DEVELOPMENT OF AN IMPROVED SECURITY AIDED AND GROUP ENCOUNTER PROPHET ROUTING PROTOCOL OF AN OPPORTUNISTIC NETWORK

By

Husseina Jibrin<u>ABUBAKAR</u>

Department of Electrical and Computer Engineering
Faculty of Engineering
Ahmadu Bello University Zaria, Nigeria

MAY, 2017

DEVELOPMENT OF AN IMPROVED SECURITY AIDED AND GROUP ENCOUNTER PROPHET ROUTING PROTOCOL OF AN OPPORTUNISTIC NETWORK

By

Husseina Jibrin ABUBAKAR, B.Eng. (ABU, 2014)
P14EGEE8005

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES, AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF A MASTER OF SCIENCE (M.Sc) DEGREE IN TELECOMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
FACULTY OF ENGINEERING
AHMADU BELLO UNIVERSITY, ZARIA
NIGERIA

MAY, 2017

DECLARATION

I, Husseina ABUBAKAR Jibrin declare that this dissertation entitled "Development of an Improved Security Aided and Group Encounter PRoPHET Routing Protocol of an Opportunistic Network with Node Cooperation" was carried out by me in the Department of Electrical and Computer Engineering, Ahmadu Bello University, Zaria as part of the requirements for the award of degree of Master of Science in Computer Engineering. To the best of my knowledge, this research work has never been submitted anywhere for the award of any certificate. All literatures used and cited are duly acknowledged in the reference pages.

Husseina Jibrin ABUBAKAR _____ _____

(Signature) **Date**

CERTIFICATION

This Dissertation entitled "DEVELOPMENT OF AN IMPROVED SECURITY AIDED AND GROUP ENCOUNTER PRoPHET ROUTING PROTOCOL OF AN OPPORTUNISTIC NETWORK USING NODE COOPERATION" meets the requirements for the award of Master of Science (M.Sc.) degree and has been approved by the Department of Electrical and Computer Engineering, Ahmadu Bello University, Zaria for its contribution to knowledge and literary presentation.

Chairman, Supervisory Committee	_____	_____
(Dr. A.M.S. Tekanyi)	Signature	Date
Member, Supervisory Committee	_____	_____
(Dr. S.M. Sani)	Signature	Date
Head of Department	_____	_____
(Dr. Y. Jibril)	Signature	Date
Dean, School of Postgraduate Studies	_____	_____
(Prof. S.Z. Abubakar)	Signature	Date

DEDICATION

This research work is dedicated to Almighty Allah (S.W.T) the most merciful.

ACKNOWLEDGEMENT

In the name of Allah, most passionate, most merciful. All praises are due to Him for all His blessings in my life. Blessings and salutation be upon our noble Prophet Muhammad (S.A.W.).

My sincere gratitude goes to the chairman of my supervisory team, Dr A.S.M. Tekanyi, for his endless guidance, support and encouragement. You instilled in me, the hunger for learning new things every day. My endless appreciation also goes to my other supervisor, Dr S.M Sani for his patience, guidance and support, I am especially lucky to work with you both. Your positive influences on my life will never be forgotten, Allah bless and reward you both abundantly.

My sincere gratitude also goes to ETISALAT Nigeria for their educative support in the Etisalat Telecommunications Engineering Programme (ETEP). Having undergone the various training through this programme, it helped in better understanding of this work, I can boldly say it is an eye opener as well as a poiner to the path of success. Thanks to ETISALAT Nigeria, thanks to Ahmadu Bello University Zaria for this collaboration.

Special thanks to my parents, Malam Sheidu and Hajiya Maryam A.S. Abu for their endless prayers, love and encouragement. Allah bless you both abundantly.

My deepest appreciation to my hubby Mustafa Ojonuba Jibrin for his forbearance, kindness, understanding, love and care always, you are one in a million. I say thanks to my lovely daughter, Maryam Unekwu-Ojo Mustafa for her patience, understanding and love. To my siblings: Sa'adat Abu Hassan, Hajara Abubakar Abdullahi, Hassana UmmuMunir Abubakar, Muhammad Mujahed Ugbede Abubakar, Amina Ajuma Abubakar, my inlaws, especially, Hajara Afor Jibril, Abdullahi A.O Jibril, I love and appreciate you all.

To Ahmed Tijjani, for the support rendered towards the completion of this work. To Basira yahaya, you are a friend and a sister indeed. To my colleagues who made my stay in the university a memorable one, thank you very much. Allah bless us all abundantly.

ABSTRACT

One of the requirements for enabling two nodes to communicate through a network is the existence of a fully connected path between them. However, there are scenarios in wireless networks where this is not the case and yet nodes still need to communicate freely. Despite concerted efforts to resolve this problem of unconnected wireless nodes trying to relay crucial information, network users still experience significant communication challenges owing to failures or non-existence of critical infrastructural links between nodes and their security challenges. This research work is aimed at enhancing the security component of PRoPHET routing protocol by incorporating internodes cooperation. Simulation was carried out using the opportunistic network environment (ONE). This report presents the modeled opportunistic network using the security aided and group encounter forPRoPHET routing protocol. Node cooperation technique was developed and incorporated into the security aided and groups encounter PRoPHET routing protocol in order to improve its security. For the 20-node test scenario considered, the improved security aided and group encounter PRoPHET routing protocol outperformed the method proposed in the security aided and group encounter PRoPHET routing protocol of Basu et al., (2015) by 19.6%, 7.9%, 34.7% for delivery probability, hop count and buffer timeand for the benchmark Helsinki simulation area considered, it outperformed the method implemented in the work of Basu et al., (2015) by 25.7%, 62.9%, 55.5% with respect to delivery probability, hop count and buffer time respectively. Results showed that, node cooperation technique improved the security aided and groups encounter PRoPHET routing protocol because it increased the delivery probability, reduced the latency, reduced the hop count and increased the buffer time when tested on a 20-node test program and on the bench mark Helsinki simulation area at the end of the simulation time of 44000 seconds.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Acronyms	Definitions
DTN	Delay Tolerant Network
MANET	Mobile Ad-Hoc Network
SAGE	Security Aided and Group Encounter
ISAGE	Improved Security Aided and Group Encounter
OppNet	Opportunistic Network
GUI	Graphic User Interface
IDE	Integrated Development Environment
LAN	Local Area Network
ONE	Opportunistic Network Environment
RAM	Random Access Memory
TCP	Transmission Control Protocol
Wi-Fi	Wireless Fidelity
NC	Node Cooperation
CN	Camp Node
SN	Shelter Node
FN	Forwarder Node
DN	Destination Node
PRoPHET	Probability Routing using History of Encounter and Transitivity
MobySpace	Mobility Space
CAR	Context-Aware Routing

CHAPTER ONE

INTRODUCTION

1.1 Background

Mobile Ad-hoc Network (MANET) is defined as a collection of communication devices or nodes that communicate without any fixed infrastructure and pre-determined organization of available links (Dinakar *et al.*,, 2012). The Opportunistic Network (OppNet), also called any path routing, is characterized as a necessary evolution of traditional MANET with providing wireless network properties. OppNet consists of both fixed and human-carried mobile devices (nodes) that communicate with each other with or without any infrastructure (a central command station that monitors and controls the activities of the network)(Papaj *et al.*,, 2012). OppNets are formed by individual nodes. All nodes can be disconnected for some time intervals and each opportunistically exploits any contact with other nodes to forward its messages (Papaj *et al.*, 2012). Each node computes for the best paths based on its knowledge of the routes. The messages are routed and transmitted by "store-carry-forward" model as shown in Fig. 1.1.

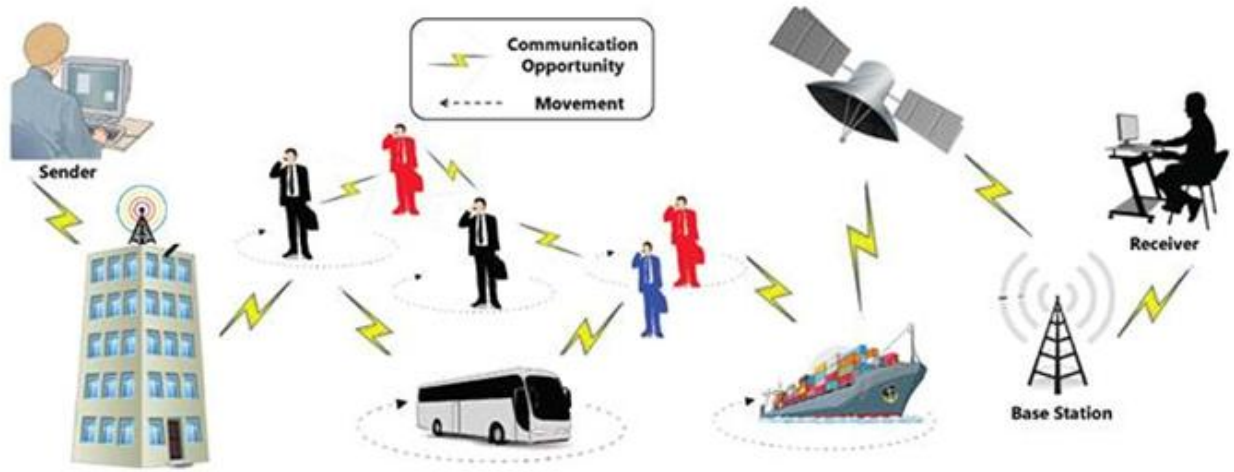

Fig.1.1: Example of opportunistic network(Papaj *et al.*, 2012).

The main function of an OppNet is to provide ability to exchange messages between source and destination nodes. Nodes can be of two types, mobile and fixed (Yogi & Chinthala, 2014). They are responsible for the control and management decisions on locally available information in

order to provide effective communication between nodes. The mobility in an OppNet is used to provide efficient communication between unconnected groups of nodes.

In MANET, the security mechanisms are based on the assumption that there is a connection between source and target nodes (end-to-end connections). In an OppNet, however, there is need for security solutions, which provide security for all the nodes, all services and applications that participate in routing and transmission process. It is a type of Delay Tolerant Network (DTN) that is composed of mobile and super nodes, and comprising of the following features:(Dumytro*et al.,* 2011).

1. Nodes are intermittently connected.
2. There is no permanent source-destination and end-to-end paths between nodes.
3. Disconnections and reconnections frequently occur between nodes.
4. Network connectivity is highly variable.

All network topologies are variable; moreover, node mobility enables an OppNet to be applied in crisis management areas (Dumytro*et al.,* 2011).

During data transmission, OppNets are connected through user devices as they move, thus completing message transmission. However, this transmission method is accompanied by the security problem of uncertainty during movements. For example, users may not be aware of whether randomly encountered nodes are secured and may be attacked when they encounter malicious nodes (Nicholas *et al.,* 2013). In addition, protecting user's personal privacy is another crucial concern. OppNet-related studies have mostly emphasized designing resource-efficient routing methods and have seldom focused on the protection of personal data and privacy. There is sporadic connectivity of nodes and there is need to provide secure delivery of the messages from source node to destination node (Guo *et al.*,, 2015).

OppNets are going to be the future generation technology with limitless applications and scopes because in times of war, where network becomes sparse or in remote areas of developing countries where there is limited access to the Internet, the OppNet routing protocols promise to be a better message delivery, since it is mainly characterized by store, carry and forward paradigm (Huang *et al.*,, 2008). With the recent growth in wireless devices, there is a huge opportunity of message delivery where every node can become a participant. Routing in an

OppNet is challenging because it is not known in advance as to when a node will get the opportunity to deliver message to its right next candidate node. Network topology is also unknown to every node in the network and it changes dynamically. Even if an appropriaterouting methodology is chosen, it is hard to know whether a candidate node behaves appropriately or maliciously in the system. Thus, node cooperation is required in a systematic manner. This requires some techniques that let the routing node know the exact behavior of other nodes. This helps in identifying the malicious behavior of nodes in the network. A malicious behavior leads to a considerable delay in the message delivery or no delivery at all to the intended destination in the network under consideration (Wu *et al.*,, 2015). In such networks, continuous end-to-end connectivity may be impossible. Because of unique features of high mobility of nodes, frequent link variation, and long communication delays, many opportunistic forwarding protocols present major security issues (Wu *et al.*, 2015). The design of OppNets faces serious problems such as how to effectively implement node authentication and access control, confidentiality and data integrity as well as ensuring routing security, privacy protection, cooperation, and trust management. In other words, systematic research on security solutions for forwarding protocols in OppNets is still open and challenging (Basu *et al.*,, 2015).

For example, any large-scale disasters like flood and cyclone have severe impact on communication infrastructure. Services like cell phone/internet connectivity immediately become non-functional in emergencies due to the failure of the supporting infrastructure through both system damage and system overuse (Luo *et al.*,, 2006). Therefore, the possibility of information exchange using normal communication infrastructure is almost ruled out. According to the World Disasters Report, (2013), when disaster strikes, access to information is as important as access to food and water (Vinck, 2013). As identified by project RESCUE, any crisis response activity consists of several interrelated phases each of which requires appropriate situational information for its execution (Mehrotra *et al.*,, 2004). This acute need for information exchange demands setting up of a temporary post-disaster communication network until the normal communication infrastructure is operational again.

Therefore, PRoPHET(Grasic *et al.*,, 2011) is one of the benchmark routing protocols for DTN. It fits well for such encounter-based forwarding as it uses the history of previous encounters with

other nodes, as well as the transitive properties of the network for bundle forwarding over the network (Grasic *et al.*, 2011). Nevertheless, to use PRoPHET for such group encounter based routing of situational messages; the protocol needs to be aligned with the group mobility patterns and history of group encounters.PRoPHETis one of the major forwarding protocols in an opportunistic network which relies on the implicit assumption that all nodes in the network are honest and are working towards the common goal of message forwarding (Kaur&Kaur, 2009; Orozco *et al.,*Verma & Srivastava, 2012). However, this assumption turns out incorrect in the presence of unauthorized and malicious nodes that can launch serious attacks like black-hole attack, identity spoofing, bundle store overflow and other forms of attacks on the network. However, some nodes in the opportunistic network may not be willing to participate in the routing process at all times (Ciobanu *et al.*,, 2015). Thus, a node may be selfish towards another node, for various reasons, for example, it might be low on resources such as battery life, memory or lack of interest in helping nodes outside its own group. The existence of such selfish nodes in an OppNet might leads to messages having high delays or never to be delivered at all, so these nodes must be acknowledged and avoided when possible. Therefore, incorporating security features into PRoPHET for protecting the network from possible attacks and guarding messages from possible eavesdropping are inevitably important in maintaining its operation.

1.2 Significance of Research

The significance of this research work is to ensure availability and to resist malicious dropping in OpptNets where a node may refuse to act as a relay and only settle for sending and receiving its own data or information, thus, causing considerable delay degradation in the network. The improvement which involved the development of a node cooperation routing capable to ensure improvement in network security helped in identifying the malicious behavior of nodes in the network which cause considerable delay in the message delivery or no delivery at all.

1.3 Problem Statement

In OpptNets, there is problem of availability and malicious dropping of messages. There is need to identify those nodes that maliciously drop and cause considerable delay degradation in the message timely delivery. This research work is aimed at improving the existing security aided

and group encounter prophet routing protocol by incorporating node cooperation in to it as a means of controlling malicious behavior of nodes.

1.4 Research Aim and Objectives

This research work is aimed at enhancing the security component of PRoPHET routing protocol by incorporating inter-nodes cooperation.

The objectives are:

1. To replicate the SAGE-PRoPHET of Basu et al (2015) and develop the improved SAGE-PRoPHET (ISAGE-PRoPHET) using inter-nodes cooperation.
2. To develop the improve SAGE-PRoPHET of an Opportunistic network using node cooperation technique
3. To simulation environment of the 20-nodes and benchmark Helsinki test in ONE simulator and implement the simulations using the routing protocols in 1.
4. To compare the results obtained in 2 using the metrics of delivery probability, hop count and buffer time management.

.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

The literature review provides the background knowledge relevant to the scope of this research in two perspectives: Review of fundamental concepts and that of similar works. These reviews reveal some of the techniques, algorithms, model equations, etc used by fellow researchers as well as the extent of their researches on security problems in a PRoPHET routing protocol under study.

2.2 Review of Fundamental Concepts

This covers the review of relevant fundamental concepts. It covers the concept of opportunistic network, routing in opportunistic network, security measures, security threats and requirements and also, the tools for evaluating a Delay Tolerant Network (DTN).

2.2.1 Opportunistic networks

In opportunistic networks, routes are computed at each hop while a packet is forwarded. So, each node receiving a message for an eventual destination exploits local knowledge to decide the best next hop among its current neighbors to reach the eventual packet destination. When no forwarding opportunity exists (for example, no other nodes are in the transmission range, or the neighbors evaluated are not suitable for that communication) the node stores the message and waits for future contact opportunities with other devices to forward the information. However, each single node acts as a gateway. This makes opportunistic networks a very flexible environment (Pelusi *et al.,* 2006a) and calls for a more radical revision of legacy routing approaches designed for the Internet or for well-connected MANETs.

