether the i^{th} packet is selected into delivery subset ($A_i = 1$) or not ($A_i = 0$). The problem can be formulated as:

$$Max \sum_{i \in C} R_i A_i \quad s.t. \quad \sum_{i \in C} L_i A_i \leq L_{size}$$

The Knapsack problem is an NP problem [52]. Thus, a dynamic programming algorithm is given to solve the issue in the decreasing order of reliability over packet size. The basic

idea is to find the final solution in terms of sub-problems. This means we create a table, use predefined recursive formula (in Algorithm 3) for sub-problems and fill value in the table. The final solution is based on the trace of maximum value in the table. The details are shown in Algorithm 3.

Algorithm 3: Optimizing forwarding list

begin

Compute the reliability R for each packet in C

Sort C in the decreasing order of R_i/L_i (Let i denote the i^{th} packet in C)

for *packet* $i \leftarrow 1$ **to** $|C|$ **do**

  for $l \leftarrow 0$ **to** L_{size} **do**

    if $l[i] \leq l$ **then**

      $R[i,l] = Max\{R[i-1,l], R[i] + R[i-1, l-l[i]]\}$

      $Keep[i,l] = 1$

    else

      $R[i,l] = R[i-1,l]$

      $Keep[i,l] = 0$

$L = L_{size}$

for *packet* $i \leftarrow |C|$ **to** 1 **do**

  if $Keep[i,L] == 1$ **then**

    record i in Trace

    $L = L - l[i]$

return the Trace

It takes $O(|C|)$ time to compute the reliability for each packets in the list C and $O(|C| \log |C|)$ time to sort list C. To optimize the forwarding list, we use $O(|C|L_{size})$ time. When L_{size} is much larger than the length of list C, the time complexity of this

algorithm is $O(|C|L_{size})$ which is acceptable. $|C|$ is the length of potential forwarding list in the buffer.

5.5 The Protocol

After new models, assumptions and definitions are described, three problems are solved in previous sections. We will summarize the detailed steps regarding messages forwarded between two nodes (N and M) in this section. The architectures are implemented in all nodes moving in the system. Only node N is described how it determines which packets to transfer to node M. Generally, node M operates in a similar manner. Figure 5.11 is used to illustrate how routing works, Algorithm 4 abstracts the process describing our routing and protocol provides detail steps:

Protocol 1 : The protocol of improved socially selfish routing

(1) During the travel time, PDM drops the packets due to their timer value or counter value larger than predefined threshold. PDM also records the number of dropped packets.

(2) When neighbours are discovered, node N may select one node M, from the detecting list of neighbour nodes based on the strategy of OSM. The purpose is to find a balance point which obtains desirable routing performance and maintains reasonable social selfishness.

(3) Two nodes deliver packets destined to each other in the decreasing order of priority. OSM records how many packets are successfully forwarded to the destination.

(4) They exchange information related to their willingness to forward. The information is the list of basic status of packets in the buffer. The format is <destination, priority>.

(5) PPM in N calculates the new priority value for each packet in node M based on the bidirectional model.

(6) DPE in N also calculates the delivery probability for each packet according to its capability and encounter probability.

Algorithm 4: The Process of Routing

```
begin
    while System running do
        if the travel time then
            if packets'TTL > threshold then
                Packet Dropping Manager (PDM) drops packets
                PDM records the number of dropped packets
        if detect other nodes then
            Object Selecting Manager (OSM) selects the most suitable forwarding
            object
            OSM records the number of successfully delivered packets
        if contact time then
            Packet Priority Manager (PPM) calculates packets' new priorities
            Delivery Probability Estimator (DPE) determines delivery probability
            Forwarding Strategy Maker (FSM) calculates reliability
            FSM prepares forwarding candidate list
            Switch packets in limited time
        Pause for a while and move again
```

(7) After calculation, two nodes swap their results with format <destination, new priority, delivery probability>.

(8) Node N figures out the reliability for each packet.

(9) FSM optimizes the forwarding candidates list based on the reliability, packet size, and available forwarding size.

(10) Node N forwards packets to node M.

— End of Protocol —

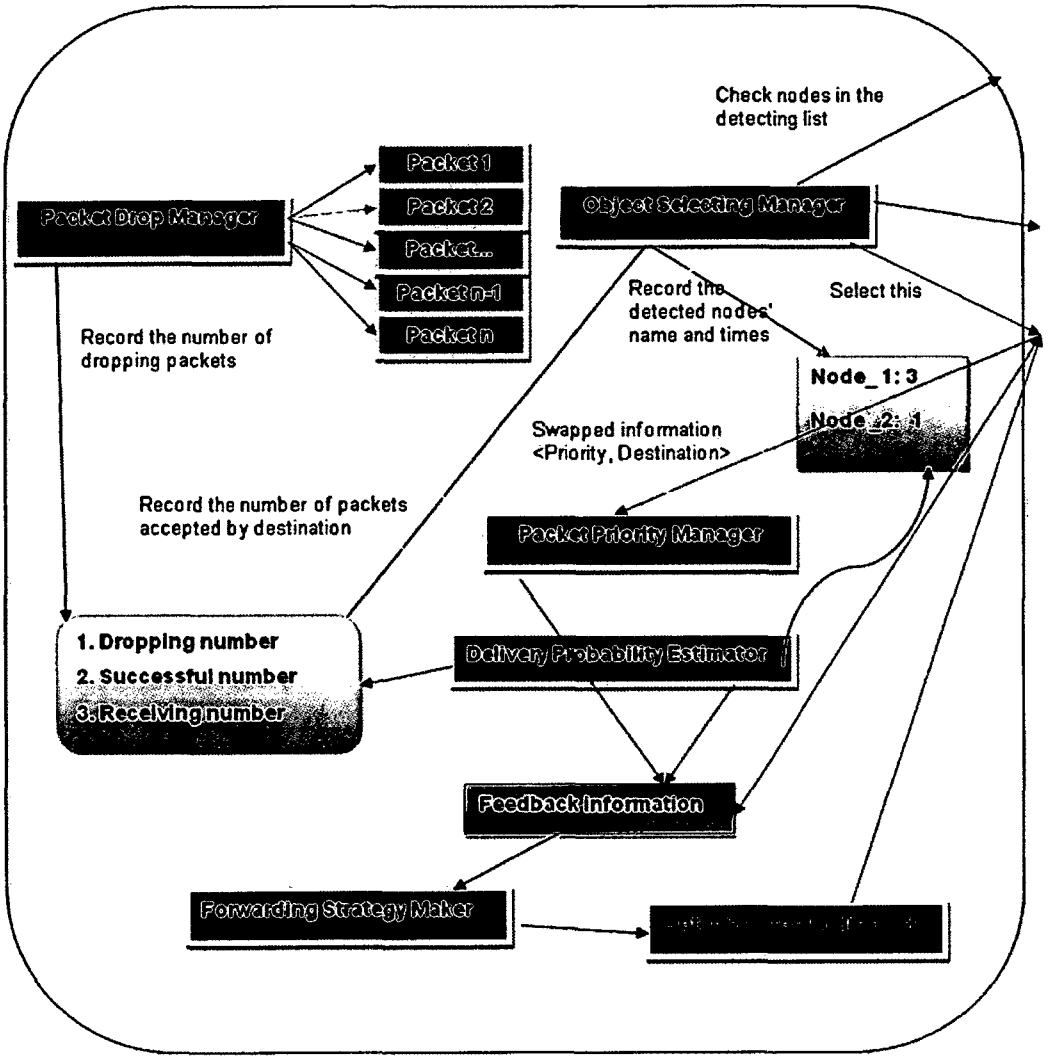

Figure 5.11: The summary of routing architecture.