As shown in Fig. 2.1, the lady at the desktop opportunistically transfers a message for a friend to a bus crossing the area via a Wi-Fi link and hope that the bus will carry the information closer to the destination.

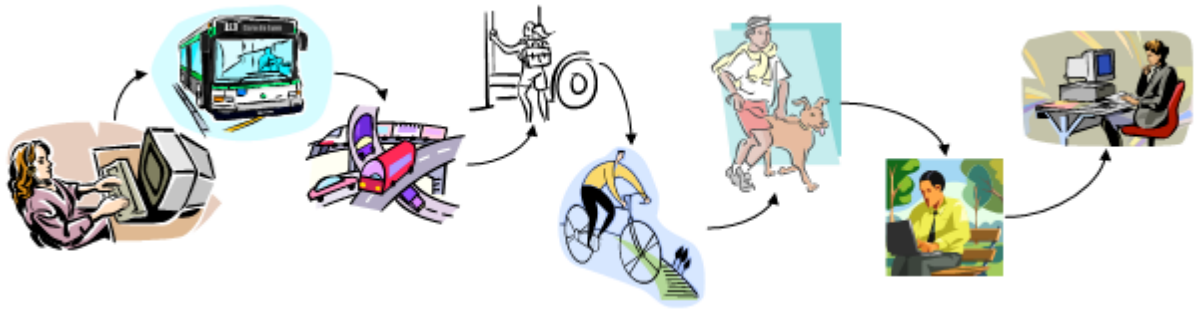

Fig. 2.1:Opportunistic Networking Example (Pelusi *et al.*, 2006a).

The bus moves through the traffic, then uses its Bluetooth radio to forward the message to the mobile phone of a lady that is getting off at one of the bus stops. The lady walks through a nearly park to reach the university. Her cellular phone sends the message to a cyclist passing by. By proceeding in the same way some hops further, the message eventually arrives at the receiver (Pelusi *et al.*, 2006a). As clearly shown in this example, a network connection between the two women never exists, but by opportunistically exploiting contacts among heterogeneous devices, the message is delivered through hop-by-hop transmission and eventually gets to the destination. And also, allowing nodes that are not connected at the same time to the same network to communicate with each other.

2.2.2 Routing in an Opportunistic Network

An opportunistic network is a Delay Tolerant Network (DTN), where data is routed from source to destination with delay tolerance (Fall *et al.*,, 2007). This type of network is used for emergency applications. An opportunistic network (OppNet) is a network of either mobile nodes or fixed nodes. It is different from traditional network, where nodes are deployed together and end to end paths exist for data forwarding. In an opportunistic network, on the other hand, there is no fixed path between source and destination due to mobile nodes. It is an extension of a mobile ad hoc network. In an opportunistic network (Yahaya, 2015) routing is the most compelling challenge, since the design of efficient routing strategies is generally a complicated task due to the absence of knowledge about the topological evolution of the network. Routing performance improves when more knowledge about the expected topology of the network is exploited (Pelusi *et al.*, 2006a). Unfortunately, this kind of knowledge is not easily available and a trade-off must be met between performance and knowledge requirements. Classification is

between algorithms designed for completely flat ad hoc networks (without infrastructure) and algorithms in which the ad hoc networks exploit some form of infrastructure to opportunistically forward messages as shown in Fig. 2.2. In the former case, approaches can be further divided in to dissemination-based (Epidermic) and context-based (PRoPHET) routing. Dissemination-based algorithms are essentially forms of controlled flooding, differentiated by the policy used to limit flooding. Context-based approaches usually do not adopt flooding schemes, but use the knowledge of the context that nodes are operating on to identify the best next hop at each forwarding step. Algorithms that exploit some form of infrastructure can be divided in to fixed and mobile infrastructures depending on the type of infrastructure they rely on. In both cases, the infrastructure is composed of special nodes that are more powerful with respect to the other nodes commonly present in the ad hoc network. They have high storage capacity and hence can collect messages from many nodes passing by, even for a long time they also have high energy. Nodes of a fixed infrastructure are located at specific geographical points, whereas nodes of a mobile infrastructure move around in the network following either pre-determined known paths or completely random paths.

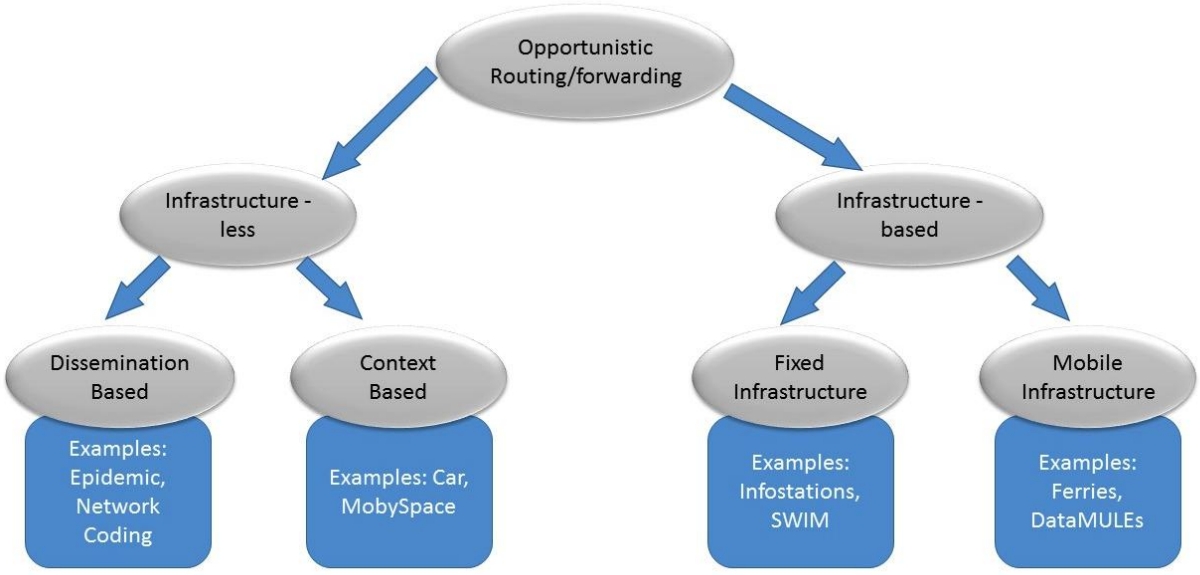

Fig. 2.2: Taxonomy of Routing Techniques for Opportunistic Networks(Pelusi *et al.*, 2006b)

2.2.3 Other Candidate Routing Protocols for Context-Based Opportunistic Network

The candidate routing techniques for context-based routing protocols for OppNets are:

2.2.3.1 Context-Aware Routing (CAR)

Nodes are divided in groups; Nodes inside the same group have social links between each other. A node selects the forwarder node/group

(i) Probabilistically

(ii) Based on the social attraction among nodes.

CAR computes delivery probabilities proactively, and disseminates them in their ad hoc cloud. Context information is exploited to evaluate probabilities just for known destinations; CAR can be seen as a conjunction between mobility-based protocols and social context-based protocols.

2.2.3.2 Mobility Space Routing (MobySpace Routing)

The mobility pattern of nodes is the context information used for routing. Decisions rely on the notionthat a node is a good candidate for taking custody of a bundle ifit has a mobility pattern similar to that of the bundle's destination.Routing is done by forwarding bundles toward nodes that have mobilitypatterns that are more and more similar to the mobility patternof the destination. In the MobySpace, the mobility pattern of a node provides its coordinates. Several questions arise. What type of dimensions do we choose, how many, and what kind of range for values do we define? How do we define the notion of distance? And so on.

2.2.3.3 Bubble-Rap

Thestructure of human mobility is used to design the forwarding algorithms for Delay Tolerant Networks for thedissemination of data amongst mobile users. Cooperation binds but also divides human society into communities. Members of the same community interact with each other preferentially.Automatically inferring the parameters of the underlyingsocial structure, they exploit the structure properties to select paths and dynamically identify member's communities. However, it is generally customized for a specific type of context information.

2.2.3.4PRoPHET +(Probability Routing using History of Encounter and Transitivity plus)

PRoPHET+ is a routing protocol for opportunistic networks that reduces chances of data loss and delivery delay while maintaining the high probability of delivery successfulness between a source and destination node by providing communication opportunities. Buffer, power, bandwidth, popularity, and predictability value from PRoPHET are used to create a weighted function. In theory, the weights could be shifted to adapt to various environments.The weights are determined based on a few trials and are chosen based on qualitative considerations. However, if logical choices of weights are not made, its performance will be very poor.

2.2.3.5 PRoPHET (Probability Routing using History of Encounter and Transitivity)

It is a Probabilistic routing duringwhich, contact nodes exchange their delivery predictability (DP) to destinations of messages they store in their buffers, messages are requested only if the DP is higher than that of the node currently storing the message. DP is the probability for a node to encounter a certain destination, it increases when the node meets the destination and decreases (according to an ageing function) between meetings. Context information used by PRoPHET is the frequency of meetings between nodes.

2.2.4PRoPHETRouting Protocol

PRoPHETis a data forwarding protocol designed for DTNs (Pelusi *et al.*, 2006b; Verma & Srivastava, 2012), which uses the encounter history and transitivity to predict the possibility of two nodes encountering in the future. Probabilistic metric called delivery predictability is established at each node for each known destination indicating the predicted chance of that node delivering a message to that destination. When a node encounters another node, they exchange information about the delivery predictabilities they have and update their own information accordingly. Based on the delivery predictabilities, a decision is then made on whether or not to forward a certain message to this node.PRoPHETdepends on the following three important equations formulated as equations, (2.1) through equation (2.3) to update the delivery probability values (Lindgren *et al.*,, 2003a). The detail formulation of each of these equations is explained as follows:

The protocol relies on the delivery predictability metric, $P \in [0, 1]$, that should reflect the probability of encountering a certain node. That metric should be used to support the decision of

whether or not to forward a message to a certain node. Whenever a node is encountered, the metric should be updated according to (2.1), where $P(A,B)$ is the delivery predictability node *A* has for node *B* and *Pinit* $\in$[0, 1] is an initialization constant according to (Lindgren *et al*., (2013). (Lindgren *et al.*,, 2003b) . This ensures that nodes that are often encountered have high delivery predictability and the relationship of these probabilities (Lindgren *et al.*, 2003b) is given as:

$$P(A,B) = P(A,B)_{old} + \left(1 - P(A,B)_{old} \times Pinit\right) \quad (2.1)$$

If a pair of nodes do not encounter each other for a while, they are less likely to be good forwarders of messages to each other, thus the delivery predictability values must age (Lindgren *et al.*, 2003b), and is reduced in the process. The aging equation is shown in (2.2), where $\gamma \in$[0, 1] is the aging constant, and *k* is the number of time units that have elapsed since the last time the metric was aged. The time unit used can differ and should be defined based on the application and the expected delay in the targeted network with the aging equation (Lindgren *et al.*, 2003b) as:

$$P(A,B) = P(A,B)_{old} \times \gamma^{k} \quad (2.2)$$

The delivery predictability also has a transitive property, that is based on the observation that if node A frequently encounters node B and node B frequently encounters node C, then node C probably is a good node to forward messages destined for node A. Equation (2.3) shows how this transitivity affects the delivery predictability, where $\beta \in$[0, 1] is a scaling constant that decides how large an impact the transitivity should have on the delivery predictability and the equation (Lindgren *et al.*, 2003b) is as:

$$P(A,C) = P(A,C)_{old} + (1 - P(A,C)_{old}) \times P(A,B) \times P(B,C) \times \beta \quad (2.3)$$

Apparently, in PRoPHET, a malicious node that arbitrarily claims delivery probability will be able to intercept data from other nodes and then it either drops or arbitrarily forwards them, which will detrimentally degrade the network performance.

2.2.5 Security Threats and Requirements

Opportunistic networks inherit many characteristics of DTNs and MANETs including vulnerabilities (Grasic *et al.*, 2011) but suffer from more security threats which are listed as follows:

(i) **Identity Spoofing:** A malicious node may misrepresent itself by claiming to be someone else. This could be used to steal messages that are meant for a particular node.

(ii) **Black Hole Attack:** Malicious node acting as a black hole sets its delivery predictabilities for all destinations to a value close to or equal to 1, requests all messages from nodes it meets and forwards none of them. A node encountering such a malicious node will try to forward all its bundles to the malicious node, creating the belief that the bundle has been very favorably forwarded. Due to the transitivity, the delivery predictabilities reported by the malicious node will affect the delivery predictabilities of other nodes.

(iii) **Bundle Store Overflow:** A malicious node may generate a large number of fake messages for a particular destination and fill up the buffer of a target node at the expense of other legitimate messages. Moreover, malicious nodes may redistribute older messages from known sources; this will probably worsen storage overflow if the message is replayed before it expires.

2.2.6 Disaster Response and Infrastructure

The infrastructures that are needed in the event of an emergency cannot be carefully planned and modeled beforehand. They emerge spontaneously and rapidly change over time.

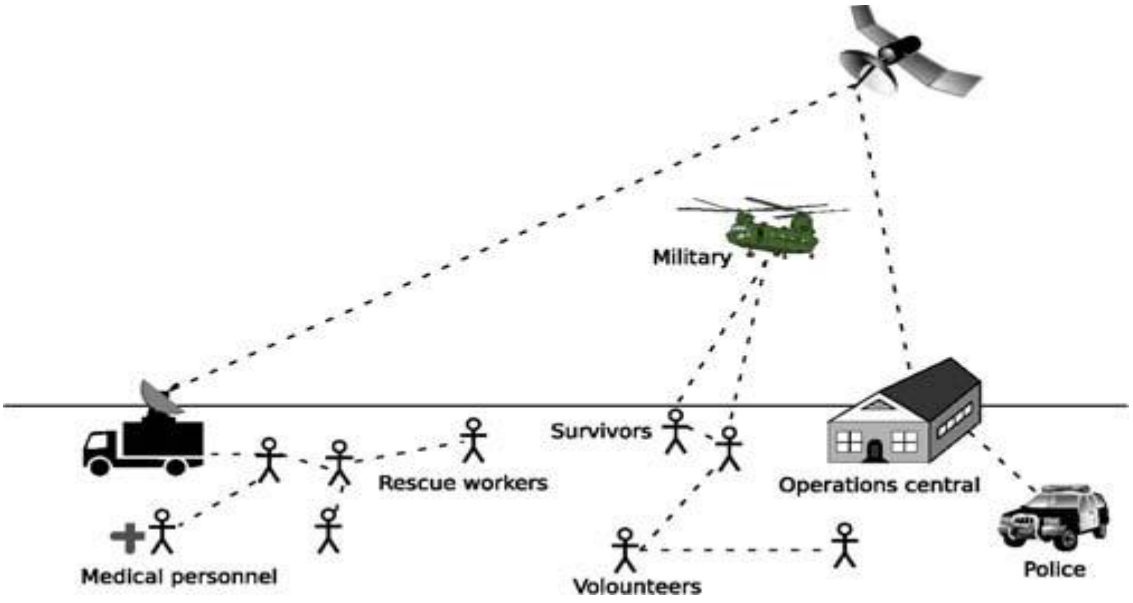

Fig. 2.3: Disaster Response Scenario(Asplund *et al.*, 2008).

Such systems are not intended to replace current systems for everyday use since they are in many ways suboptimal. Their strength is the fact that they can be deployed when the other communication networks have failed. Disaster response needs lie in the nature of unforeseen events and disasters that they are impossible to characterize in a uniform way. The needs and resources differ drastically depending on circumstances such as the scale of the event, which part of the world is affected, and the type of events (earthquake, flooding, fire, epidemic, etc). However, three important problems can be identified: (i) Lack of a common operational environment (ii) Non matching of needs and resources (iii) Security issues and poor utilization of the network to keep exchange of information going.

The military is often one of the key actors, in the event of a disaster or the initial group of people to establish and manage interim information infrastructures, to distribute information, and to coordinate the relief engagement in the event of a disaster. The armed forces have long experience of dealing with disaster. On the other hand, one of the biggest challenges for the military is to be able to participate in collaborative networked environments (Uddin *et al.*,, 2009), such as hastily formed networks for disaster mitigation, while safeguarding valuable information, and upholding confidentiality, integrity, and non-repudiation properties of the network and its information.

2.2.7 Post Disaster Relief Operation

After a large scale natural disaster, victims normally take shelter in nearby safe areas like school buildings, temporary tents in some highland areas and other risk free zones. Several disaster response agencies set up relief camps in and around the disaster affected area and mobilize manpower (volunteers) and resources to those camps in order to carry out relief operations. Each camp has a dedicated number of volunteers who provide a specific type of service relevant to the camp. For example, volunteers associated with Health Care Relief Camps will provide medical aid, whereas those associated with Logistic Relief Camps will provide relief materials like foodstuff, clothes, blankets, tents, etc. to the victims in each shelter. A command and control station is established away from the disaster affected area to organize resource distribution and coordinated relief work in the different relief camps. This leads to a coordinated and collaborated effort towards disaster rescue, response, relief, and rehabilitation. The hierarchical structure is illustrated in Fig. 2.4:

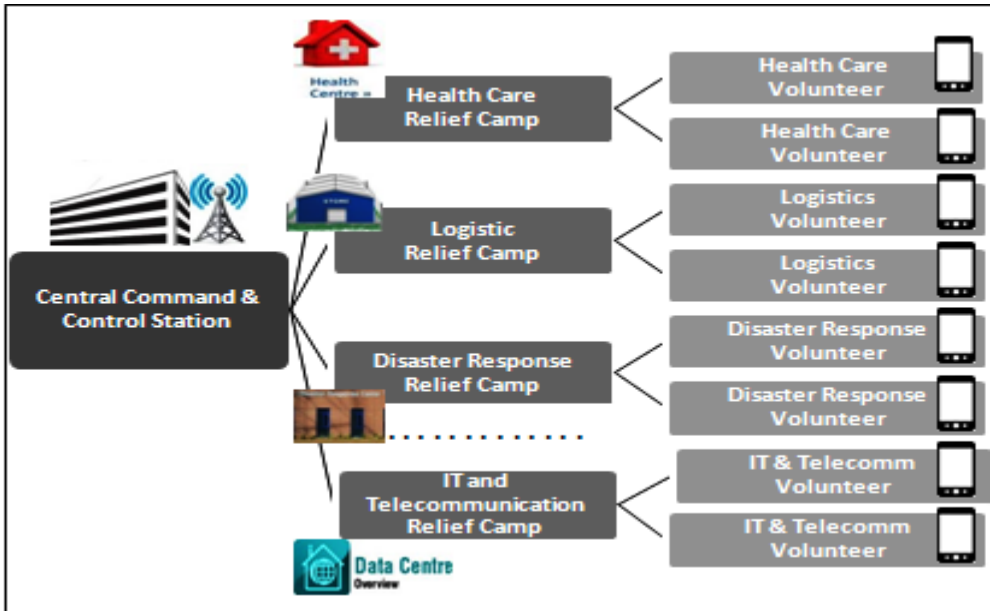

Fig. 2.4: Hierarchical Structure of Disaster Relief Operations (Basu *et al.*, 2015)

2.2.8PRoPHETfor Group Encounter Routing

Situational messages are categorized according to their content and hence pertain to any one of the groups working in the disaster area. Such categorized messages need to be forwarded to their respective group relief camps for necessary action. In a typical disaster relief environment, volunteers belonging to a particular group periodically visit their corresponding relief camp for

collecting resources, etc. Therefore, they can be considered as the most suitable forwarders of messages destined to that particular camp (Basu *et al.*, 2015). Now, due to group mobility pattern or interdependence among the groups, a volunteer belonging to a particular group encounters volunteers of its own group or some specific groups more frequently than volunteers of other groups. Therefore, it is sensible to judiciously exploit this encounter pattern for forwarding categorized situational messages to their respective destinations. PRoPHETfits well for such encounter history based forwarding as it uses the history of previous encounters with other nodes as well as the transitive properties of the network for bundle forwarding over the network (Pelusi *et al.*, 2006b; Verma & Srivastava, 2012). Some of the basic assumptions regarding distribution of pins, etc, are:

2.2.9 Pin Distributions at the Setup Phase

At the setup phase, each rescue group is assigned a group-id, known GID_j, where $j=1,..., n$ by the central command and control station. In addition, central command and control station creates a pair of pins called Group-Pin (GP_j) and Modified Group-Pin (MGP_j) for volunteers of each group (GID_j). GP_j is kept secret within the members of group j and MGP_j is distributed among registered members of all other groups. A group-pin is actually a common private key generated uniquely for each specific group and the modified group-pin is a corresponding public key common for all members of a particular group. Such group keys reduce the burden of maintaining private key-public key pairs for each individual volunteer, which is essential due to the limited resources (storage, battery power, etc.) of the smart phones.

Each shelter-node is assigned a shelter-id, say SID_k, where $k = 1,..., K$ and a shelter-pin, modified shelter-pin pair, ($SPin_k$, $MSPin_k$) respectively. $MSPin_k$ is distributed among all forwarder-nodes and camp-nodes. Lastly, each camp-node is assigned a camp-id (CID_j) and a camp-pin, modified camp-pin pair, ($CPin_j$, $MCPin_j$) respectively. CID_j and $MCPin_j$ are shared with all shelter-nodes. If a new volunteer wants to join the relief operation he/she has to register with any one of these groups. On registration, the volunteer is provided with the necessary details like *GID* of all groups, *GP* of his own group, *MGP* of other groups and *MSPin*of all

shelter-nodes which will be loaded in his smart-phone. Thus, new entrants (volunteers) can always be added into the network. The central command and control station, which is assumed to be a trusted authority, distributes keys to different groups, shelters and relief camps at the set-up phase. Once this is done, the central control station has no role in the runtime phase when forwarder-nodes check authenticity of peers and forward situational messages using the secured PRoPHETrouting protocol in a fully decentralized fashion.

2.2.10 Modifying PRoPHETfor Group Encounter Routing

For group encounter based PRoPHET, a group Delivery Predictability (DP) table is maintained at each node, storing the delivery predictabilities of all nodes in the network with all rescue groups working in the area. The $P(FN_i, GID_j)$ *is* defined as the group delivery predictability of a forwarder-node FN_i, belonging to group *i* with any member of group *j*. Whenever FN_i belonging to group *i* and FN_j, belonging to group *j* meet with each other, they exchange group-DP table and update their own group-DP table based on the identity of the group and the received table. For an encountered node FN_j, node FN_i updates its group-DP using equation (2.4)(Basu *et al.*, 2015) as:

$$P(FN_{i,GID_j})_{new} = P(FN_{i,}GID_j)_{old} + (1 - P(FN_i, GID_j)_{old}) \times Penc \quad (2.4)$$

Similarly, for an encountered node FN_i , node FN_j updates its group-DP using equation (2.5)(Basu *et al.*, 2015) as:

$$P(FN_j, GID_j)_{new} = P(FN_j, GID_i)_{old} + (1 - P(FN_j, GID_i)_{old}) \times penc \quad (2.5)$$

Note that, $P(FN_i, GID_j)$ is different from $P(FN_j, GID_i)$ because FN_i may not have met as many number of volunteers from group *j*, as FN_j has met volunteers of group *i*. FN_i updates transitive group-DPs for all other groups *k* encountered by FN_j using equation (2.6)(Basu *et al.*, 2015) as:

$$P(FN_{i,}GID_k)_{new} = \max(P(FN_iGID_k)_{old,}P(FN_{j,}GID_K) \times P(FN_i, GID_j)_{new\times}\beta) \quad (2.6)$$

FN_j updates transitive group-DPs for all other groups *k* encountered by FN_i using equation (2.7)(Basu *et al.*, 2015) as:

$$P(FN_{j,}GID_k)_{new} = \max(P(FN_{j,}GID_k)_{old}, P(FN_{i,}GID_k) \times P(FN_j, GID_i)_{new} \times \beta) \quad (2.7)$$

DP table is periodically aged for all groups *k* using equation (2.8*)* (Basu *et al.*, 2015) as:

$$P(FN_i, GID_k)_{new} = P(FN_i, GID_k)_{old} \times \gamma^{T} \quad (2.8)$$

For this purpose, using equations (2.4) to (2.8), prophet can route categorized messages destined to a particular relief camp through volunteers of that group or who encounter members of that group very often to accelerate the delivery of such messages. A forwarder-node only needs to store delivery predictabilities with different groups instead of delivery predictabilities with each individual node in the network. This alleviates the overhead of calculating and storing delivery predictabilities for each and every node in the network.

2.2.11 Group Encounter Based and Security

In other to avoid possible attacks by malicious network nodes in PRoPHET, the threats mentioned in section 2.2.5 are addressed using techniques such as:

(i) Shelter-Node's Generation and Encryption of Messages
(ii) Signing Message at Shelter-Node to Avoid Bundle Store Overflow Attack
(iii) Handling Identity Spoofing Attack using Group Based Authentication
(iv) Challenge-Response Technique
(v) Key Encryption Technique
(vi) Verification of Messages at Forwarder-Node
(vii) Preventing Black Hole Attacks using Encounter Tokens
(viii) Encounter Token Verification at Forwarder-Node

2.2.11.1 Shelter-Node's Generation and Encryption of Message

A shelter-node generates categorized messages containing information about the requirements in that shelter. Suppose, shelter-node SID_k, creates a message *M* for the relief camp of the j_{th} group, *j = 1…..., n* with CID_j at time T_t. To provide end-to-end confidentiality to situational messages from shelter-node SID_k to the relief camp CID_j, the shelter-node encrypts *M* with modified camp-pin $MCPin_j$ (Basu *et al.*, 2015) as:

$$E_{MCPin_j}(M)$$

With this, only the camp-node of the j_{th} group can decrypt the message to obtain M with its camp-pin $CPin_j$ (Basu *et al.*, 2015) as:

$$D_{CPin_j}\left(E_{MCPin_j}(M)\right)$$

Confidentiality of sensitive situational messages is preserved this way, as it prevents malicious nodes to eavesdrop messages destined to the relief camps. The system provides confidentiality to commands, alerts and disaster management instructions generated from the main control station for the different rescue groups. The control station encrypts messages with modified group pins so that only volunteers of the intended group can decrypt them.

2.2.11.2 Signing Message at Shelter-Node to Avoid Bundle Store Overflow Attack

To avoid bundle store overflow attack at the shelter-node, SID_k generates a signature S_k to digitally sign the generated message using its shelter-pin SP_{in_k} (Basu *et al.*, 2015) as:

$$S_k = E_{SPin_k}\left(H\left(E_{MCPin_j}(M), SID_k, CID_j, T_t\right)\right) \qquad (2.9)$$

where *H* is a one way hash function. Shelter-node SID_k then forwards the augmented message *Mkj* for camp-node CID_j, where:

$$M_k{}^j = \left\{M_{CPin_J}(M), SID_{k,} CID_J, T_t, S_k\right\} \qquad (2.10)$$

Volunteers of the group j or volunteers who meet members of group j frequently transmit $M_k{}^j$ to the relief camp CID_j.

2.2.11.3 Handling Identity Spoofing Attack Using Group Based Authentication

Before involving in any forwarding activities by any two nodes, they is mutual authentication between them to avoid the identity spoofing attack from unauthorized nodes (Basu *et al.*, 2015). As such, whenever FN_i of group *i* intends to forward a shelter message to FN_j of group *j*, they need to authenticate each other as registered members of valid rescue-groups, since only volunteers who are registered with a valid rescue-group are provided with the modified pin values. Therefore, both FN_i and FN_j ask each other for their *GIDs* and *MGPs* to authenticate each other. These *MGP* values cannot be exchanged over the network as plaintext because unauthorized nodes that are not registered members of any group may eaves drop these values and use them in the future to claim authenticity. Basu *et al*., (2015) suggest two mechanisms to address this issue, one using challenge-response technique and the other using key encryption technique.

2.2.11.4 Challenge-Response Technique

In the challenge-response technique, FN_i generates a random challenge string, say challenge (FN_i) and sends it to FN_j . FN_j generates another random challenge string, say *Challenge (* FN_j *)* and computes a hash of MGP_j + *Challenge(* FN_i *)* + *Challenge(* FN_j *)* (Basu *et al.*, 2015) as:

$$H_j^{\,j} = H\left(MGP_j + challenge\left(FN_i\right) + challenge\left(FN_j\right)\right) \tag{2.11}$$

Where '+' denotes normal string concatenation.

FN_j sends GID_j, *Challenge (* FN_j *)* and $H_j^{\,j}$ to FN_i . FN_i searches the modified group-pin MGP_j corresponding to GID_j from the list of pins maintained within it and computes the hash of + *Challenge(* FN_i *)* + *Challenge(* FN_j *)* (Basu *et al.*, 2015) as:

$$H_i^{\,j} = H\left(MGP_j + challenge\left(FN_i\right) + challenge\left(FN_j\right)\right) \tag{2.12}$$

FN_i Compares H_i^j and H_j^j. If the comparison results in a success, FN_i verifies FN_j as a registered member of group *j* and considers FN_j as an authentic receiver. Next, FN_i computes a hash of MGP_i + *Challenge(* FN_i *)* +*Challenge(* FN_j *)* (Basu *et al.*, 2015) as:

$$H_i^i = H\left(MGP_i + challenge\left(FN_i\right) + challenge\left(FN_j\right)\right) \tag{2.13}$$

and sends GID_i and H_i^i to FN_j. FN_j searches the modified group-pin MGP_i corresponding to GDI_i and computes the hash of MGP_i + *Challenge(* FN_i *)* + *Challenge(* FN_j *)* (Basu *et al.*, 2015) as:

$$H_j^i = H\left(MGP_i + challenge\left(FN_i\right) + challenge\left(FN_j\right)\right) \tag{2.14}$$

FN_i Compares H_i^i and H_j^i. If the comparison results in a success, FN_i verifies FN_j as a registered member of group *j* and considers FN_j as an authentic sender.

2.2.11.5 Key Encryption Technique

In this technique, a forwarder-node FN_j encrypts MGP_j with the group-pin GP_j *(Basu et al., 2015) as:*

$$E_{GPj}\left(MGP_j\right)$$

This encrypted pin along with GID_j [that is, $GID_j, E_{GP_j}\left(MGP_j\right)$] is then forwarded to FN_i. Now, the pair (GP_j , MGP_j) has been so generated that data encrypted with GP_j can be decrypted with MGP_j and vice-versa. Therefore, FN_i, who has with it the MGP_j of group *j*, decrypts the data as:

$$E_{GP_j}\left(MGP_j\right) asD_{MGP_j}\left(E_{GP_j}\left(MGP_j\right)\right)$$

FN_i Compares MGP_j with the list of modified group pins for all groups to get MGP_j. If the outcome is successful, FN_i verifies FN_j as a registered member of group *j* and hence considers FN_j as a forwarder. Similarly, FN_j validates FN_i as a registered member of group *i*. Since the modified group-pin MGP_j is distributed only among registered volunteers, intruders to the

system will not be able to decrypt the encrypted pin without the knowledge of MGP_j. These help authenticate two participating nodes as registered rescue-group members and disallow unauthorized nodes from participating in the forwarding activities without the intervention of central authentication system.

2.2.11.6 Verification of Message at Forwarder-node

Before accepting and storing the message $M_k{}^j$, any intermediate forwarder-node decrypts S_K with $MSPin_k$ (Basu *et al.*, 2015) as:

$$D_{MSPin_k}\left(E_{SPin_k}\left(H\left(E_{MCPin_j}(M), SID_k, CID_j, \mathrm{T}_t\right)\right)\right)$$

This gives a decrypted hash, $H\left(E_{MCPin_j}(M), SID_k, CID_j, \mathrm{T}_t\right)$

The node computes a hash of $\left\{E_{MCPin_j}(M), SID_k, CID_j, \mathrm{T}_t\right\}$ as

$$H' = H\left(E_{MCPin_j}(M), SID_k, CID_j, \mathrm{T}_t\right) \quad (2.15)$$

It then compares H' with the decrypted hash. If the two hashes match, the forwarder-node trusts that the message has been sourced from SID_k and puts it in its buffer. Based on these, the buffer of a volunteer node cannot be filled with fake messages for the destination for which the volunteer has the highest delivery predictability by malicious nodes as the intermediate nodes would have verified the origin of the message before storing it in its buffer.

2.2.11.7 Preventing Black hole attacks Using Encounter Tokens

Digitally signed encounter tokens was proposed by (Basu *et al.*, 2015) to restrict malicious nodes from intercepting messages from other nodes by setting high delivery predictabilities for all destinations in the network to prevent black hole attacks. As such, if a forwarder-node FN_i encounters another node FN_j, a new encounter evidence is generated (Basu *et al.*, 2015) as:

$$\left\{FN_i, FN_j, GID_i, GID_j, \mathrm{T}_{ts}\right\}$$

This serves as evidence of the encounter with FN_i belonging to group i and FN_j belonging to group j at time T_{ts}. FN_i generates a signature S_i to digitally sign the encounter evidence using its group-pin GP_i (Basu *et al.*, 2015) as:

$$S_i = E_{GP_i}\left(H\left(FN_i, FN_j, GID_i, GID_j, \mathrm{T}_{ts}\right)\right) \quad (2.16)$$

It then creates for FN_j, a digitally signed encounter token (Basu *et al.*, 2015) as:

$$E_{i \longrightarrow j} = \left\{FN_i, FN_j, GID_i, GID_j, \mathrm{T}_{ts}, S_i\right\} \quad (2.17)$$

FN_i gives $E_{i \longrightarrow j}$ to FN_j.