5.6 Summary

In this chapter, we discussed three problems in detail and then proposed an improved socially selfish routing to solve these problems. We made assumptions, constructed models

and described the architecture of the routing. In the next chapter, we will introduce simulation environment and mobility pattern.

CHAPTER 6

MOBILITY PATTERNS AND SIMULATION

In this chapter, we describe the simulation environment including the mobility models and discuss details of the implementation.

6.1 Mobility Models for Ad Hoc Network Research

A mobility model which represents the mobile nodes (MNs) is used to simulate a new protocol for an ad hoc network for the purpose of protocol utilization. This approach would finally determine whether the protocol is functional or not. The most used mobility models can be grouped into two types according to these models properties: traces and synthetic models [53] [54]. According to the ad hoc network property, if there is no trace created in simulation environment, the network is difficult to model. Synthetic models are usually applied to mimic the behaviours of mobile nodes with different directions and speeds.

In addition to entity mobility models which have been used in the performance evaluation of an ad hoc network, there are some synthetic group mobility models. Group mobility modes simulate environment in which the movement of an MN in one group relies on the other MNs in this group [54]. For example, a task is assigned to a group of ships which might scout a particular plot of sea in order to avoid enemy invasion or work together in a cooperative manner to accomplish a common goal. The following is the list of group mobility modes:

- Exponential Correlated Random Mobility Model
- Column Mobility Model
- Nomadic Community Mobility Model
- Pursue Mobility Model

6.1.1 Synthetic Models Overview

As mentioned above, trace models are the mobility patterns which are observed in real life systems, and they provide accurate information. Given the limitation of trace models in simulated environment, synthetic models are used as replacements to represent MNs' behaviours. The most used synthetic models are listed below:

- Random Walk Mobility Model: simple approach with random directions and speeds
- Random Waypoint Mobility Model: the combination of pause time and random walk
- Random Direction Mobility Model: running MNs are forced to travel to the edge of the simulation area before changing direction and speed
- A Boundless Simulation Area Mobility Model: MNs are simulated in a tours-shaped area converted from a 2D rectangular area

6.1.2 Random Walk Mobility Model

The random walk (or Brownian motion) was described mathematically by Einstein in 1926 [53]. In [55], it was implemented to mimic unpredictable movement. During the movement, an MN walks for either a constant time interval t or a constant distance d. When an MN arrives at the destination, the new direction and speed are predefined in the range of $[0, 2\pi]$ and $[min, max]$ respectively. If an MN reaches a simulation boundary, it returns with new direction based on the incoming direction. Moreover, it must be a certainty that a random walk on a one or two-dimensional surface will return to the origin [56], which means the movement of an MN is around its starting point. If the specified time or distance is short, the movement of an MN is restricted to a small portion of the simulation area (Figure 6.1 reproduced from [54]), which is called a random roaming pattern. Random walk mobility model is suitable to evaluate the performance in a semi-static network, in which an MN's direction or speed has a small value [54].

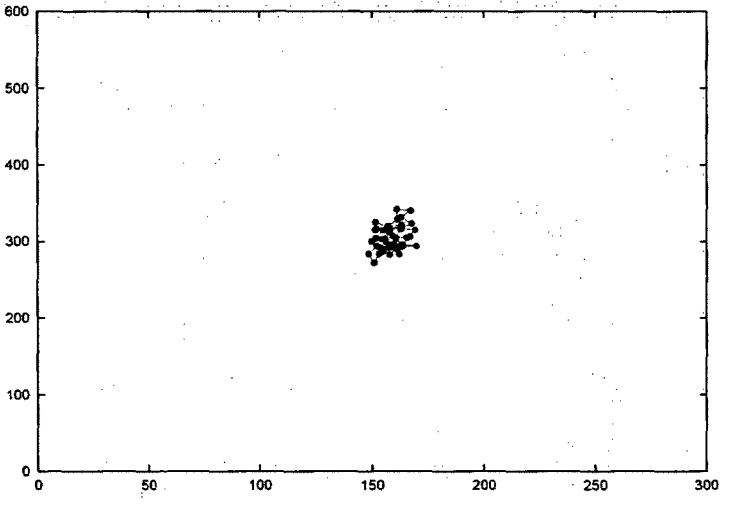

Figure 6.1: Traveling pattern of an MN using the 2-D Random Walk Mobility Model (an MN walks for a constant distance).

The memory-less nature of the random walk mobility model makes it unsuitable to

represent the behaviour that an MN travels with a destination in mind, since the MN's direction and speed in the future is not correlated with its current direction and speed [57] [58].

6.1.3 Random Waypoint Mobility Model

The difference between random waypoint mobility model and random walk mobility model is that when an MN changes its direction or speed, it will pause for a specified time before starting the process again [59] (Figure 6.2 reproduced from [54]). This model is a widely used mobility model.

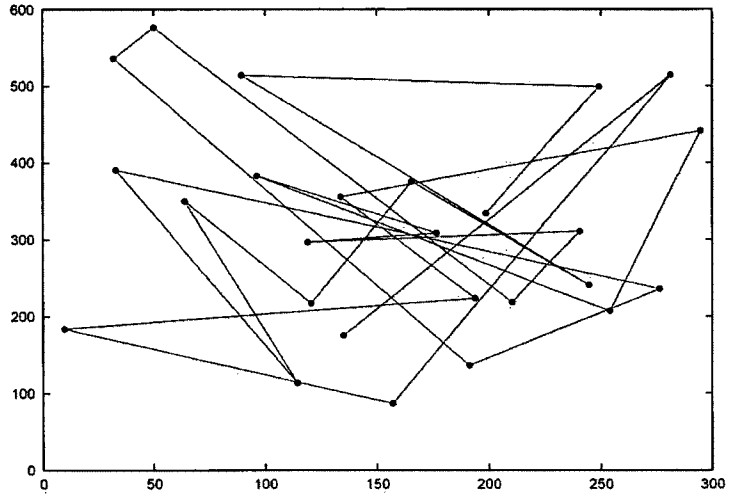

Figure 6.2: Traveling pattern of an MN using the random waypoint mobility model.

In the random waypoint mobility model, the MNs are initially distributed randomly around the simulation area, but this initial random distribution of MNs is not a representative of the manner in which nodes distribute themselves when moving. A neighbour of an MN is defined as a node within the MN's transmission range. The average MN's neighbour percentage could be considered as the cumulative percentage of total nodes within a given MN's range. As shown, there is high variability during the first period of simulation time,

which causes high variability performance unless long simulation time is applied [54].

To avoid the high variability issue, there are three possible solutions presented and discussed in [54]:

- Discard the initial simulation time (long enough) according to MNs' speed to ensure that the initialization problem is removed
- Save the locations of the MNs after initial simulation time (long enough) and use these positions as initial starting points of the MNs
- Distribute the MNs in a manner (such as a triangle) that maps to a distribution more common to the model

In [60] [61], the relationship between MN's speed and pause time is illustrated that long pause time constructs a stable network even at high speed. In this model, the pause time acts as a proxy for the degree of mobility in a simulation; longer pause time amounts to more nodes being stationary for more of the simulations. In this model, the results of a multicast protocol are only valid in the scenario that the performance is evaluated with low MN's speed and long pause time [62].