Similarly, FN_j generates a signature S_j to digitally sign the contact evidence using its GP_j as (Basu *et al.*, 2015)

$$S_j = E_{GP_j}\left(H\left(FN_i, FN_{j,} GID_i, GID_j, T_{ts}\right)\right) \quad (2.18)$$

It then creates a digitally signed encounter token for FN_i as

$$E_{j \longrightarrow i} = \left\{FN_i, FN_j, GID_i, GID_j, T_{ts}, S_j\right\} \quad (2.19)$$

FN_j gives $E_{j \longrightarrow i}$ to FN_i.

This implies that, FN_j has evidence of meeting a member of group i as it already has a digitally signed encounter token from a registered member of that group. The number, $N\left(E_{i \longrightarrow j}\right)$ of such digitally signed encounter tokens depicts the number of encounters with members of group *i*. Similarly, FN_i collects such tokens from members of group *j*. Likewise, FN_i has evidence of meeting a member of group j as it already has a digitally signed encounter token from a registered member of that group. The number, $N\left(E_{j \longrightarrow i}\right)$ of such digitally signed encounter tokens depicts the number of encounters with members of group *j*.

2.2.11.8 Encounter Token Verification at Forwarder-Node

Each time FN_k which is a member of group k wishes to forward a message destined for the i^{th} group camp-node having a camp-id CID_i through FN_j, it collects the group-DP $P(FN_j, GID_i)$ and other information to enumerate FN_j as a suitable forwarder of such a message. FN_k also collects the digitally signed encounter tokens $E_{i \longrightarrow j}$ as evidence of encounters of FN_j with members of group *i* that have accumulated with FN_j since their last encounter. FN_k decrypts *Si* for each $E_{i \longrightarrow j}$ with MGP_i (Basu *et al.*, 2015) as:

$$D_{MGP_i}\left(E_{GP_i}\left(H\left(FN_i, FN_j, GID_i, GID_j, T_{ts}\right)\right)\right)$$

FN_k then obtains a decrypted hash $H\left(FN_i, FN_j, GID_i, GID_j, T_{ts}\right)$.

FN_k computes a hash of $\left\{FN_i, FN_j, GID_i, GID_j, T_{ts}\right\}$(Basu *et al.*, 2015) as:

$$H' = H\left(FN_i, FN_j, GID_i, GID_j, T_{ts}\right) \tag{2.20}$$

And compares *H'* with the decrypted hash. If the two hashes match, FN_k believes that the encounter token is genuine and not manipulated by FN_j. Similarly, FN_k verifies each encounter token and counts the number of such tokens. A high value of $P(FN_j, GID_i)$ close or equal to 1, claims that FN_j has met members of group *i* frequently, since its last encounter with FN_k. But, as a malicious node behaving as black hole may set its delivery predictabilities for all destinations to a very high value, such a claim needs to be supported by $N(E_{i \longrightarrow j})$, the actual number of encounters with members of group *i*. As such, FN_k takes decision about FN_j based on the following algorithm (Basu *et al.*, 2015) as:

Case I: If $P\left(FN_j, GID_i\right) > P\left(FN_k, GID_i\right)_{old}$ *and,* $N\left(E_{i \longrightarrow j}\right) > N\left(E_{i \longrightarrow k}\right)$ then FN_j is a suitable forwarder of messages destined to CID_i.

Case II: If $P\left(FN_j, GID_i\right) < P\left(FN_k, GID_i\right)_{old}$ *and,* $N\left(E_{i \longrightarrow j}\right) < N\left(E_{i \longrightarrow k}\right)$ then FN_j is a not suitable forwarder of messages destined to CID_i.

.***Case III:*** If $P\left(FN_j, GID_i\right) > P\left(FN_k, GID_i\right)_{old}$ *and,* $N\left(E_{i \longrightarrow j}\right) < N\left(E_{i \longrightarrow k}\right)$ then FN_j is malicious and is trying to attract messages destined to CID_i and might drop them.

From Case 1, FN_j is considered a suitable forwarder of messages destined to CID_I as FN_j claims to have a greater group-DP with members of group i than FN_k since the claim is validated by the fact that FN_j has more encounter tokens signed by members of group *i*, than FN_k has. Similarly, in *Case II*, FN_j exhibits a lesser group-DP with members of group *i*, than FN_k, which is justified by the number of digitally signed encounter tokens it has. However, in *Case III*, FN_j's announcement of having a greater group-DP with members of group *i* than FN_k

is not verified by the number of digitally signed encounter tokens it has. As a matter of fact, FN_K, having more such tokens has evidently met more members of group *i* than FN_j has, as such, FN_k detects FN_j as suspicious, refrains from forwarding messages to it and enlists it as a potential black hole. FN_k applies the three rules of the three cases on all other group-DPs and encounter tokens provided by FN_j to ensure that the DPs provided by it are pure.

2.2.12 Node Cooperation in Opportunistic Network

In OppNets, the cooperation of nodes can be considered as the nodes probability either to drop a message copy upon its reception or to forward the message to its encountering node. The main requirement of OpptNets is that the participating nodes should be unselfish, since communication is performed with the help of other nodes. However, this might not always be the case, since selfish nodes might decide that they do not want to help others. Such nodes should be detected and not allowed to participate in the dissemination process. This way, their messages will not be delivered, so they will be forced to become unselfish if they want a good networking experience. Cooperative behavior of nodes will largely affect the performance of OppNets, For example, lack of node cooperation, where a node may refuse to act as a relay and only settle for sending and receiving its own data, causes considerable delay degradation in the networks. To deal with this issue, the general method is to enable trust across communicating entities. The establishment of trust can evaluate nodes trust level and resist the Sybil nodes more effectively in OppNets. It further helps in identifying the malicious behavior of nodes in the network. A malicious behavior leads to a considerable delay in the message delivery or no delivery at all. Cooperation and trust between nodes in the network saves them from malicious attacks. The trust of a node is the basic value that symbolizes the magnitude of its social responsibility in the network, which include helping groups of nodes in message delivery, saving these nodes from malicious attacks, timely delivery of messages, just to name a few.

2.2.13 Helsinki simulation area

Helsinki is the capital of Filland, It is a benchmark for validation in an OppNet. Researchers of opportunistic networks use the Helsinki region to validate results due to the fact that the

opportunistic network environment simulator originated from this region. An example of this is a research carried out by Verma and Srivastava, (2012), Keranen and Ott, (2007)).

2.3 Review of similar works

Considerable work has been done by many researchers relevant to the subject area of security in PRoPHETrouting protocol of an OpptNets. The key ones are critically reviewed as presented here under:

Shikfa *et al., (2010)*proposed a topology-dependent key management system to address the problem of key management with the aim of providing a complete solution dedicated to Multiple Layer Commutative Encryption (MLCE). The technique was in two phases. At first setup phase, nodes were connected to an Identity Manager (IM) that generated and distributed these anchors in the form of certificates. The IM certified that a given node has one and only one pseudonym which in turn guaranteed that a node could impersonate other nodes, since the pseudonym of a node was strongly linked with its real identity. This prevented malicious nodes from simulating multiple nodes and thus could not access any private message they were not authorized to. At the second phase, their secure key agreement scheme depended on the topology of the neighbor and nodes were able to determine the position of their neighbors and security associations were bootstrapped along with neighborhood discovery as well. This technique enables the discovery of the neighbor's topology and could withstand tampering by malicious nodes. However, nodes could have several identities, which would lead to Sybil attacks. Also, a node could pretend to be at several positions at the same time, and therefore break the multi-layer scheme security.

Li and Das., *(2010)* considered a reputation-assisted data forwarding solution for OpptNets (RADON) that was based on the notion of positive feedback messages (PFMs) which calculated the first hand information (FHI) and second hand information (SHI). The FHI, SHI and an ageing concept were used to derive the reputation engine. The trust calculations were based only on the number of bundles forwarded by the nodes. These were special confirmation messages that helped the reputation mechanism to monitor the behavior of a forwarder. They contained the IDs of the nodes exchanging data, as well as the number of total encounters between the two

nodes and the signature of the PFM creator, and were generated by a node after a message reception and disseminated epidemically in the network. They were used by nodes in the network which assessed the reputation of other nodes, since counters were updated for each forwarder depending on whether the PFM arrived or not in time (or if it arrived with the expected content). The result showed that, RADON prevented a malicious node from deliberately dropping and arbitrarily forwarding data, which dramatically improved the network performance. However, despite that feedback PFM, packet size was very small compared to ordinary data packet, RADON generated many feedback packets by using PFM mechanism and resulted in increased burden of the network, which is the major drawback. Furthermore, the reputation mechanism for opportunistic networks to counteract complicated attacks with collusion was not considered, since in collusion attacks, malicious nodes can forward data to each other to earn reputation and sometimes, even deliberately refuse to forward data to other nodes.

Trifunovic *et al.*, (2010) developed two approaches for social trust establishment that were robust to Sybil attacks: explicit social trust and implicit social trust. The implicit social trust was meant to approve whether the node was honest or dishonest, whereas the explicit social trust method was designed to determine how much connected the nodes were with respect to handling the message passing via them. According to the researchers, since one cannot prevent users from generating multiple identities, one way to limit the influence of Sybil's was to proactively establish trust in the identities being genuine. The paper relied on both friend ties and conditional transitivity of trust, depending on hop distance and social interconnection. Due to the mobility of the devices, users could establish secure and reliable friend ties whenever they meet by secure pairing. The paper assumed conditional transitivity of trust depending on hop distance and connectivity, that is, trust in a user connected to several common friends was higher than in a user with one single connection over various hops. The result showed that, through chains of paired friends, the human entity behind the identity was verified which ensured that the identity was not Sybil and the approach limited the maximum number of Sybil's independently of the network size and was more robust against manipulation attacks. However, secure pairing requires conscious user interaction and cannot be performed automatically. Thus, the resulting graph will only be loosely connected without guarantees of regular interactions.

Chen *et al., (2012)*integrated social trust and Quality of Service (QoS) trust into a composite trust metric for determining the best node among the new encountered for message forwarding. They considered honesty and unselfishness for social trust to account for a node's trustworthiness for message delivery, and connectivity for QoS trust to account for a node's capability to quickly deliver the message to the destination node. According to the researchers, by assigning various weights associated with these QoS and social trust properties, a class of DTN routing protocols was formed, from which the paper examined two versions of the trust management protocol: an equal-weight QoS and social trust management protocol and a QoS trust only management protocol. The technique used for the analysis of the performance of their proposed trust-based routing protocol for DTN message forwarding was by a probability model based on stochastic Petri net (SPN) techniques. It was used to obtain each node's information (for example, connectivity and honesty) and to derive the trust relationship with other nodes in the system. Results demonstrated that a malicious node would never attain trustworthy status and by properly selecting weights associated with QoS and social trust metrics for trust evaluation, the trust management protocols achieved the ideal performance level in the delivery ratio and delay obtainable by epidemic routing, especially as the percentage of malicious nodes increased. However, other trust metrics such as technical competence, similarity, and cooperativeness of nodes where not considered, since Cooperative behavior of nodes will largely affect the performance of Opportunistic network.For example, lack of node cooperation, where a node may refuse to act as a relay and only settle for sending and receiving its own data, causes considerable delay degradation in the network.

Chang *et al.*, (2011) proposed a trust threshold-based routing protocol for delay tolerant networks (DTN), leveraging two trust thresholds for accepting recommendations and for selecting the next message carrier for message forwarding. The paper considered healthiness and cooperativeness for social trust to account for a node's trustworthiness for message delivery, connectivity and energy for QoS trust to account for a node's QoS capability to quickly deliver the message to the destination node. The paper considered DTN environment without a centralized trust authority and every node may have a different level of energy and speed reflecting node heterogeneity, the paper differentiated uncooperative nodes from malicious

nodes, and once a node became uncooperative, it stayed as uncooperative and acted to maximize its own benefit regardless of the global benefit of the DTN, so it might drop packets arbitrarily just to save energy. As soon as a malicious node was detected, the trust value of the malicious node was set to zero, and thus excluded it as a message carrier for message forwarding. The performance analysis results demonstrated that when operating under proper trust thresholds and social vs. QoS trust weight settings as identified in the paper, Trust Threshold Based Routing (TTBR) could effectively trade off message delay and message overhead for a significant gain in message delivery. However, an uncooperative node according to the researchers acted to maximize its own benefit regardless of the global benefit of the DTN. Consequently, it might drop packets arbitrarily just to save energy, but this can lead to considerable delay degradation and even congestion in the networks. Social network structure-based properties such as similarity, centrality, and betweenness were not considered because they did not use trust encounter histories exchanged to avoid self-promoting or false information attacks by malicious nodes, if some or all of these were considered, it would have yielded a better result.

Poonguzharselvi and Vetriselvi, ***(2012)*** implemented a trust framework for opportunistic network (OpptNets) where the nodes in the network follow the trace based mobility model and used trust value thresholds for selecting nodes and providing transmission services (Papaj *et al.*, 2012). The paper assumed trust values between neighboring nodes were similar trust levels between people and the value lies between one and zero. Initial trust value for source node was assumed to be one since they were fully trusted nodes, but for forwarder nodes, the initial trust values were calculated and which was based on friendship vector associated with nodes in the network, when data was transmitted, a node individually calculated the trust values between itself and the remaining nodes since there may be some malicious nodes that could drop or forward the data with long delivery delay. This eliminated malicious nodes, increased the security of data transmission and better delivery probability was obtained. However, the paper did not consider privacy to conceal the trace of the nodes for their activities not to be peeped by others to prevent intentionally collecting node's data and exposing node traces and improves the transmission privacy of users and privacy of mobile network equipment.

Gupta *et al., (2013)*proposed a Trust-based Security Protocol (TSP) to mitigate black hole attacks in opportunistic networks (OpptNets) that used PRoPHETas the underling routing protocol. In their proposed TSP protocol, the trust was not only based on the number of successfully transferred messages, but also on three fundamental pillars: Social Group Value (SGV), Creadits and Hop count. The proposed TSP scheme relied on the calculation of a social group value and a trust distribution technique. It was assumed that a particular node belongs to an individual that is a part of static society. A Static society means people that belong to a group of businessman, high class politicians, military people etc. Trust was calculated by the destination node and distributed to each node according to its hop number in the message. The destination did the necessary calculation of the trust value for each hop in the message vector. Trust was distributed among other pears that had participated in the delivery process. The destination node of the message used a backward path to achieve this distribution. Through this security feature, a malicious node was quickly identified since its trust value would never increase due to the fact that, this node did not participate in the routing operation. However, a higher trusted node does not guarantee higher delivery probability since it does not have an idea of the destination position. Furthermore, a node providing forged information regarding delivery probability was also not addressed, as it could be a malicious node that may eventually be trusted unknowingly thereby degrading the network performance.

Li and Das, *(2013)*proposed a Positive Feedback Message (PFM) as evidence of forwarding behavior of a node which were collected at the Watchdog and fed into the main body of their trust-based frame work.The functionality of the Watchdog embedded at a node, say node A, was to monitor the behavior of the other node, say node B, to which a data message has been forwarded from node A. In the paper, node B's forwarding behavior was evaluated in terms of the evidence Positive Feedback Message (PFM) created by another node, say node C, which got that data message further forwarded by node B. Specifically, after the data message was received, node C generated a PFM destined to node A with the intention of convincing node A that node B has helped forward data successfully. Each node that sent out data to its next-hop forwarders, kept record for each of those forwarders how many PFMs corresponding to data sent out had and had not come back by using two counters which helped them form the reputation and trust opinion to another node's forwarding behavior in the future, the trust-based framework

was integrated with PRoPHET, a data forwarding protocol in opportunistic network, and showed that two kinds of attacks, deliberately dropping data and arbitrarily forwarding data, was effectively counteracted and the result also demonstrated the effectiveness of their proposed framework on improving the network performance under black hole attack. However, the paper employed epidemic forwarding for PFM transmission which degrades the network performance due to the increased burden on the network as a result of flooding the network with the PFM, and also, there was no privacy protection for the context-information which may be peeped in to by others.

Chen *et al.*, (2014)Designed and implemented a trust management protocol for Delay Tolerant Networks (DTNs) and applied it to secure routing to demonstrate its utility. The trust management protocol combined Quality of Service (QoS) trust with social trust to obtain a composite trust metric, QoS trust was evaluated through the communication network by the capability of a node to deliver messages to the destination node. The paper considered "connectivity" and "energy" to measure the QoS trust level of a node. The connectivity QoS trust is about the ability of a node to encounter other nodes due to its movement patterns. The energy QoS trust was about the capability of a node to perform the basic routing function. Social trust was based on honesty or integrity in social relationships and social ties. The paper considered "healthiness" and social "unselfishness" to measure the social trust level of a node. The healthiness social trust was the belief of whether a node was malicious. The unselfishness social trust was the belief of whether a node was socially selfish. The protocol was resilient against bad-mouthing, good-mouthing, and white washing attacks performed by malicious nodes. Trust values were calculated and arranged with different weights, each node was associated with this calculated true value. The researchers performed a comparative analysis of their routing protocol with simulation validation against existing trust-based (SReD) and non-trust based (PRoPHETand epidemic) protocols, and under identified best settings, their protocol outperformed SReD and PRoPHET, and approaches the ideal performance of epidemic routing in delivery ratio and message delay without incurring high message or protocol maintenance overhead. However, the privacy of the information itself which may be sensitive to users was not considered, this preventsintentionally collecting node data and exposing node traces and improves the transmission privacy of users and privacy of mobile network equipment.

Guo *et al.*, (2015)proposed an authentication mechanism with privacy protection for opportunistic networks. The proposed mechanism finished the authentication with less data, and provided anonymity and user privacy in the network. In their network environment, the nodes were registered and authenticated at the super node (immobile nodes that were set within Opportunistic networks) when entering the network area, privacy protection of a node were transmitted when the node was registered at the super node. During this process, the data stored in the super node were processed hash values, and plaintexts were not kept, as such, it protected the super node from privacy leakage. In addition, super nodes strictly provided node registration within networks, network topology was not maintained. Therefore, user and node mobility were not recorded and their trajectories not traced, thereby eliminating any concern regarding location privacy. Moreover, during each authentication transmission, each authentication used different data, excluding equipment data. This prevented intentionally collecting node data and exposing node traces and improved the transmission privacy of users and privacy of mobile network equipment. The results of the security analysis demonstrated superior security to other schemes in terms of man-in-the-middle attack, forgery. However, In their network environment, malicious attackers can tap and collect data, such as messages transmitted between devices and the super node during authentication processes and also, secure channels were absent during registration, nodes provided strictly their own public keys and ID hash values. Malicious nodes that present intention to forge the super node and bait node data may obtain the public keys and ID hash values.

Basu *et al.* (2015)presented a security aided and group encounter based PRoPHETrouting protocol that disseminated situational messages and avoided malicious nodes in the network. Their proposed protocol helped forwarder nodes to mutually authenticate each other, guarded the network from black-hole and other severe attacks and also ensured end-to-end confidentiality of sensitive situational messages. The central command and control station, which was assumed to be a trusted authority, distributed keys to different emergency groups (shelters and relief camps) at the set-up phase. After that was done, the central control station had no role in the runtime phase when forwarder-nodes checked authenticity of peers and forward situational messages using the secured PRoPHETrouting protocol in a fully decentralized fashion. However, they did

not consider the issue of compromised or hijacked nodes, where an unauthorized user may physically capture a node and maliciously use the group pins (GPs) and modified group pins (MGPs) stored in it to claim authenticity, and also, the creditability which is the trust that ensures the magnitude of cooperation in the delivery of message in the network was not considered as well which will lead to Sybil attacks. On the other hand, PRoPHETassumes that nodes in the network are trusted and cooperate towards message forwarding. Such assumption turns out inaccurate in presence of malicious nodes that may severely impede the delivery, accuracy and timeliness of situational messages. Therefore, integrating cooperative component with their PRoPHETis extremely important and can resist the Sybil nodes more effectively.

It is evident from the reviewed researches that Opportunistic Network (OpptNets) suffers from a lot of security problems including unauthorized access, message tempering, Denial of Service (DoS), privacy invasion, non-cooperative nodes and confidentiality disclosure. This research work is an improvement of Basu *et al.* (2015) which has a limitation of non-cooperative nodes. The challenge at hand is to ensure availability and to resist malicious dropping in OpptNets where a node may refuse to act as a relay and only settle for sending and receiving its own data or information, thus, causing considerable delay degradation in the network. Furthermore, the improvement helped in identifying the malicious behavior of nodes in the network which cause considerable delay in the message delivery. The improvement involved developing a node cooperation routing capable to ensure improvement in network security and was implemented in an Opportunistic Network Environment (ONE) simulator. Validation was done by comparing results obtained with the results of Basu*et al.*, (2015).

CHAPTER THREE

MATERIALS AND METHODS

3.1 Introduction

This chapter describes the procedure carried out in the opportunistic network modeling using the PRoPHETrouting protocol in the ONE simulator. The modeling and simulation process of the improved Security Aided Group Encounter PRoPHETRouting with node cooperation is explained in this chapter.

1. Setting up the java platform and runtime environment by:
   (i) Downloading and installing the *JDK*.jar,
   (ii) Configuring the environment variables.

2. Setting up the Java Development Environment (IDE) by:
   (i) Downloading the IDE,
   (ii) Unzipping the IDE and extracting the Eclipse for configuration.
3. Setting up the ONE simulator by:
   (i) Downloading the ONE 1.5.1-RC2,
   (ii) Extracting the downloaded simulator in (i),
   (iii) Compiling the simulator using the command prompt.

4. Interfacing the IDE in item (2) with ONE 1.5.1-RC2 in item (3) by:
   (i) Creating the IDE work space,
   (ii) Creating the project name,
   (iii) Setting-up the project directory to import the ONE 1.5.1-RC2 source files,
   (iv) Exporting the ECLA, DTN connection, and Junit library files.
5. Setting up the routing protocol by:
   (i) Modifying and developing the routing class *(PRoPHET.class)*,
   (ii) Incorporating the routing class into the ONE simulator core files,
   (iii) Developing the setting scenario for the routing class for simulation,

(iv) Simulating a test bed for the developed scenario.

6. Developing and incorporating the node cooperation into the routing protocol in item (5).
7. Simulation and validation with the methods proposed in Basu et al, 2015.

3.2 Modelling the Java Platform

In order to develop the proposed technique, the java development kit (jdk) was downloaded and was installed. This is done in order to gain access to the relevant java virtual machine. The procedures are discussed as follows:

3.2.1 Java Development Kit

Java SE was downloaded from http://www.oracle.com/java/javase/downloads/index.html. we ran the downloaded installer (e.g., "jdk-8u{xx}-windows-x64.exe"), which installs both the JDK and JRE. By default, both files were installed in the program file directory. The snippet showing the installed directory is show in Fig. 3.1

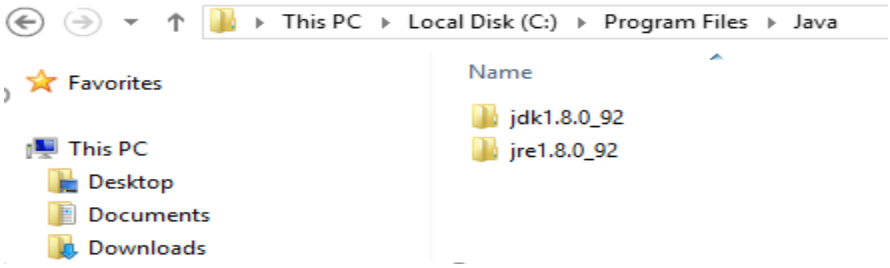

Fig. 3.1: The jdk and jre Installed Directory

3.2.2 Configuring of Environment Variables

Windows OS searches the current directory and the directories listed in the PATH *environment variable* for executable programs. JDK's programs (such as Java compiler javac.exe and Java runtime java.exe) reside in directory bin installed in 3.2.1, the procedures are highlighted as follows:

(i) Launch ⇒ "System" ⇒ Click "System properties", then to "Advanced system settings".

(ii) Switch to "Advanced" tab ⇒ "Environment Variables".

(iii)Under "System Variables", scroll down to select "Path" ⇒ "Edit...

The Snippet of the figure showing the directory implementation of these steps is given in Fig. 3.2

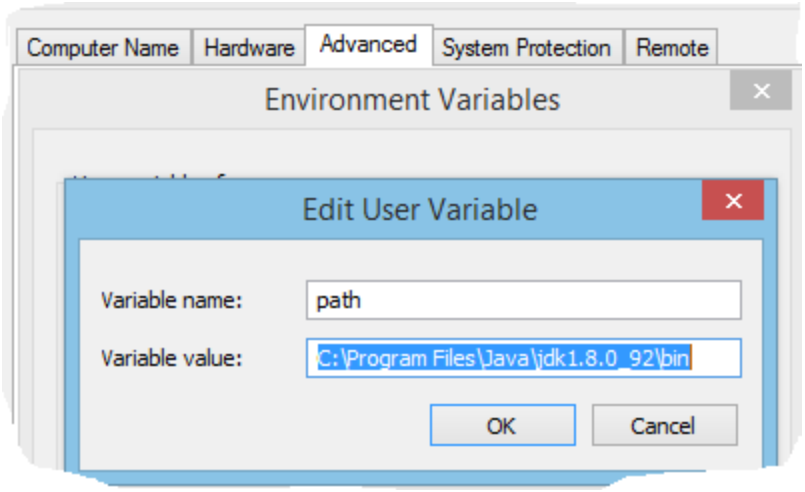

Fig. 3.2: Configuring the Environment Variable

After successfully setting up the java path in the environment variable as shown in Fig. 3.2, the java classes module in the java virtual machine were verified using *java* and *javac* command in the system command prompt.

3.3 Setting up the Java Development Environment

The IDE for Java Developers was downloaded from http://www.eclipse.org/downloads/ to debug and run the ONE project. The downloaded Eclipse is uncompressed and compiled in order to get access to the relevant source file and the IDE environment. The ONE simulator source codes were then imported into the eclipse IDE environment using the following steps.

(i) Open the Eclipse IDE and choose the desired workspace. In this research, the default workspace was used.

(ii) Click on the file menu on the Eclipse IDE environment and select the desired project. In this research, the "java project" was employed. This is to enable hand on interface and development of the technique. The snippet of this approach is shown in the Fig. 3.3

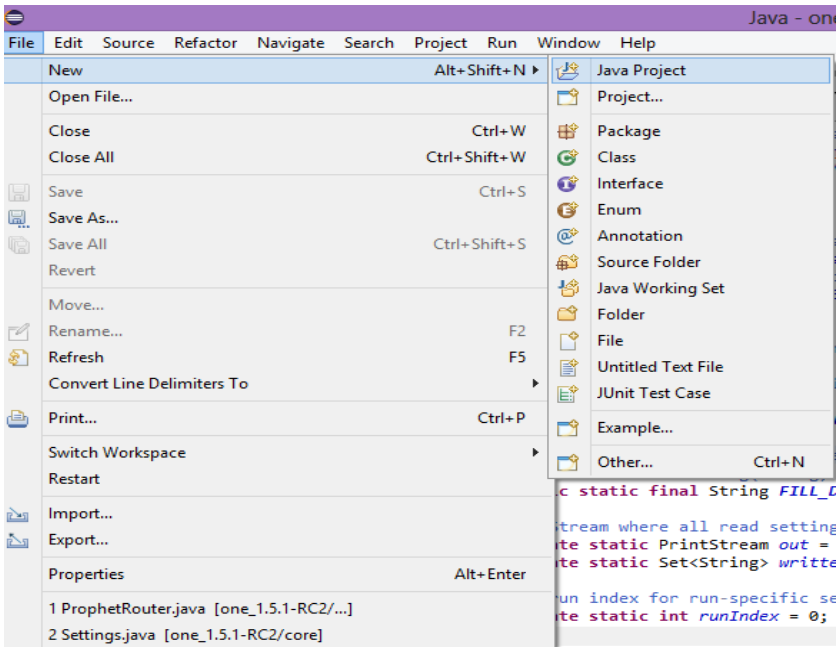

Fig. 3.3: Creating the ONE project in Eclipse IDE

The java project option in Fig. 3.3 enables the user to import the necessary ONE simulator file and also import the necessary libraries.

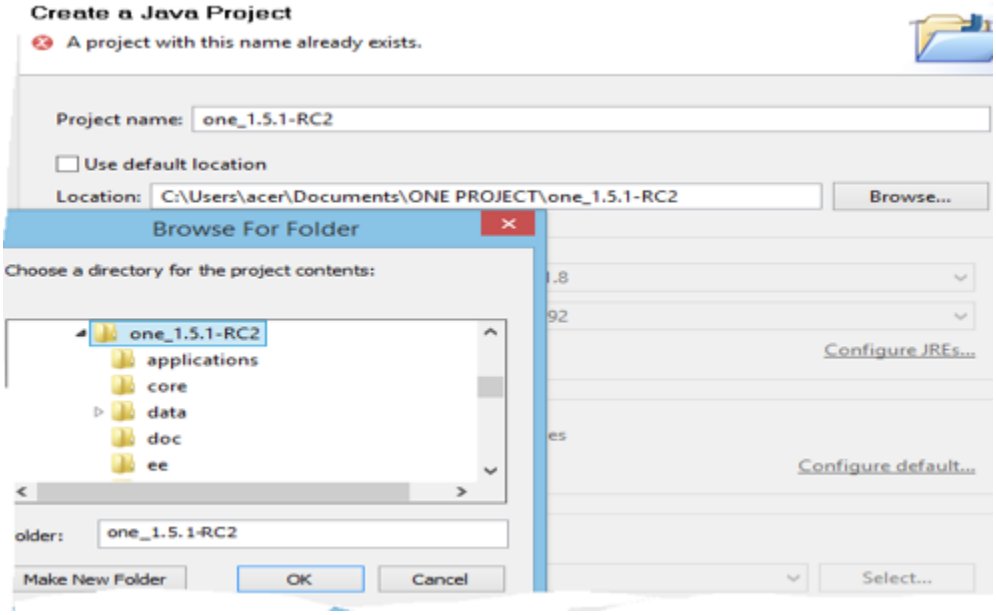

Fig. 3.4 Create a New Project from Eclipse

After we finished selecting" Java Project", we will get a configuration window. "use default location" was unchecked and we navigated to our ONE PROJECT folder "ONE PROJECT" one

1.5.1-RC2" in order to import our ONE PROJECT. We can define the project name by ourselves. In this report, we defined the project name as "one 1.5.1-RC2".

3.4 Setting up the ONE Simulator

In setting up the ONE simulator, the following processes were followed:

3.4.1 Download the ONE 1.5.1-RC2

A compressed source code of the ONE simulator was downloaded from http://www.netlab.tkk.fi/tutkimus/dtn/theone/down/one_1.5.1-RC2 in to a folder called "ONE PROJECT". The downloaded simulator was un-compressed from the command prompt using \tar xvf one 1.5.1.-RC2 "command to un-compress the package. This automatically creates a folder called \one 1.5.1-RC2" in our \ONE PROJECT" folder. After uncompressing the package of the ONE, the command "./com-pile.bat" was used to compile the Java codes and enabled simulation.

.

3.5 Interfacing the IDE with ONE-RC

Three important "jar" (DTNConsole-Connection.jar, ECLA.jar and junit.jar) are required to compile the ONE with the eclipse IDE and correct any necessary error. This was done by going to the folder "ONE PROJECT/one 1.5.1-RC2 /lib" and import DTNConsoleCon-nection.jar, ECLA.jar and junit.jar. The DTNConsole-Connection.jar allows the DTN simulator to use command to communicate with DTN simulated scenarios. The ECLA.jar is useful to simulate Network Interface Hardware and it tries to finish the work on Link Layer and the junit.jar is used to correct any configuration error in the simulator. At this point, the ONE PROJECT have been configured in eclipse environment.

3.6 Setting up the Routing Protocol

In the configured ONE simulator environment, a source package named "routing" manages the entire routing class. This gives the permission to designers to incorporate their own new routing algorithm or to modify the existing routing algorithm. In this research, a post disaster scenario based PRoPHET routing protocol considering node cooperation was developed.

3.6.1 Modelling PRoPHETRouting Protocol on Test Case

In this research, an opportunistic network environment consisting of a total of 20 (twenty) nodes were used as a test case scenario. This is with the aim of evaluating the proposed model before applying it to the Helsinki region for validation. It is important to note that the choice of twenty (20) nodes is at the discretion of the researcher and this number could be varied as the case may be. The mode of communication is assumed to be Bluetooth with a transmission range and speed of 10m and 2Mbit/s. The devices available for communication were assumed to be either a laptop, a mobile phone or a work station with a message buffering size capacity of 10MB to 15MB. For a high-speed movement, a transmission speed of 10Mbit/s and 1000m range was adopted. This setting is configured and saved in the same directory as the ONE simulator source files. The following java script was used to import the setting during simulation.

```
publicclass Settings {
    /** properties object where the setting files are read into */
    protectedstatic Properties props;
    /** file name of the default settings file ({@value}) */
    publicstaticfinal String DEF_SETTINGS_FILE ="Husseina_Test.txt";


    publicstaticfinal String SETTING_OUTPUT_S = "Settings.output";
```

The “Husseina_Test.text” in the script is the name of the file containing the settings described above. The ProPhet routing is implemented in the setting file using the following script.

```
    # Scenario settings
    Scenario.name = HUSSEINAH_TEST
    Scenario.simulateConnections = true
    Scenario.updateInterval = 1.0
    Scenario.endTime = 44000
    Scenario.nrofHostGroups = 1
    btInterface.type = SimpleBroadcastInterface
    btInterface.transmitSpeed = 250k
    btInterface.transmitRange = 10
```

highspeedInterface.type = SimpleBroadcastInterface

highspeedInterface.transmitSpeed = 10M

highspeedInterface.transmitRange = 1000

Group.router = ProphetRouter

ProphetRouter.secondsInTimeUnit=1

The complete program for the Prophet routing setting on the test node can be found in appendix A1. A snippet of the initial simulation scenario for the twenty test node is given in Fig. 3.5.