In this model, an interesting phenomenon is observed. Since an MN must select a destination after each pause time, it is most likely to travel in the direction in which there are the most destinations from which to choose. It causes clustering near the center of the simulation area: either the selected destination or the travel through. This characteristic creates a situation in which nodes converge, disperse, and then re-converge in the center of the area. It is called density waves and fixed in [63].

In this simulation, random waypoint mobility model will be applied. So we skip the details of other mobility models. These models are discussed in [63] for random direction mobiles model and in [64] for a boundless simulation area mobile model.

6.2 Implementation

This simulation is developed under Sun Java JDK-6 64 bit environment. All methods were implemented and visually demonstrated.

6.2.1 Program Structure

Figure 6.3 illustrates the process of application implantation. This process follows the object-oriented principles [65] [66] such as open-closed principle (OCP) and single responsibility principle (SRP).

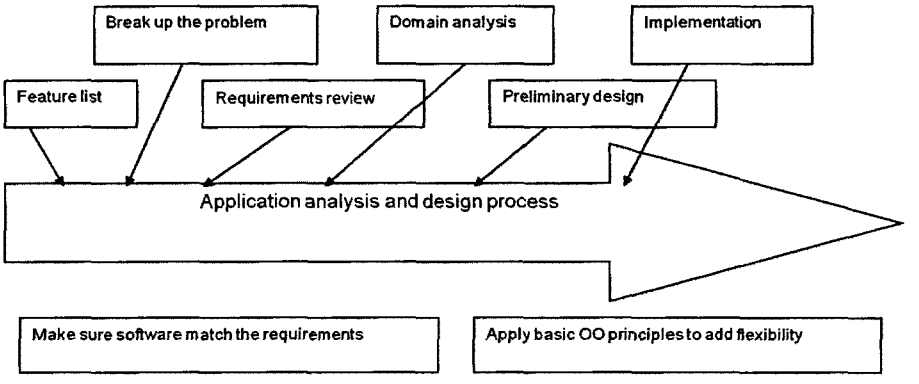

Figure 6.3: The roadmap of application implementation.

This application is the future driven development. All features listed below are picked up from the application, planned, analyzed and completed:

- The framework supports different sizes of node-roaming areas.
- The framework supports different number of nodes in predefined area.
- The framework coordinates the random waypoint mobility model for nodes.
- The nodes could generate different number of messages with random destinations.

- A node is able to detect other nodes and swap messages with the most suitable one when it is paused.
- The framework provides final results after application stops running.

6.2.2 Application Overview

The application was developed by the model, in which nodes and edges made up the graph. The basic idea for representing the graph is to keep only a collection of all nodes, which allows quick access to nodes, but if it is ever needed to get all the edges (such as willingness between two connected nodes), the collection of edges should be built up. During the simulation, all nodes are randomly created in the predefined area (Figure 6.4) and all messages destinations are also random. For each movement of nodes in the graph, we need to create a separate thread.

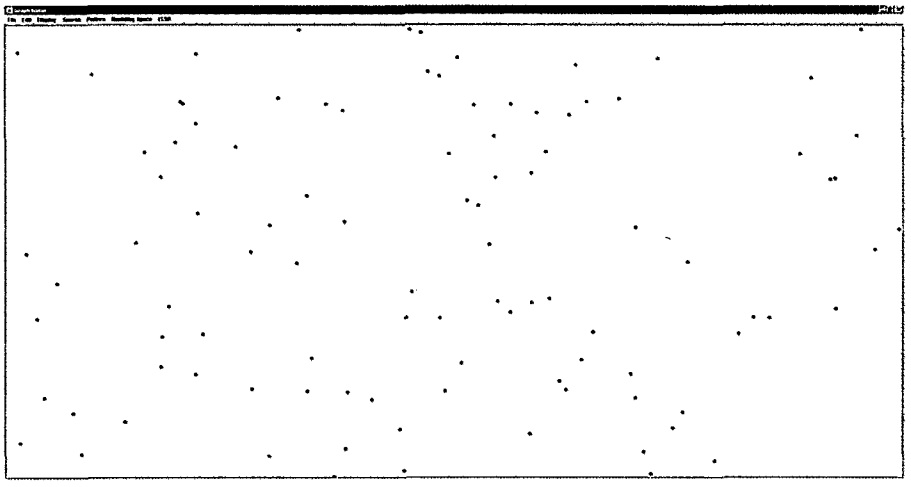

Figure 6.4: 100 nodes are randomly created in the predefined area.

6.3 Summary

In this chapter, we reviewed basic concepts of mobility models and discussed two models in detail. The random waypoint mobility model was applied in our simulation. Then we represented the simulation environment and implementation. In the next chapter, we will simulate and evaluate the performance of our improved routing and compare results to another socially selfish routing (SSAR).

CHAPTER 7

SIMULATION RESULTS AND DISCUSSION

In this chapter, we discuss our simulation setup and evaluate the performance of our improved socially selfish routing in two different parameter size settings. The performance of our new routing is compared to another socially selfish routing (SSAR).

7.1 Simulation Setup

7.1.1 Test Environment

There are two test environments used in this simulation. Usually, routing is simulated in the stable environment. This means we should eliminate any factor having side effect such as the forwarding capacity. For this reason, we created and optimized simulation environments with busy status in order to observe the behaviours of two routings and evaluate their performance. So there are 10 nodes in small test environment and each has 50 initial messages. In another test environment, 100 nodes are created and each has

500 initial messages. Since the generated messages have completely random destinations, in order to quickly distribute these messages, the setting of a node speed makes sure it could cross about 10% of the area of simulation environment and then encounter as many different other nodes as possible. Moreover the speed should be set in a reasonable range. For example, the small test environment looks like a university in which mostly people walk around. The speed in the small sample is much less than that in large one which can be viewed as a small town in which mostly people drive. After each node moves around the area in a unit time, it will pause for 3 unit time (in small test environment) or 5 unit time (in large test environment), which makes sure all nodes have long enough pause time in random waypoint mobility model. All nodes have the same detecting range (the radius is 50) in both test environments. Since SSAR is designed without considering encounter number, we modify the SSAR with introducing the concept of encounter number for fair comparison. Table 7.1 lists in detail settings for the simulation. We use student t-test to derive 95% confidence intervals [67] and Prism [68] to analyze test results.

Table 7.1: Test Environments.

	Environment 1	Environment 2
Simulation area	400×340	1920×990
Nodes number	10	100
Initial messages number	500	50000
Node speed	$[0, 100]$ / per unit time	$[0, 500]$ / per unit time
Detecting range	50	50
Pause time	3 unit time	5 unit time

7.1.2 Performance Metrics

The major purpose of routing algorithms for DTNs is to optimize system performance. Two routings will be compared with following test items:

- Delivery ratio with different TTL: In this test, we calculate the proportion of messages which are delivered to their destinations among the total messages generated. During the test, we also assess the performance with the increase of messages' TTL.
- Delay: The cost is the average number of forwards done per successfully delivered message. Obviously, the desirable routing should have higher performance with the lower cost.
- Stress Test (the effects of workload): we also evaluate the performance of two routings under higher workloads with different packet generation rate. In environment 1, each node generates from 1 message per unit time to 3 messages per unit time. In environment 2, each node generates from 1 message per unit time to 10 messages per unit time. All messages are generated when nodes are moving. When in pause time, no additional messages are created.
- Dropping rate: we estimate the messages' dropping rate with different predefined counter value in a fixed TTL.