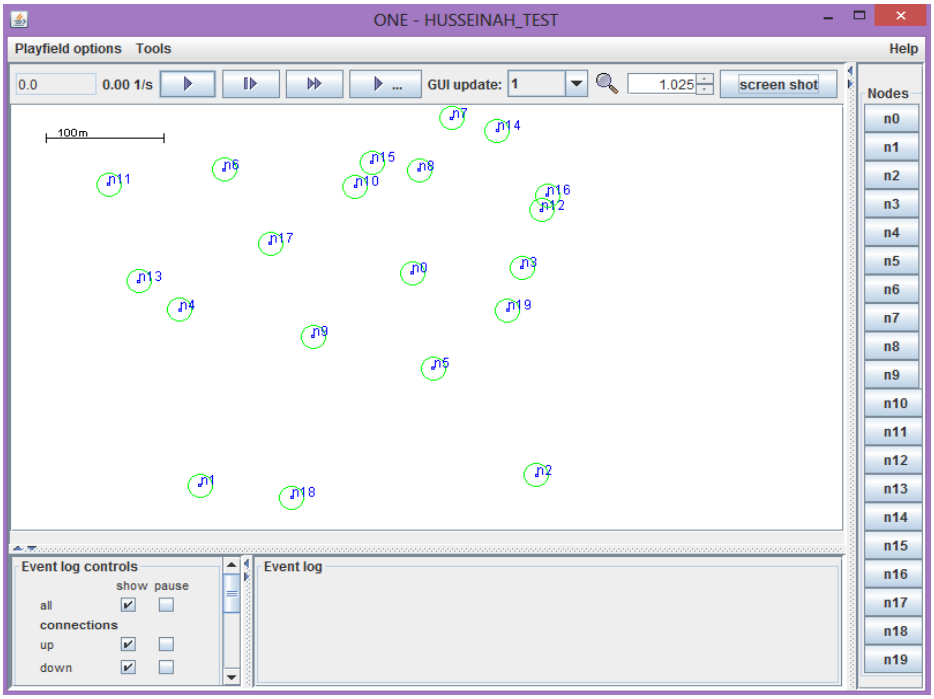

Fig. 3.5: Snippet of the Twenty Test Nodes

The snippet of the massage builds up process of the test node given in Fig. 3.5, is shown in Fig. 3.6.

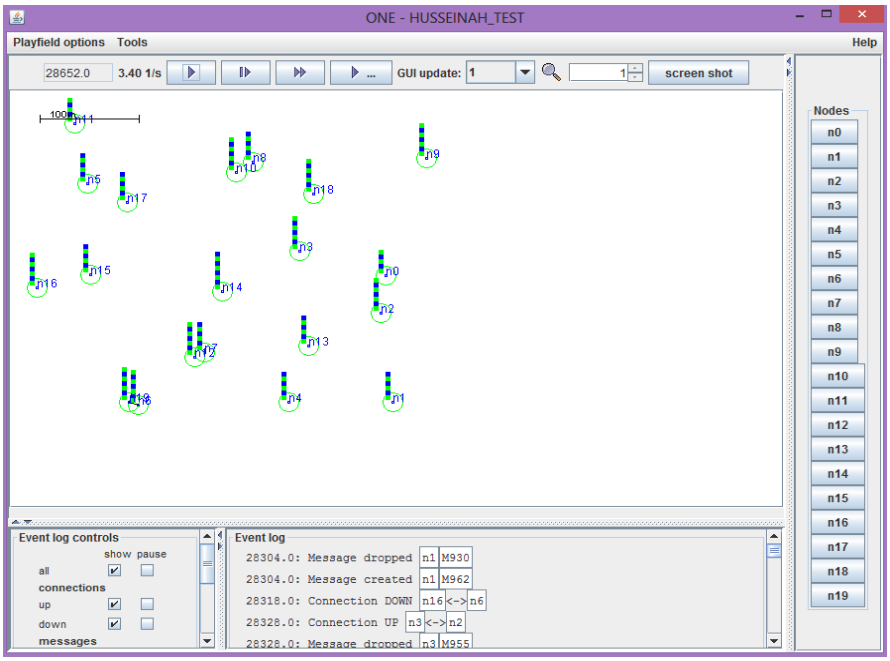

Fig. 3.6: Message Build of the Test Nodes

3.6.2 Modelling Post Disaster based Scenario for PRoPHETRouting in Helsinki

Unlike in the standard Helsinki region, where nodes are made of pedestrians, cars and Trans, this research considers a post disaster scenario. Since in a post disaster scenario, rescue and relief operations are carried out by different response group, this research assumes that messages are categorized on the basis of their relevance to a particular group. Thus, in this research, the following assumptions were considered.

1. Only four different types of nodes exist in a post disaster scenario.
2. Communication can only exist via Bluetooth with a communication range of 10m and a speed of 2Mbit/s.
3. Messages are grouped and categorized on their relevance to a particular group and are there by forwarded based to their corresponding group relief camp.

The choice of these assumptions was made in order to depict the real life scenario of a post disaster in terms of simulation as much as possible. Based on assumption I, the nodes involved are detailed as follows:

(i) **Transport Nodes (*TN*):** Theyconsist of vehicles (e.g., any type of vehicle capable of carrying food items) capable of generating situational messages about when water and other shelter items will be available.

(ii) **Shelter Nodes (*SN*):** The shelter node in this case could be a laptop, mobile phone or a work station which categorizes situational messages, stating the number of items (i.e., food, water, clothing, medicine, etc.) required in the shelter.

(iii) **Camp Nodes (*CN*):** This node received categorized messages from the different shelters and accesses their requirement to organize distribution of resources.

(iv) **Forwarded Nodes (*FN*):** These are nodes which moves around the disaster area and forward shelter messages to the relieve camps.

The message size in all these nodes ranges between 500kB to 1MB. The event generation range is between 20-35seconds and a scenario time-to-life of 5hours was assigned. All other scenario setting described in subsection 3.3. was adopted. Fig. 3.6 shows the Helsinki model of the post disaster scenario.

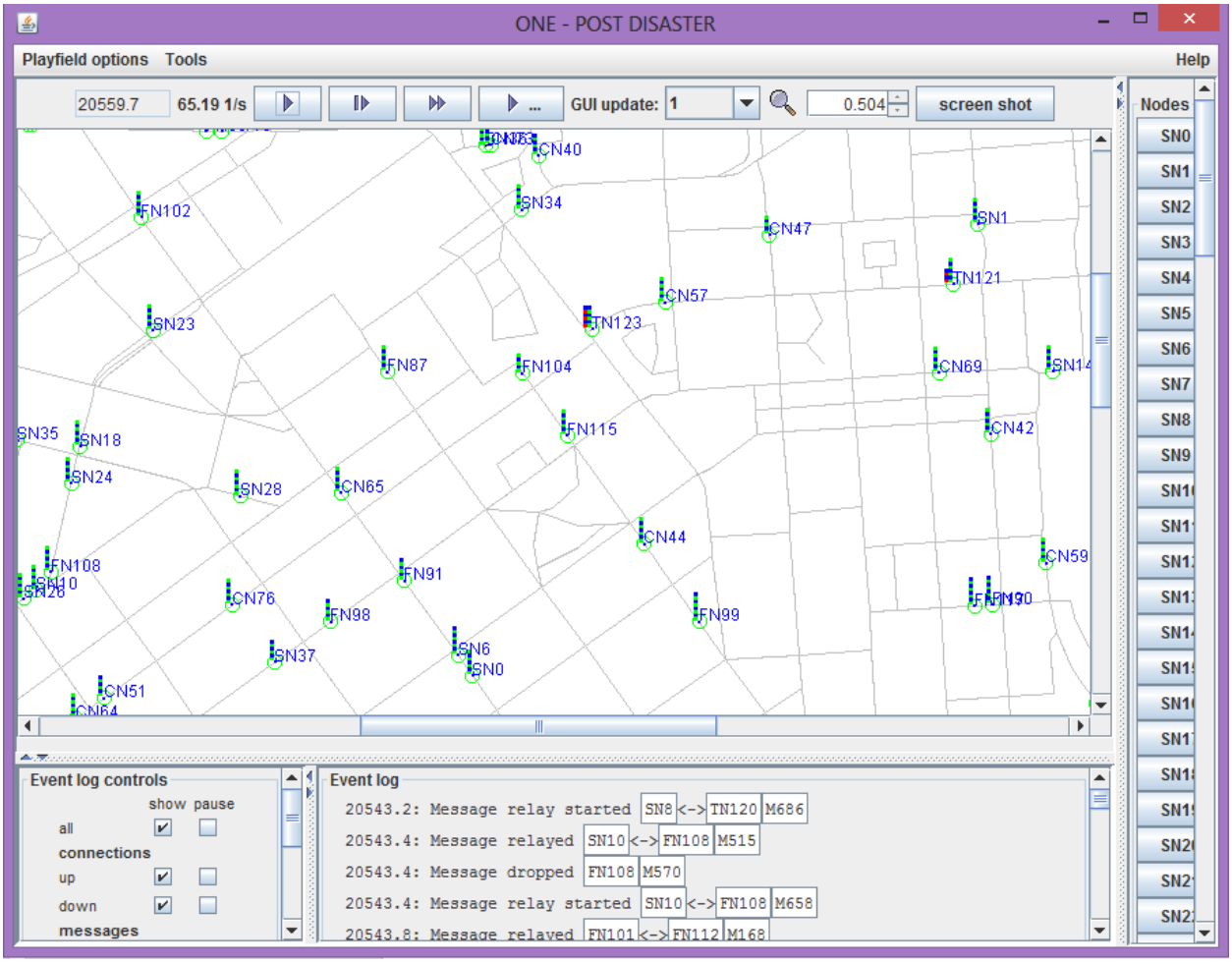

Fig. 3.7: Helsinki Simulation of Post Disaster Nodes

From Fig. 3.6, it can be observed that the post disaster nodes which are label FN, SN, CN and TN were implemented on the Helsinki platform. The post disaster node implementation subprogram is given in the following script.

```
Group1.groupID = SN

# group2 specific settings
Group2.groupID = CN
Group2.okMaps = 1
# 10-50 km/h
Group2.speed = 2.7, 13.9
Group3.groupID = FN
# The Tram groups
Group4.groupID = TN
Group4.bufferSize = 50M
Group4.movementModel = MapRouteMovement
Group4.routeFile = data/tram3.wkt
Group4.routeType = 1
Group4.waitTime = 10, 30
Group4.speed = 7, 10
Group4.nrofHosts = 2
Group4.nrofInterfaces = 2
Group4.interface1 = btInterface
Group4.interface2 = highspeedInterface
Group5.groupID = TN
Group5.bufferSize = 50M
Group5.movementModel = MapRouteMovement
Group5.routeFile = data/tram4.wkt
Group5.routeType = 2
Group5.waitTime = 10, 30
Group5.speed = 7, 10
Group5.nrofHosts = 2
```

Group6.groupID = TN

Group6.bufferSize = 50M

Group6.movementModel = MapRouteMovement

Group6.routeFile = data/tram10.wkt

Group6.routeType = 2

Group6.waitTime = 10, 30

Group6.speed = 7, 10

Group6.nrofHosts = 2

3.6.3 PRoPHETbased Node Cooperation

Mutual behaviours between the nodes, where nodes do not only send and receive its own data, but participate in shearing information from others was modelled in this research. In this research, specifically, cooperative behaviour is modelled for three nodes (shelter node, camp node and forwarded node) out of the four nodes considered for the post disaster scenario. This is because, the fourth (transport node) only participate at the time of need, thus does not have constant influence on the behaviour of the network. Assuming a node *NC,* is sending a message through the network to another node, let's say *FN*. The node cooperation technique is implemented as follows:

$$DN = \begin{cases} (CN, SN) & if\ CN\ \&\ SN \le FN \\ (CN, FN) & if\ CN\ \&\ FN \le SN \\ (SN, FN) & if\ SN\ \&\ FN \le CN \\ 0 & if \quad otherwise \end{cases} \tag{3.1}$$

Where, *DN* is the final destination node, *CN* in this case is the camp node, SN is the shelter node, and *FN* is the forwarder node. The setting subprogram for the node cooperation is given as follows:

package routing;

publicclass nodeCooporation {

Scenario settings

Scenario.name = NODE COOPORATION POST DISASTER

Scenario.simulateConnections = **true**

Scenario.updateInterval = 0.1

```
# 43200s == 12h
Scenario.endTime = 10000000
Scenario.endTime = 43200
# "Bluetooth"interfacefor all nodes
btInterface.type = SimpleBroadcastInterface
# Transmit speed of 2 Mbps = 250kBps
btInterface.transmitSpeed = 250k
btInterface.transmitRange = 10
highspeedInterface.type = SimpleBroadcastInterface
highspeedInterface.transmitSpeed = 10M
highspeedInterface.transmitRange = 10
```

The complete java program for the PRoPHETrouting based node cooperation can be found in Appendix A2. The subprogram implementation of the node cooperation on PRoPHETrouting is given as follows:

```
privatevoid updateTransitivePreds(DTNHost host){
        MessageRouter otherRouter = host.getRouter();
        assertotherRouterinstanceof ProphetRouter : "PRoPHET only works "+
                " with other routers of same type";
        doublepForHost = getPredFor(host); // P(a,b)
        Map<DTNHost, Double>othersPreds =
                ((ProphetRouter)otherRouter).getDeliveryPreds();
        for (Map.Entry<DTNHost, Double>e : othersPreds.entrySet()) {
        if (e.getKey() == getHost()) {
                continue; // don't add yourself
        }
        doublepOld = getPredFor(e.getKey()); // P(a,c)_old
        doublepNew = pOld + ( 1 - pOld) * pForHost * e.getValue() * beta;
                preds.put(e.getKey(), pNew);
        }
        If (e.getKey()<=getHost()){
```

```
        NC;
        Elseif(e.getKey()>=getHost()){
        FN;
        }
        else{
        continue;}
        }
    }
```

The flow chart implementation of the node cooperation based PRoPHETrouting for post disaster scenario is given in Fig. 3.7

From fig. 3.7, it can be observed that, the messages were initialized and the scenario name (Husseina_NCPro) was defined. The prophet routing was initialized and the routing process was started through the ONE simulator Graphical User Interface (GUI). Each node evaluated their messages and performed cooperation which is the basis of this work. Finally, the message was verified and stated reports were examined in order to determine the performance of the model.

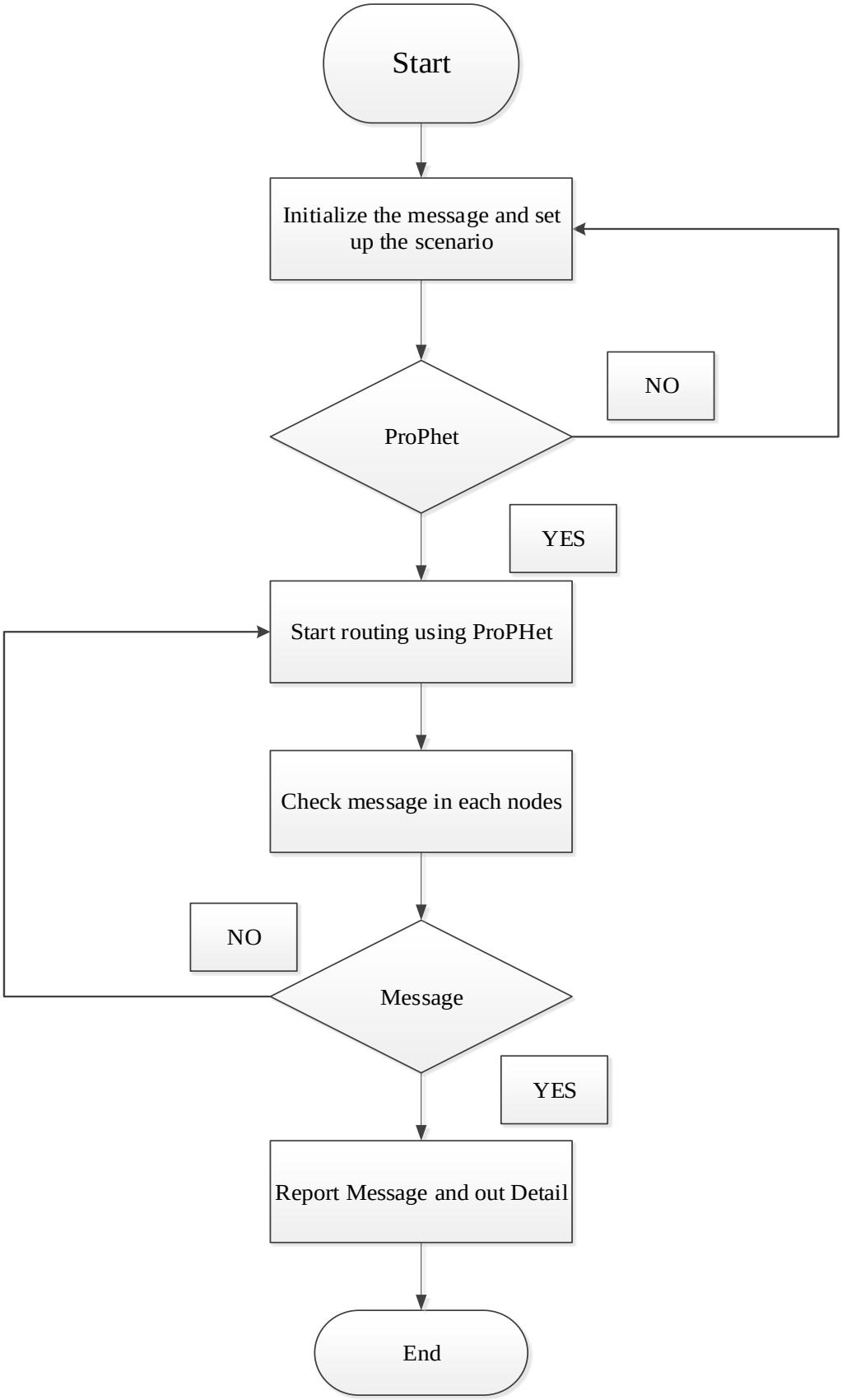

Figure 3.7: Flow Chart of Proposed Model

CHAPTER FOUR

RESULTS AND DISCUSSIONS

4.1 Introduction

In this chapter, the performance of the proposed Node Cooperation security based prophet routing protocol which improved the security aided and group encounter PRoPHETrouting protocol is evaluated. In order to efficiently evaluate the performance of the proposed model, three metrics (delivery probability, buffer time and hop count) were evaluated.