7.2 Test Results

7.2.1 Delivery ratio

In environment 1(Table 7.2 & Figure 7.1), the number of initial messages is 500. We change the messages' TTL from zero to 25 time units in order to observe the effects of TTL on the message delivery ratio. To simplify the calculation, a message's TTL ignores the pause time. Moreover, we focus on the effects of TTL and ignore the dropping because of counter

value (defined in Section 5.3). We repeated each simulation 20 times (same in other tests) with different random seeds and took the mean of each simulation as the results.

Table 7.2: Performance Test: Message Delivery Ratio in environment 1 with different message's TTL (t-test and 95% confidence).

TTL	SSAR	Our Routing
5	17.92% ± 1.56%	21.72% ± 1.41%
10	27.76% ± 1.85%	32.42% ± 1.93%
15	36.26% ± 2.23%	42.04% ± 1.99%
20	39.36% ± 2.68%	47.12% ± 1.2%
25	40.12% ± 2.47%	48.75% ± 1.43%

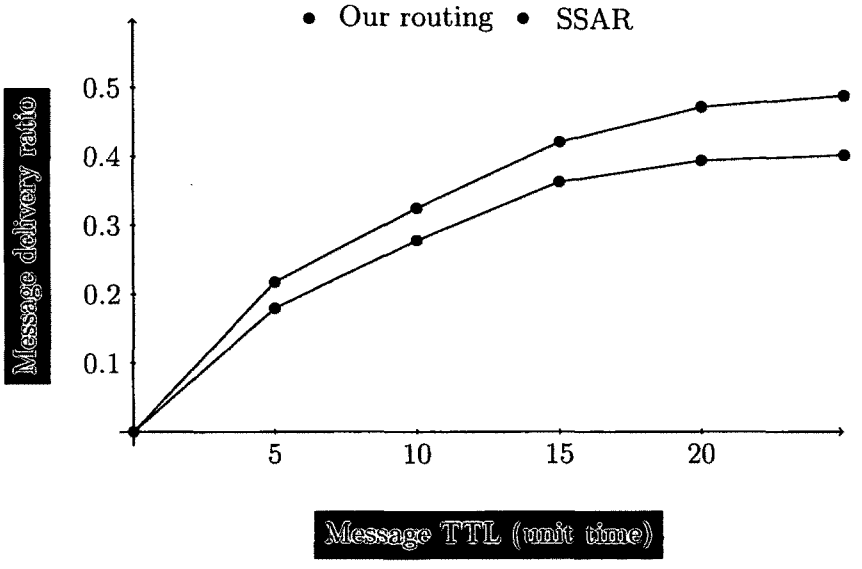

Figure 7.1: Performance Test: Message Delivery Ratio in environment 1 with different message's TTL.

In environment 2 (Table 7.3 & Figure 7.2), the number of initial messages is 50000. The

messages' TTL is changed from zero to 250 time units.

Table 7.3: Performance Test: Message Delivery Ratio in environment 2 with different message's TTL (t-test and 95% confidence).

TTL	SSAR	Our Routing
25	7.23% ± 0.12%	7.67% ± 0.05%
50	14.28% ± 0.04%	14.19% ± 0.05%
100	25.54% ± 0.27%	27% ± 0.23%
150	35.26% ± 0.22%	37.46% ± 0.24%
200	44.22% ± 0.15%	46.74% ± 0.22%
250	49.76% ± 0.14%	51.82% ± 0.09%

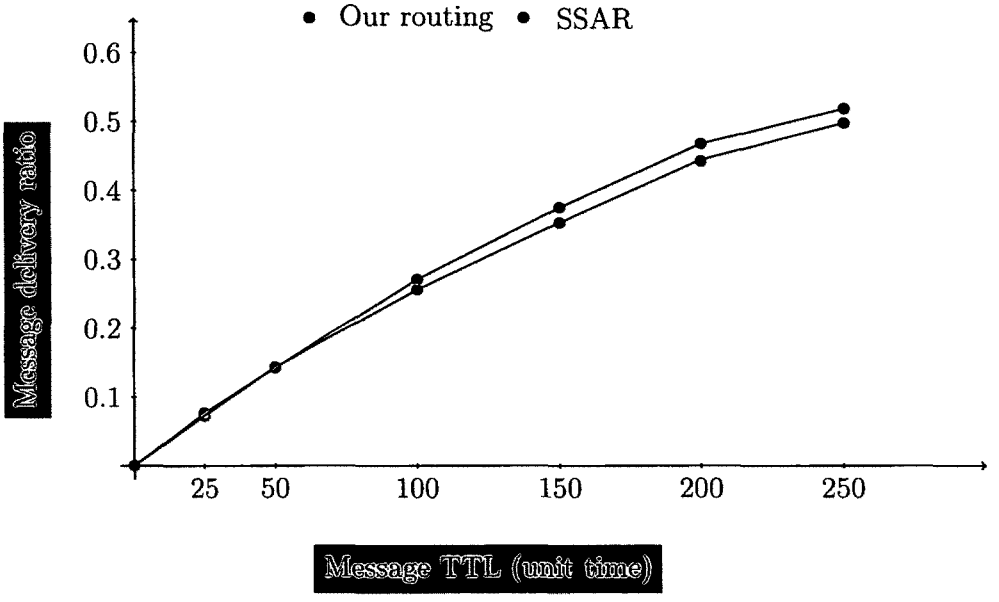

Figure 7.2: Performance Test: Message Delivery Ratio in environment 2 with different message's TTL.

It is obvious in Figures 7.1 and 7.2 that both routings can deliver more messages to the destinations when the messages' TTL increases. However, the delivery ratio of two routings could not increase sharply after the TTL reaches a certain value (about 20 time units in environment 1 and 150 time units in environment 2). The reason is that the forwarding capacity of the simulation environment becomes the performance bottleneck after the TTL reaches a certain value.

Compared to the result of SSAR, our routing has higher messages delivery ratio in both environments. It outperforms SSAR about 20% in environment 1 and 6% in environment 2. This is because our routing takes several factors into consideration such as forwarding object and next hop's willingness to the destinations. These considerations avoid undesirable phenomena mentioned in Chapter 5. Although a node has lower opportunities to encounter expected nodes in environment 2, the bidirectional model in our routing always prepares its candidate forwarding list according to the next hop's reliability on each message. This strategy avoids unreasonable forwarding so that the performance of our routing still dominates that of SSAR.

7.2.2 Cost and Efficient System

Table 7.4: The cost ratio of our routing to SSAR in environment 1, 95% confidence.

TTL	5	10	15	20
Our routing / SSAR	1.217 ± 0.225	1.665 ± 0.656	1.68 ± 0.241	1.44 ± 0.208

Table 7.4 and Table 7.5 list the average number of forwards done per successfully delivered message. It is the ratio of our routing to SSAR corresponding to environment 1 or environment 2 with different messages' TTL. It is obvious that two routings have more transmissions with increasing TTL, because each message has longer life cycle in the network and has more opportunities to be transmitted. In most cases, our routing has higher

performance than SSAR (Figure 7.1 & Figure 7.2), although the cost of our routing is still slightly higher than that of SSAR.

Table 7.5: The cost ratio of our routing to SSAR in environment 2, 95% confidence.

TTL	25	50	100
Our routing / SSAR	1.297 ± 0.269	1.184 ± 0.121	1.163 ± 0.11
TTL	150	200	250
Our routing / SSAR	0.935 ± 0.04	1.16 ± 0.06	1.12 ± 0.114

However, we cannot ignore another very important factor – the efficiency of forwarding. Deliveries of any message from its source to destination are called successful deliveries. The efficiency of forwarding could be defined as the ratio of forwarding in successful deliveries to total forwarding in the network. Any other forwarding is useless and wastes the resource of network. It is clear that the higher ratio of the efficiency of forwarding in the system, the more efficient the network. Table 7.6 & Figure 7.3 and Table 7.7 & Figure 7.4 illustrate our routing has higher ratio of the efficiency of forwarding than SSAR does in both simulation environments.

Table 7.6: The ratio of forwarding in successful deliveries to total forwarding in the network (environment 1), 95% confidence.

TTL	SSAR	Our Routing
5	$9.1\% \pm 1.6\%$	$9.95\% \pm 0.98\%$
10	$14.13\% \pm 1.9\%$	$14.97\% \pm 0.84\%$
15	$16.28\% \pm 2.5\%$	$19.64\% \pm 1.97\%$
20	$23.1\% \pm 1.52\%$	$30.21\% \pm 2.3\%$

With the increase of message delivery ratio, the efficiency of forwarding ratio also rises

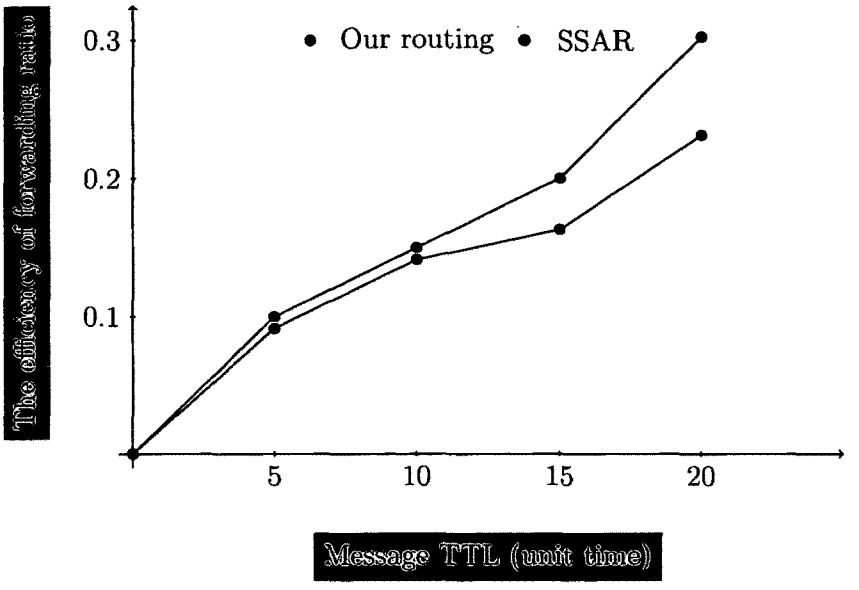

Figure 7.3: The ratio of forwarding in successful deliveries to total forwarding in the network (environment 1).

Table 7.7: The ratio of forwarding in successful deliveries to total forwarding in the network (environment 2), 95% confidence.

TTL	SSAR	Our Routing
25	$3.7\% \pm 0.3\%$	$3.4\% \pm 0.4\%$
50	$7.45\% \pm 0.2\%$	$8.6\% \pm 0.38\%$
100	$13.66\% \pm 1.3\%$	$14.63\% \pm 1.4\%$
150	$19.86\% \pm 1.3\%$	$20.34\% \pm 1.4\%$
200	$23.63\% \pm 1.6\%$	$26.145\% \pm 1.3\%$
250	$29.48\% \pm 2.1\%$	$32.69\% \pm 1.86\%$

steadily. Our routing always outperforms SSAR, because the bidirectional model in our routing considers the next hop's willingness to destination. Moreover, new priority formula

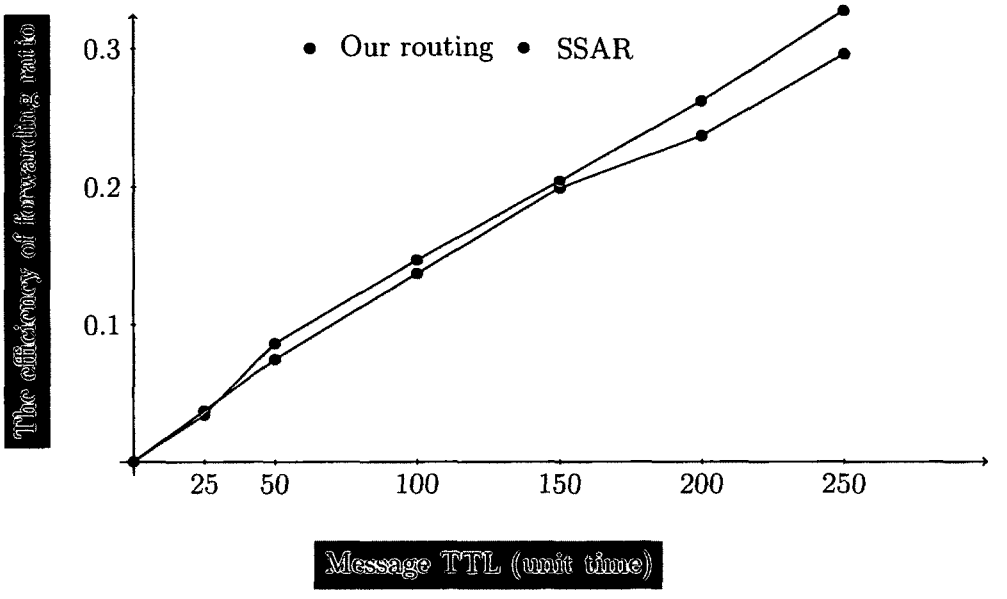

Figure 7.4: The ratio of forwarding in successful deliveries to total forwarding in the network (environment 2).

makes message's priority decrease not too sharply. This avoids zombie messages existing in system, which will miss forwarding opportunities to desirable intermediates after several forwarding because of sharp decrease of their priorities in SSAR.

7.2.3 Stress Test

To evaluate the performance of our routing under higher workload, we change the message generation rate from 1 message per node per unit time to 3 messages per node per unit time in environment 1 with messages' TTL of 20 time units. We also modify the message generation rate from 1 message per node per unit time to 10 messages per node per unit time in environment 2 with messages' TTL of 150 time units. The test results are displayed in Table 7.8 & Figure 7.5 and Table 7.9 & Figure 7.6.

When we increase workload, both routings have lower packet delivery ratio and more

Table 7.8: Stress Test: Message Delivery Ratio in environment 1 with different Message Generation Rate, 95% confidence.

Message Generation Rate	SSAR	Our Routing
1	$39.2\% \pm 2.68\%$	$47.12\% \pm 1.2\%$
2	$35.02\% \pm 0.69\%$	$39.94\% \pm 0.7\%$
3	$32.36\% \pm 0.66\%$	$37.15\% \pm 0.44\%$

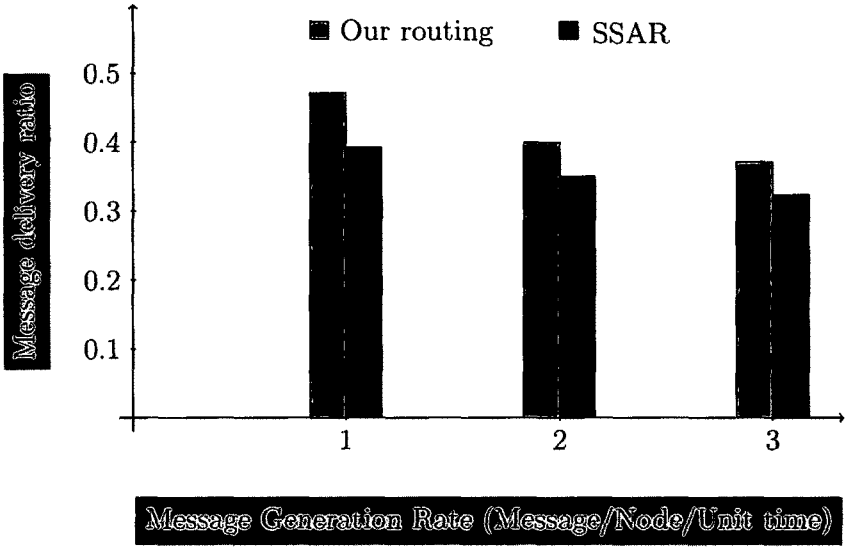

Figure 7.5: Stress Test: Message Delivery Ratio in environment 1 with different Message Generation Rate.