4.2 Performance Evaluation for Test Node

The performance of the developed model was first evaluated on twenty random nodes in order to ascertain its performance. The simulation was performed at discrete simulation time interval of 4000 seconds through 44000 seconds. Detailed results obtained for the twenty test node can be found in Appendix B1 without node cooperation and in Appendix B2 with node cooperation. Figure 4.1 shows the delivery probability of the improved security aided and group encounter prophet routing protocol as compared with the conventional security aided and group encounter PRoPHETrouting protocol for the 20-nodes test case.

Readers should note that, the model presented Basu *et al.,* 2015 is the basis of this research work. This work was first replicated and the results obtained which has been presented in the legend labelled "*without node cooperation*" throughout this chapter is used as a benchmark for the proposed model.

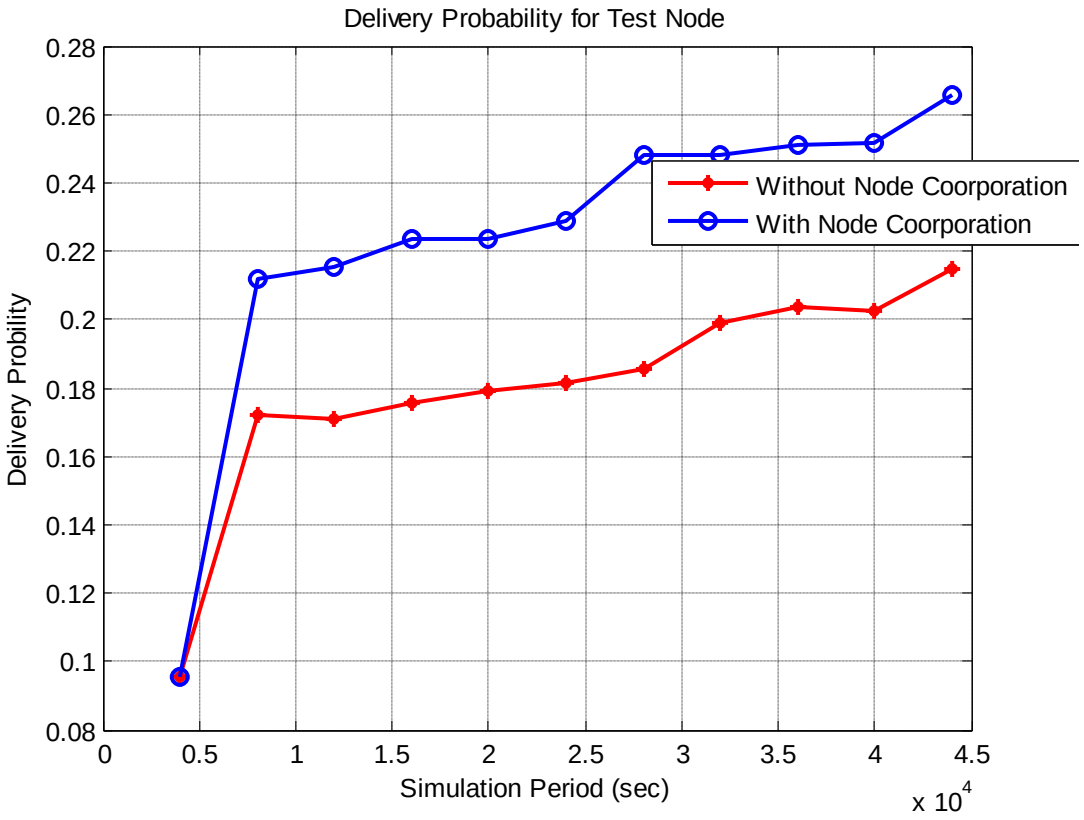

Fig. 4.1: Delivery Probability of for Test Nodes

From Fig. 4.1, it can be observed that the response in blue represents the delivery probability obtained for the proposed model on the test nodes. This fig. shows that the performance of both model increased as the nodes stay longer in the network. However, the proposed model performed much better as compared with the model without node cooperation.

The result obtained for the buffer time management for the test node using the conventional security aided and group encounter PRoPHETrouting protocol is given in Fig. 4.2.

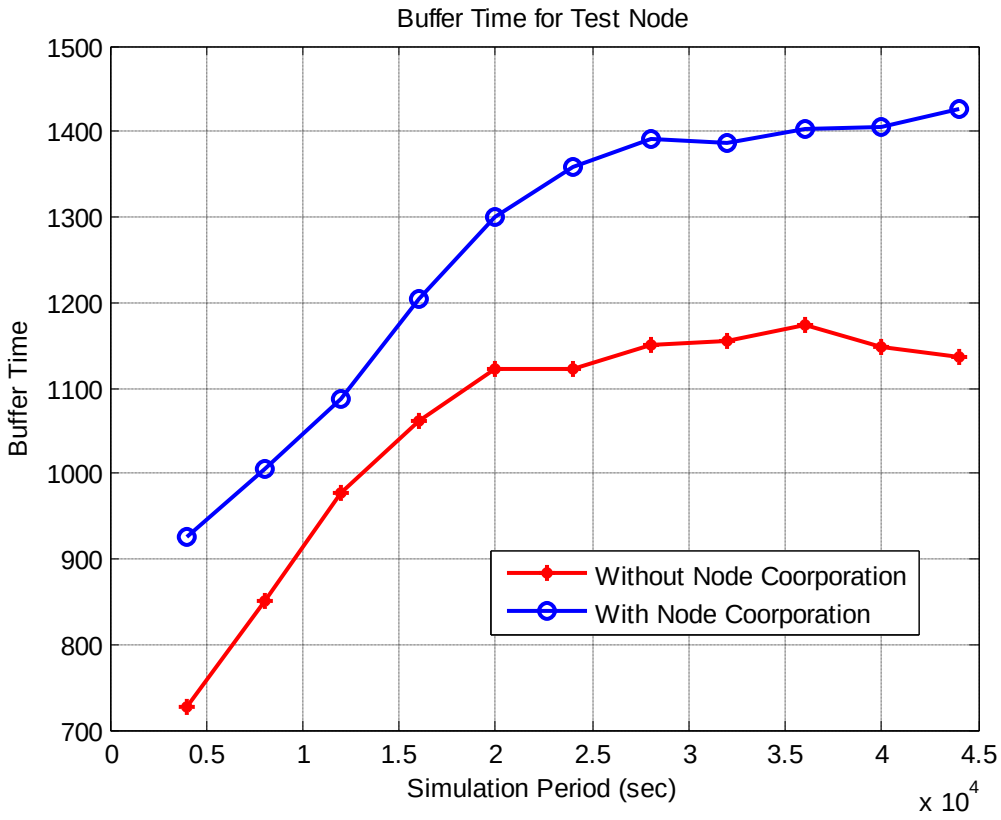

Fig. 4.2: Buffer Time for Test Node

From Fig. 4.2, it can be observed that the proposed model with node cooperation has much higher buffer time when compared with the model without node cooperation. This is expected, since every node in the proposed model has mutual behaviours by helping to distribute or pass message towards neighbouring nodes through cooperative behaviour.

The result obtained for the hop count for the test node using the conventional security aided and group encounter PRoPHETrouting protocol is given in Fig. 4.3.

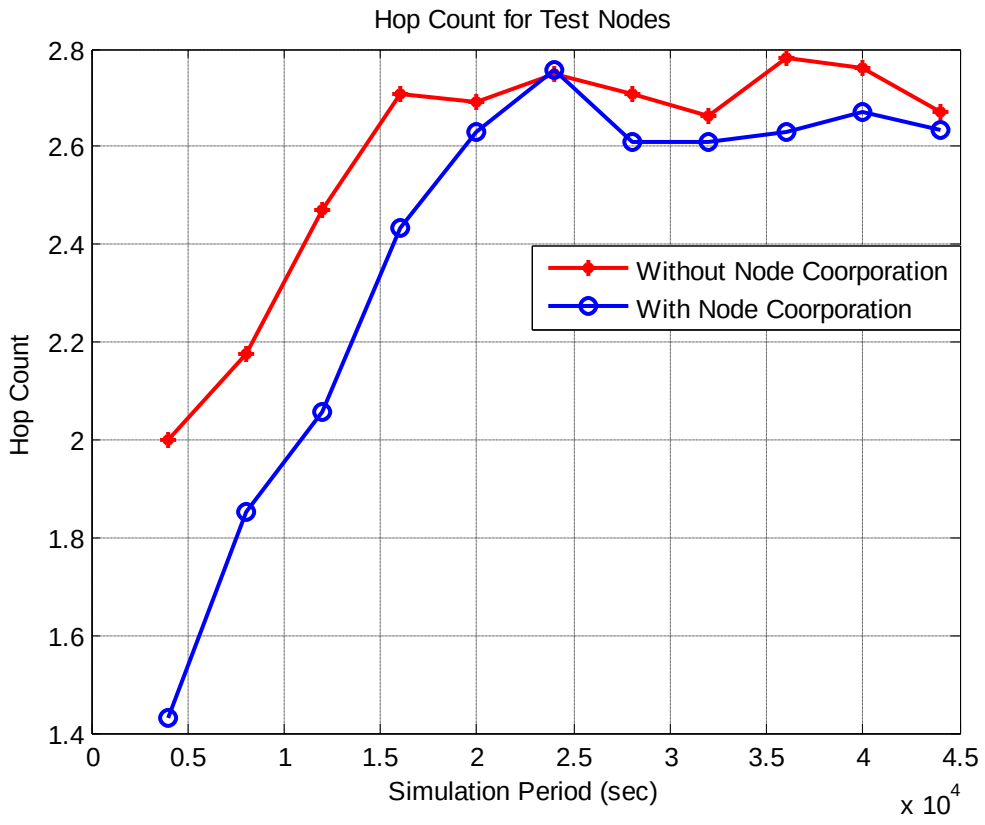

Fig. 4.3: Hop Count for Test Node

It can be observed from fig. 4.3 that, the security aided and group encounter PRoPHETrouting with node cooperation has a less hop count, since store and forward and other latencies are incurred through each hop, a large number of hops between source and destination implies lower real-time performance.

4.3 Performance Evaluation for Helsinki

In this section the performance of the model on the benchmark Helsinki simulation area are presented. Simulation was also performed for a period of 44000 seconds in an interval of 4000 seconds. Detail results obtained for both the proposed model with node cooperation and the standard model without node cooperation can be found in Appendix B3 and Appendix B4 respectively.

Fig. 4.4 shows the superimposed results obtained for the delivery probability of the proposed with node cooperation and standard without node cooperation.

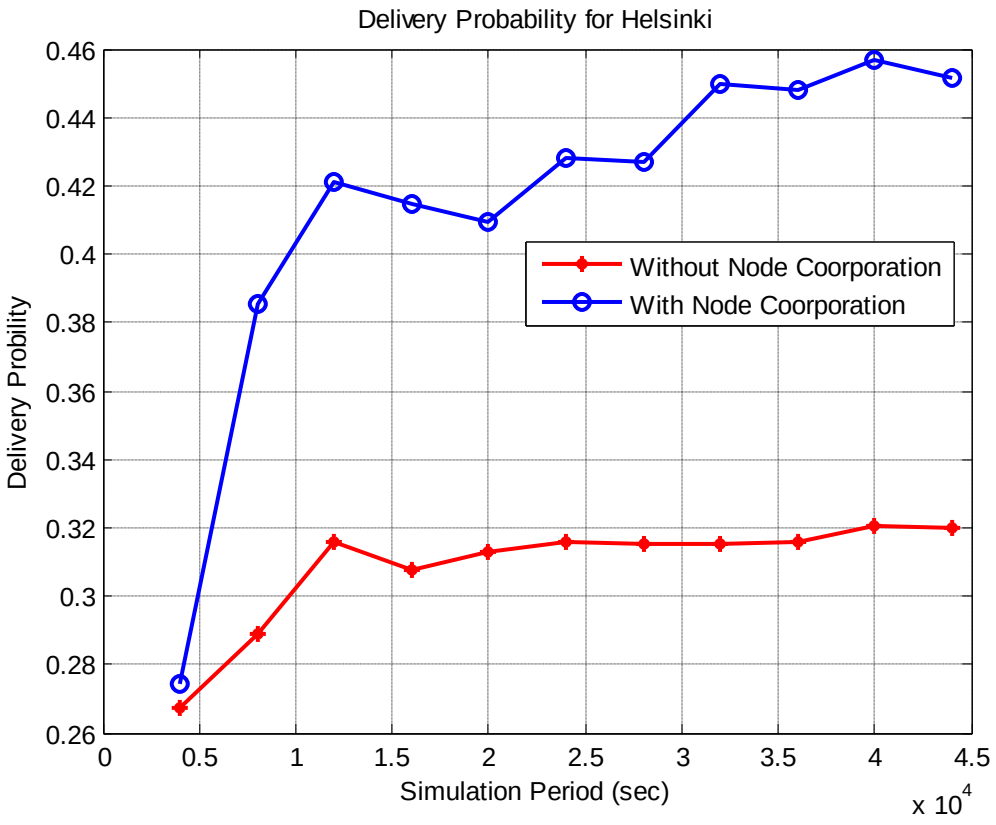

Fig. 4.4: Delivery Probability on Helsinki

Fig. 4.5 shows the graphical representation of improved security aided and group encounter prophet routing protocol as compared with the conventional security aided and group encounter prophet routing protocol for the benchmark Helsinki simulation area. From the response presented in Fig. 4.4, it can be observed that, the proposed model attained a higher delivery probability throughout, compared with the standard model without node cooperation.

.

The result obtained for the buffer time management for Helsinki region using the proposed node cooperation based security aided and group encounter PRoPHETrouting protocol is given in Fig. 4.5.

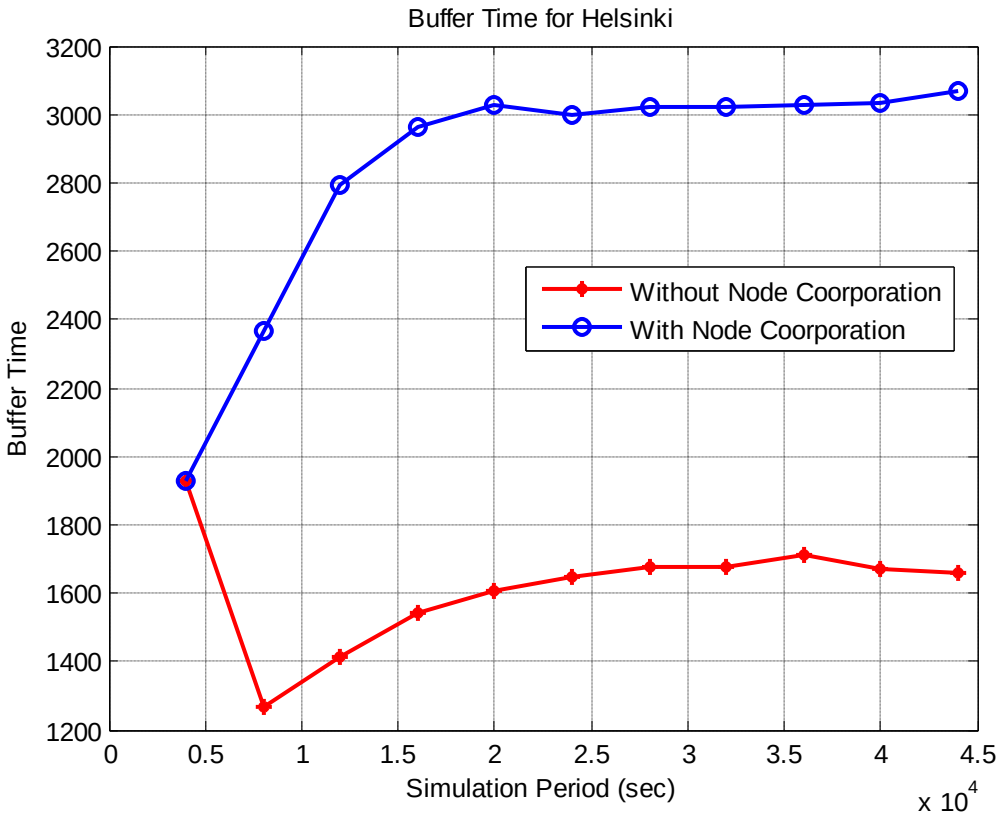

Fig. 4.5 Buffer Time for Helsinki Region

From, Fig. 4.5, it can be observed that the proposed model with node cooperation has a much higher buffer time when compared with the model without node cooperation. This is also expected since every node in the proposed model has mutual behaviours by helping to distribute or pass message towards neighbouring nodes even if the message is not meant for it.