transmissions. However, they change at different rates in different environments. When the packet generation rate is 1 packet per node per unit time in environment 1, our routing delivers 22% more messages than SSAR does, because it makes more efficient usage of system resource. When the rate increases to 3 messages per node per unit time, our routing still delivers 15% more messages than SSAR. Although the dominance of our routing in environment 2 is not as remarkable as that in environment 1, its performance is still higher

that SSAR's about 7% at least. This means that our routing is more efficient under high workloads.

Table 7.9: Stress Test: Message Delivery Ratio in environment 2 with different Message Generation Rate, 95% confidence.

Message Generation Rate	SSAR	Our Routing
1	$35.26\% \pm 0.22\%$	$37.46\% \pm 0.24\%$
5	$24.93\% \pm 0.39\%$	$26.52\% \pm 0.09\%$
10	$22.23\% \pm 0.47\%$	$23.79\% \pm 0.08\%$

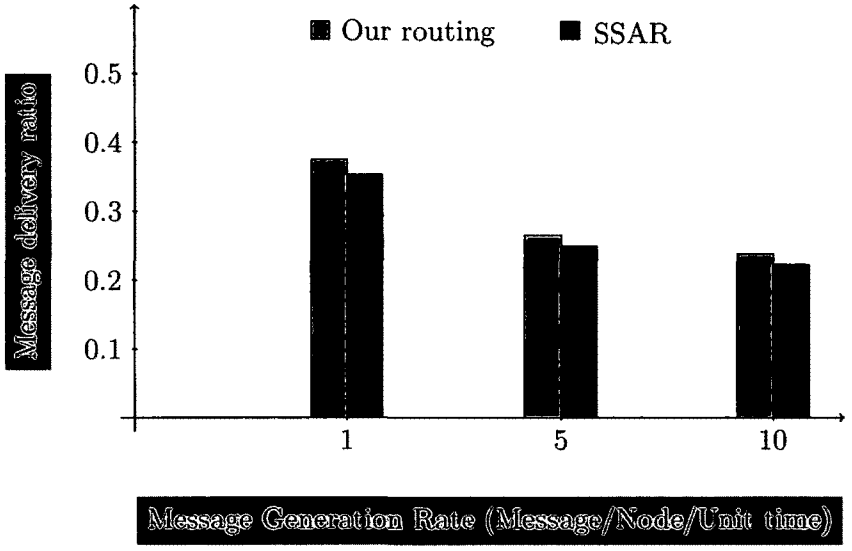

Figure 7.6: Stress Test: Message Delivery Ratio in environment 2 with different Message Generation Rate.

7.2.4 Dropping Test

We do dropping test only in environment 1 with the consideration of encounter number as a message dropping factor affecting the whole system and different routings. The messages' TTL is set as 20 time units and message generation rate is 1 message per node per unit time. The threshold of counter value is changed from 1 to 3. Table 7.10 & Figure 7.7 display the test result.

Table 7.10: The number of dropped messages with different threshold in envrionment 1, 95% confidence.

Threshold	SSAR	Our Routing
1	34.7 ± 4.87	72 ± 3.303
2	5 ± 0.84	36.4 ± 2.023
3	0.6 ± 0.27	21.7 ± 1.001

When the threshold is set as 1, the number of dropping messages in our routing is about two times as many as in SSAR. The message delivery ratio of two routings is almost same in this case, which means the performance of our routing decreases seriously than that of SSAR. With the increase of the threshold, the number of dropping messages reduces in both routings. The performance of two routings almost recovers at the original level if we set the threshold as 4 or more. However, the number of dropping messages decreases so fast in SSAR that the ratio of our routing to SSAR raises sharply. The value could not be displayed in the graph if the threshold is larger than three (Figure 7.7). Obviously, the threshold of encounter number has much more effect on our routing. This is because that our priority formula reduces message priority not as sharp as SSAR does. It means that such messages have more opportunities to be delivered and then seem more likely dropped if the threshold is too small. But in SSAR, a message has a low probability of dropping if its priority decreases too fast. This is the reason that we do not test in environment 2. The

effect could be totally ignored when the threshold is larger than two because of sharply decreasing priority, more messages number per node and limited resource of transmissions.

Fortunately, we could set this supporting factor with a reasonable value in order to reduce the side effect on delivery rate and waste on system resources in the real world environment.

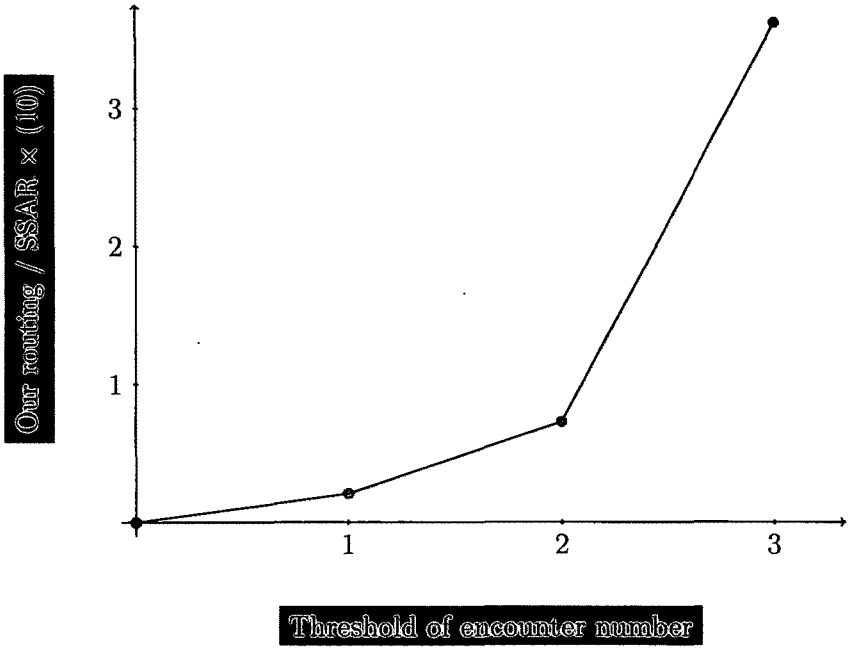

Figure 7.7: The encounter number as a message dropping factor affects the different routings.

7.3 Summary

In this chapter, we evaluated the performance of our routing in two environments. In environment 1, there are 10 nodes with initial 500 messages. In environment 2, there are 100 nodes with initial 50000 messages. Overall, our routing performed well and achieved higher performance and more efficient usage of system resource, although the cost of each

successful delivery was slightly higher. Our routing also worked well with high workload. The threshold of encounter number has more effects on our routing.

CHAPTER 8

CONCLUSION AND FUTURE WORK

We introduced the basic concepts and structure of DTNs and discussed the difference between DTNs and traditional networks in detail. Data delivery in DTNs is based on an abstraction of message switching capabilities with limited expectations of end-to-end path and node resources. We reviewed the two models acting as DTNs, including PeopleNet and Pocket Switched Networks in Chapter 3 and then we classified routing schemes implemented for DTNs into some categories based on their characteristics. We described the importance of social tie and social selfishness in DTNs and compared the performance of two routings in Chapter 4. One routing (SSAR) took social ties and social selfishness into consideration and the other did not.

We studied the routing SSAR and proposed three problems in Section 5.1 and then stated our improved model with corresponding assumptions in Section 5.2. We solved these three problems in the statement of our routing architecture steadily in Section 5.4. The simulation implementation was described in Chapter 6 and two simulation environments were proposed for the deployment of two routings. The performance of these routings was evaluated in Chapter 7. Section 8.1 summarized the work presented and highlighted our

contributions. We also provided recommendations for future work in Section 8.2.

8.1 Contributions

We studied the importance of social ties and social selfishness in DTNs and some routings implemented for DTNs. Then we stated three key problems which were not considered in other routings. We constructed our model with reasonable assumptions and approached these proposed problems. Our strategy found a balance point between the achievement of maximum system performance and social selfishness when a node selected its forwarding object. Our bidirectional model took the next intermediate's willingness to the destination into consideration in order to avoid undesirable forwarding and increase the message delivery ratio. We rebuilt the formula for message's priority calculation for the purpose of intelligent maintenance of messages' priority. Our new routing prevented the scenarios mentioned in Chapter 5.

We built two simulation environments to evaluate our routing performance compared to SSAR. In environment 1, there were 10 nodes with 500 initial messages. In environment 2, there were 100 nodes with 50000 initial messages. We assessed the performance according to different message' TTL as well as the workload. We not only calculated the cost of each successful delivery but also the efficient usage of network resources. We also studied the effect of threshold of encounter number on different routings.

Overall, our routing performed well and achieved higher performance and more efficient usage of system resources, although the cost of each successful delivery was slightly higher. Our routing also worked well with higher workload. The threshold of counter value had more effects on our routing than on SSAR.

8.2 Future Work

In our system, all messages are delivered silently. This means that source nodes never know whether the messages are received by their destinations, corrupted during the forwarding or dropped by intermediates. Usually, if there is no feedback to the source node, the source node might not suspect any problem during the forwarding. For example, sometime node A encounters node B, A receives some messages from B and then they separate. After that node A checks these messages, finds one of them is corrupted and then hopes to send a message to notify node B. However, it results in a critical issue of how node A makes sure node B could get this feedback and then receive a corrected message in the future. We need a special fault tolerant mechanism. For example, node A could set up a critical message with a special mark, which could not be dropped during its travel. The purpose is to make sure such a message is able to be received by its destination. To avoid abusing this power, each node in DTNs has only one opportunity to construct such a message of a certain time duration.

Another potential improvement to our routing is about adjustable willingness. The basic idea is that if node M helps node N to forward its messages, then willingness of N to M could be increased with some small value. With the increase of willingness, node N will be much happier to assist in delivering messages for M in next contact. It is possible to observe this phenomenon in real social networks. The situation that the value of willingness could be reduced should be also considered when M drops N's packets for any reason or M refuses to help.

8.3 Summary

In this chapter, we discussed our contributions and highlighted possible extensions to our future work. Simulation environments were presented and our routing exhibited good performance with different messages' TTL and high workload. Although the cost of every

successful delivery in our routing was higher than that in SSAR, our routing made highly efficient usage of system recourses.

BIBLIOGRAPHY

[1] Forrest Warthman. Delay-tolerant networks (dtns): A tutorial. *DTN Research Group Internet Draft*, 2003.

[2] Hemal Shah and Yogeshwar Kosta. Exploring and exploiting opportunistic network routing in a dtn environment. *International Journal of Future Generation Communication and Networking*, 4(2):13–22, 2011.

[3] Kevin Fall. A delay-tolerant network architecture for challenged internets. *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, 2003.

[4] Evan P.C. Jones and Paul A.S. Ward. Routing strategies for delay tolerant networks. *[Online]: http://ccng.uwaterloo.ca/~pasward/publications.shtml*, 2005.

[5] Eyuphan Bulut. Zijian Wang and Boleslaw K. Szymanski. Cost effective multi-period spraying for routing in delay tolerant networks. *IEEE/ACM Transactions on Networking*, 18(5):15301543, 2010.

[6] Eyuphan Bulut. Zijian Wang and Boleslaw K. Szymanski. Cost efficient erasure coding based routing in delay tolerant networks. *IEEE International Conference on Communications, ITACS*, pages 1–5, 2010.

[7] Hyewon Jun. Mostafa H. Ammar and Ellen W. Zegura. Power management in delay tolerant networks: a framework and knowledge-based mechanisms. *Sensor and Ad Hoc Communications and Networks*, pages 418–429, 2005.

[8] Qinghua Li. Sencun Zhu and Guohong Cao. Routing in socially selfish delay tolerant networks. *In Proceedings of INFOCOM IEEE*, pages 1–9, 2010.

[9] Ing-Ray Chen. Fenye Bao. Moonjeong Chang and Jin-Hee Cho. Trust management for encounter-based routing in delay tolerant networks. *IEEE Global Telecommunications Conference*, pages 1–6, 2010.

[10] W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols.* Addison-Wesley, 1993.

[11] Mooi Choo Chuah. Liang Cheng and Brian D. Davison. Enhanced disruption and fault tolerant network architecture for bundle delivery (edify). *Global Telecommunications Conference, IEEE*, 2(6):–812, 2005.

[12] Wei-Jen Hsu and Ahmed Helmy. On modeling user associations in wireless lan traces on university campuses. *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2006 4th International Symposium on*, 2:1–9, 2006.

[13] John Burgess. Brian Gallagher. David Jensen and Brian Neil Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications*, pages 1–11, 2006.

[14] Evan P.C. Jones. Lily Li and Paul A.S. Ward. Practical routing in delay-tolerant networks. *The IEEE Transactions on Mobile Computing*, 6(8):943–959, 2007.

[15] Vinod Venkataraman. Hrishikesh Bhatt Acharya and Harsh Shah. Delay tolerant networking - a tutorial. *[Online]: www.cs.utexas.edu/~vinodv/files/dtn-tutorial.pdf*, 2009.

[16] Mark S. Granovetter. The strength of week ties. *The American Journal of Sociology*, 78(6):1360–1380, 1973.

[17] Catherine Schaefer and James C. Coyne. The health-related functions of social support. *Journal of Behavioral Medicine*, 4(4):381–406, 1981.

[18] Brian Uzzi. Embeddedness in the making of financial capital: how social relations and networks benefit firms seeking financing. *American Sociological Review*, 64:481–505, 1999.

[19] David Krachardt. *Networks and Organizations: Structure, Form and Action.* Harvard Business School, Boston, MA, 1993.

[20] Eric Gilbert and Karrie Karahalios. Predicting tie strength with social media. *Proceedings of 27th international conference on Human factors in computing system*, pages 211–220, 2009.

[21] Peter V. Marsden. Core discussion networks of americans. *American Sociological Review*, 52(1):122–131, 1981.

[22] Steve Whittaker. Loren Terveen. Will Hill and Lynn Cherny. The dynamics of mass interaction. *CSCW '98 Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 257–264, 1998.

[23] Sushant Jain. Kevin Fall and Rabin Patra. Routing in a delay tolerant network. *In Proceedings of ACM SIGCOMM*, 34:145–158, 2004.

[24] Mehul Motani. Vikram Srinivasan and Pavan S. Nuggehalli. Peoplenet: Engineering a wireless virtual social network. *In Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 243–275, 2005.

[25] Pan Hui. Augustin Chaintreau. James Scott. Richard Gass. Jon Crowcroft and Christophe Diot. Pocket switched networks and human mobility in conference environments. *In Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 244–251, 2005.

[26] Amit Dvir and Athanasios V. Vasilakos. Backpressure-based routing protocol for dtns. *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, pages 405–406, 2010.

[27] Jing Su. Alvin Chin. Anna Popivanova. Ashvin Goely and Eyal de Lara. User mobility for opportunistic ad-hoc networking. *Proceedings of the sixth IEEE Workshop on Mobile Computing Systems and Applications.*, pages 41–50, 2004.

[28] Amin Vahdat and David Becker. Epidemic routing for partially-connected ad hoc networks. *Technical Report CS-200006, Duke University*, 2000.

[29] James A. Davis. Andrew H. Fagg and Brian N. Levine. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. *Wearable Computers, 2001. Proceedings. Fifth International Symposium on*, pages 141–148, 2001.

[30] Khaled A. Harras. Kevin C. Almeroth and Elizabeth M. Belding-Royer. Delay tolerant mobile networks (dtmns): Controlled flooding in sparse mobile networks. *In IFIP Networking*, 2005.

[31] Thrasyvoulos Spyropoulos. Konstantinos Psounis. and Cauligi S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259, 2005.

[32] Jason LeBrun. Chen-Nee Chuah. Dipak Ghosal and Michael Zhang. Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks. *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, pages 2289–2293, 2005.

[33] Jèrèmie Leguay. Timur Friedman and Vania Conan. Dtn routing in a mobility pattern space. *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, 1:276–283, 2005.

[34] Jèrèmie Leguay. Timur Friedman and Vania Conan. Evaluating mobility pattern space routing for dtns. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications*, pages 1–10, 2006.

[35] Shahram Ghandeharizadeh. Shyam Kapadia and Bhaskar Krishnamachari. An evaluation of availability latency in carrierbased vehicular adhoc networks. *MobiDE 06: Proceedings of the 5th ACM international workshop on Data engineering for wireless and mobile access*, pages 75–82, 2006.

[36] Pan Hui. Jon Crowcroft and Eiko Yoneki. Bubble rap: Social-based forwarding in delay tolerant networks. *In MobiHoc 08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 241–250, 2008.

[37] Samuel C. Nelson. Mehedi Bakht Robin Kravets. Encounter–based routing in DTNs. *INFOCOM 2009, IEEE*, pages 846–854, 2009.

[38] Elizabeth Daly and Mads Haahr. Social network analysis for routing in disconnected delay-tolerant manets. *In Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 32–48, 2007.

[39] MILGRAM Stanley. The small world problem. *Psychology Today*, 1:60–67, 1967.

[40] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[41] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, pages 215–239, 1979.

[42] Peter V. Marsden. Egocentric and sociocentric measures of network centrality. *Social networks*, 24:407–422, 2002.

[43] Martin Everetta and Stephen P. Borgatti. Ego network betweenness. *Social networks*, 27:31–38, 2005.

[44] Upendra Shevade. Han Hee. Song Lili and Qiu Yin Zhang. Incentive-aware routing in dtns. *IEEE International Conference on Network Protocols*, 27:238–247, 2008.

[45] M. Dawande. J. Kalagnanam. P. Keskinocak and R. Ravi. F. S. Salman. Approximation algorithms for the multiple knapsack problem with assignment restrictions. *Journal of Combinatorial Optimization*, 4:171–186, 2000.

[46] Alan Mislove. Massimiliano Marcon. Krishna P. Gummadi. Peter Druschel and Bobby Bhattacharjee. Measurement and analysis of online social networks. *In proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42, 2007.

[47] Aruna Balasubramanian. Brian Neil Levine and Arun Venkataramani. Dtn routing as a resource allocation problem. *In Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, 37(4):373–383, 2007.

[48] Eyuphan Bulut and Boleslaw K. Szymanski. Friendship based routing in delay tolerant mobile social networks. *IEEE Global Telecommunications Conference*, pages 1–5, 2010.

[49] Eyuphan Bulut and Boleslaw K. Szymanski. Semantic constraints for trust transitivity. *Proceeding APCCM '05 Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling*, 43:59–68, 2005.

[50] Michal Feldman and John Chuang. Overcoming free-riding behavior in peer-to-peer systems. *ACM SIGecom Exchanges.*, 5(4):41–50, 2005.

[51] Duncan J. Watts and Steven H. Strogatz. Overcoming free-riding behavior in peer-to-peer systems. *Nature*, 393(6684):440–442, 1998.

[52] Jon Kleinberg and Éva Tardos. *Algorithm design.* Pearson Education, 2006.

[53] Miguel Sánchez and Pietro Manzoni. Anejos: a java based simulator for ad hoc networks. *Future Generation Computer Systems*, 17(5):573–583, 2001.

[54] Tracy Camp. Jeff Boleng and Vanessa 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*, 2(5):483–502, 2002.

[55] Vanessa Ann Davies. Evaluating mobility models within an ad hoc network. *Masters thesis, Colorado School of Mines*, 2000.

[56] Eric W. Weisstein. *The CRC concise encyclopedia of mathematics, Second Edition.* CRC press, 2002.

[57] Ben Liang and Zygmunt J. Haas. Predictive distance-based mobility management for pcs networks. *In Proceedings of the ACM/IEEE international conference on communications societies (INFOCOM)*, 3:1377–1384, 1999.

[58] Xiaoyan Hong. Mario Gerla. Guangyu Pei and Ching-Chuan Chiang. A group mobility model for ad hoc wireless networks. *Proceeding MSWiM '99 Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 53–60, 1999.

[59] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless network. *In T. Imelinsky and H. Korth, editors, Mobile Computing. Kluwer Academic Publishers press*, pages 153–181, 1996.

[60] Jeff Boleng. Normalizing mobility characteristics and enabling adaptive protocols for ad hoc networks. *In Proceedings of the Local and Metropolitan Area Networks Workshop (LANMAN)*, pages 9–12, 2001.

[61] Brad Nelson Karp. Geographic routing for wireless networks. *PhD thesis, Harvard University*, 2000.

[62] Elizabeth M. Royer and Charles E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. *In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 207–218, 1999.

[63] Elizabeth M. Royer. P. Michael Melliar-Smitht. and Louise E. Mosert. An analysis of the optimum node density for ad hoc mobile networks. *In Proceedings of the IEEE International Conference on Communications (ICC)*, 3:857–861, 2001.

[64] Zygmunt J. Haas. A new routing protocol for reconfigurable wireless networks. *In Proceedings of the IEEE International Conference on Universal Personal Communications (ICUPC)*, pages 562–565, 1997.

[65] Brett McLaughlin. Gary Police and Dave West. *Head First: Object-Oriented Analysis and Design.* O'Reilly Media, Inc, 2006.

[66] Dan Pilone and Russ Miles. *Head First: Software Development.* O'Reilly Media, Inc, 2007.

[67] William Mendenhall. Robert J. Beaver and Barbara M. Beaver. *Introduction to Probability and Statistics.* Duxbury Press (12th edition), 2005.

[68] GraphPad Software. Graphpad prism. *[Online]: www. graphpad. com/prism*, 2009.