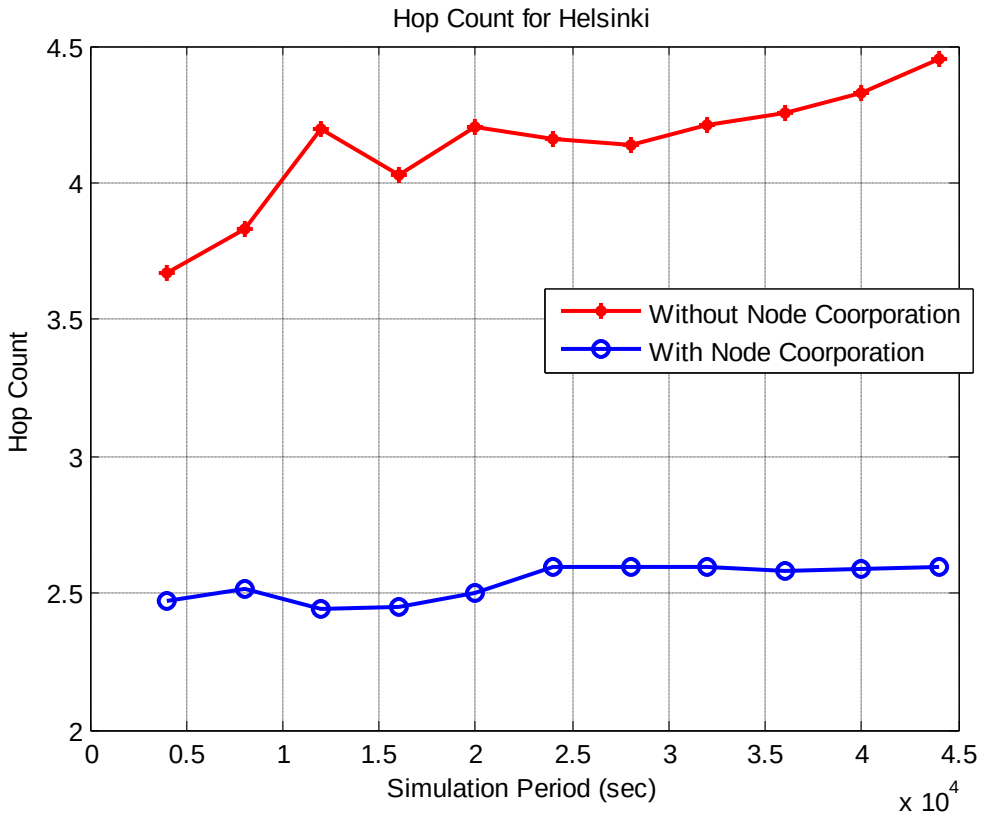

Fig. 4.6: Hop Count for Helsinki Benchmark Scenario

Similar, it can be observed that, the security aided and group encounter based PRoPHET routing with node cooperation has a less hop count when compared with the model without node cooperation. This is also expected since store and forward and other latencies are incurred through each hop, a large number of hops between source and destination implies lower real-time performance.

The matlab program used in plotting the metrics (delivery probability, latency, buffer time, hop count) can be found in appendix C1.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Introduction

This chapter presents the summary of findings; conclusion and limitations encountered in the course of this research work, and also possible area for future research work are discussed.

5.2 Summary of Findings

A lot of research works have been conducted on the area of opportunistic network using ONE simulator and validating with Helsinki benchmark region. Thus, this research work has proposed the development of a node cooperation based group encounter for PRoPHETrouting protocol with the following findings.

(i) The proposed node cooperation based model performed better with a delivery probability improvement of about 19.60% over the model proposed in Basu *et al.*,2015 when implanted on the test node and 25.66% when implemented on the benchmark mark Helsinki scenario..

(ii) Also, a reduction of about -7.85% in hop count was obtained when the proposed model was implemented on the test nodes and -62.86% when the model was implemented on Helsinki scenario.

(iii) Finally, buffer time improvement of about 34.65% was obtained when the proposed model was implemented on the test nodes and 55.54% when implemented on the Helsinki region.

5.3 Conclusions

The improved security aided and group encounter PRoPHETrouting protocol was developed by incorporating node cooperation technique in to the conventional security aided and group encounter PRoPHETrouting protocol. The improved security aided and group encounter PRoPHETrouting protocol was simulated and compared with the conventional security aided and group encounter PRoPHETrouting protocol. The results showed that, there is an improvement in the security aided and group encounter PRoPHETrouting protocol because, it improved the

delivery probability, reduced latency, improved buffer time and reduced hop count. The results for the test node at the end of the simulation time showed that, the delivery probability is improved by 19.6%, improved the buffer time by 34.7% and reduced the hop count by 7.9%. Their delivery probability, buffer time and hop count are 0.223855, 1617.475, 2.392155 for the node cooperation.

The results for delivery probability, buffer time and hop count without node cooperation are 0.180018,1056.974, 2.580055.

While the results for the Helsinki at the end of the simulation time showed that, the delivery probability is improved by 25.7%, improved the buffer time by 55.5 % and reduced the hop count by 62.9%. Their delivery probability, buffer time and hop count are0.415182, 2840.776, 2.538582for the node cooperation. The results for delivery probability, buffer time and hop count without node cooperation are 0.308664, 1263.024, 4.134218.

5.4 Significant Contributions

The significant contributions of this research work are as follows:

(i) Development of an Improved Security Aided Group Encounter PRoPHETRouting Protocol of an Opportunistic Network with node cooperation based technique.

(ii) The improved SAGE-PRoPHETachieved improvements of 25.7%, 62.9%, 55.5% with respect to delivery probability, hop count, and buffer time, respectively for the Helsinki program over the method proposed in Basu et al, 2015.

(iii) The improved SAGE-PRoPHETalso achieved improvements of 19.6%, 7.9%, 34.7% with respect to delivery probability, hop count, and buffer time,respectively for the 20-node program over the method proposed in Basu et al, 2015.

5.4 .1 Limitations

The limitation of this research work is itemized as follows:

(i) The same storage space was assumed for all devices in this network.

(ii) We experienced latency in the network even though the processing time was improved and malicious dropping of packets was prevented.

5.5 Recommendations

The recommendations for this research work are as follows:

(i) The vehicular mobility model can be used to evaluate the security performance of the security aided and groups encounter PRoPHETrouting protocol.

(ii) Other techniques like incentives can be integrated in to the operation of OppNets to enhance node cooperation.

REFERENCE

Asplund, M., Nadjm-Tehrani, S., & Sigholm, J. (2008). Emerging information infrastructures: Cooperation in disasters. Paper presented at the *International Workshop on Critical Information Infrastructures Security*. 258-270

Basu, S., Bhattacharjee, S., Roy, S., & Bandyopadhyay, S. (2015). SAGE-PRoPHET: A Security Aided and Group Encounter based PRoPHET Routing Protocol for Dissemination of Post Disaster Situational Data. Paper presented at the *Proceedings of the 2015 International Conference on Distributed Computing and Networking*. 10-20

Chang, M., Chen, R., Bao, F., & Cho, J.-H. (2011). Trust-Threshold Based Routing in Delay Tolerant Networks. Paper presented at the *IFIP International Conference on Trust Management*. 265-276

Chen, R., Bao, F., Chang, M., & Cho, J.-H. (2012). Integrated social and QoS trust-based routing in delay tolerant networks. *Wireless Personal Communications, 66*(2), 443-459.

Chen, R., Bao, F., Chang, M., & Cho, J.-H. (2014). Dynamic trust management for delay tolerant networks and its application to secure routing. *IEEE Transactions on Parallel and Distributed Systems, 25*(5), 1200-1210.

Ciobanu, R. I., Dobre, C., Cristea, V., Pop, F., & Xhafa, F. (2015). SPRINT-SELF: Social-Based Routing and Selfish Node Detection in Opportunistic Networks. *Mobile Information Systems, 2015*.

Dinakar, S., Bhavadharini, R., & Karthik, S. (2012). Study of Opportunistic Network and MANET. *International Journal of Software and Hardware Research in Engineering, 1*(4), 97-100.

Fall, K., Scott, K. L., Burleigh, S. C., Torgerson, L., Hooke, A. J., Weiss, H. S., . . . Cerf, V. (2007). Delay-tolerant networking architecture.

Grasic, S., Davies, E., Lindgren, A., & Doria, A. (2011). The evolution of a DTN routing protocol-PRoPHETv2. Paper presented at the *Proceedings of the 6th ACM workshop on Challenged networks*. 27-30

Guo, M.-H., Liaw, H.-T., Chiu, M.-Y., & Tsai, L.-P. (2015). Authenticating with privacy protection in opportunistic networks. Paper presented at the *Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE), 2015 11th International Conference on*. 375-380

Gupta, S., Dhurandher, S. K., Woungang, I., Kumar, A., & Obaidat, M. S. (2013). Trust-based Security Protocol against blackhole attacks in opportunistic networks. Paper presented at the *WiMob*. 724-729

Huang, C.-M., Lan, K.-c., & Tsai, C.-Z. (2008). A survey of opportunistic networks. Paper presented at the *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on Advanced Information Networking.* 1672-1677

Kaur, E. U., & Kaur, E. H. (2009). Routing techniques for opportunistic networks and Security Issues. *National Conference on Computing, communication and control*.

Li, N., & Das, S. K. (2010). RADON: reputation-assisted data forwarding in opportunistic networks. Paper presented at the *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*. 8-14

Li, N., & Das, S. K. (2013). A trust-based framework for data forwarding in opportunistic networks. *ELsivier journal for Ad Hoc Networks, 11*(4), 1497-1509. doi: 10.1016/j.adhoc.2011.01.018

Lindgren, A., Doria, A., & Schelén, O. (2003a). Poster: Probabilistic routing in intermittently connected networks. Paper presented at the *Proceedings of The Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003)*. 90-100

Lindgren, A., Doria, A., & Schelén, O. (2003b). Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE mobile computing and communications review, 7*(3), 19-20.

Luo, H., Kravets, R., & Abdelzaher, T. (2006). The-day-after networks: A first-response edge-network architecture for disaster relief. *NSF NeTS FIND Initiative, http://www. nets-find. net/Funded/DayAfterNet. php*.

Mehrotra, S., Butts, C., Kalashnikov, D., Venkatasubramanian, N., Rao, R. R., Chockalingam, G., . . . Huyck, C. (2004). Project RESCUE: challenges in responding to the unexpected. Paper presented at the *Electronic Imaging 2004*. 179-192

Orozco, J., Santos, R., Ochoa, S. F., & Meseguer, R. Stochastic Performance Evaluation of Routing Strategies in Opportunistic Networks.

Papaj, J., Dobos, L. u., & Cizmár, A. (2012). Opportunistic Networks and Security. *Journal of Electrical and Electronics Engineering, 5*(1), 163.

Pelusi, L., Passarella, A., & Conti, M. (2006a). Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *IEEE Communications Magazine, 44*(11), 134-141.

Pelusi, L., Passarella, A., & Conti, M. (2006b). Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *Communications Magazine, IEEE, 44*(11), 134-141.

Poonguzharselvi, B., & Vetriselvi, V. (2012). Data forwarding in Opportunistic Network Using mobile traces. Paper presented at the *International Conference on Information Technology Convergence and Services*. 425-430

Shikfa, A., Önen, M., & Molva, R. (2010). Bootstrapping security associations in opportunistic networks. Paper presented at the *8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010*. 147-152

Trifunovic, S., Legendre, F., & Anastasiades, C. (2010). Social trust in opportunistic networks. Paper presented at the *INFOCOM IEEE Conference on Computer Communications Workshops, 2010*. 1-6

Uddin, M. Y. S., Nicol, D. M., Abdelzaher, T. F., & Kravets, R. H. (2009). A post-disaster mobility model for delay tolerant networking. Paper presented at the *Winter Simulation Conference*. 2785-2796

Verma, A., & Srivastava, D. (2012). Integrated routing protocol for opportunistic networks. *arXiv preprint arXiv:1204.1658*.

Vinck, P. (2013). *World disasters report: Focus on technology and the future of humanitarian action*: International Federation of Red Cross and Red Crescent Societies.

Wu, Y., Zhao, Y., Riguidel, M., Wang, G., & Yi, P. (2015). Security and trust management in opportunistic networks: a survey. *Security and Communication Networks, 8*(9), 1812-1827.

Yahaya, B. (2015). *Development of an Improved Integrated Routing Protocol for Opportunistic Network with Congestion Control Using Pre-Emptive Data Eviction.* (Masters Degree), Ahamdu Bello University, Samaru, Zaria.

Yogi, M., & Chinthala, V. (2014). A Study of Opportunistic Networks for Efficient Ubiquitous Computing. *International Journal of Advanced Research in Computing and Communication Engineering, 3*(1), 5187-5191.

http://www.oracle.com/java/javase/downloads/index.html

http://www.eclipse.org/downloads/

http://www.netlab.tkk.fi/tutkimus/dtn/theone/down/one_1.5.1-RC

Appendix A1

The program class for PRoPHETrouting on the test node

```
#
# Default settings for the simulation
#

## Scenario settings
Scenario.name = HUSSEINAH_TEST
Scenario.simulateConnections = true
Scenario.updateInterval = 1.0

Scenario.endTime = 44000
Scenario.nrofHostGroups = 1



## Interface-specific settings:
# type : which interface class the interface belongs to
# For different types, the sub-parameters are interface-specific
# For SimpleBroadcastInterface, the parameters are:
# transmitSpeed : transmit speed of the interface (bytes per second)
# transmitRange : range of the interface (meters)

# "Bluetooth" interface for all nodes
btInterface.type = SimpleBroadcastInterface
# Transmit speed of 2 Mbps = 250kBps
btInterface.transmitSpeed = 250k
btInterface.transmitRange = 10

highspeedInterface.type = SimpleBroadcastInterface
highspeedInterface.transmitSpeed = 10M
highspeedInterface.transmitRange = 1000
```

```
## Group-specific settings:
# groupID : Group's identifier. Used as the prefix of host names
# nrofHosts: number of hosts in the group
# movementModel: movement model of the hosts (valid class name from movement package)
# waitTime: minimum and maximum wait times (seconds) after reaching destination
# speed: minimum and maximum speeds (m/s) when moving on a path
# bufferSize: size of the message buffer (bytes)
# router: router used to route messages (valid class name from routing package)
# activeTimes: Time intervals when the nodes in the group are active (start1, end1, start2, end2,
...)
# msgTtl : TTL (minutes) of the messages created by this host group, default=infinite

## Group and movement model specific settings
# pois: Points Of Interest indexes and probabilities (poiIndex1, poiProb1, poiIndex2, poiProb2,
... )
#       for ShortestPathMapBasedMovement
# okMaps : which map nodes are OK for the group (map file indexes), default=all
#          for all MapBasedMovent models
# routeFile: route's file path - for MapRouteMovement
# routeType: route's type - for MapRouteMovement

# Common settings for all groups
Group.movementModel = RandomWalk
Group.router = ProphetRouter
ProphetRouter.secondsInTimeUnit=1
EpidemicRouter.binaryMode =true
EpedemicRouter.nrofCopies = 8
Group.bufferSize = 5M
```

```
# All nodes have the same bluetooth interface
Group.nrofInterfaces = 1
Group.interface1 = btInterface
# Walking speeds
Group.speed = 0.5, 1.5
# Message TTL of 300 minutes (5 hours)
Group.msgTtl = 300
Group.nrofHosts = 20
Group.groupID = n

## Movement model settings
# seed for movement models' pseudo random number generator (default = 0)
MovementModel.rngSeed = 1
# World's size for Movement Models without implicit size (width, height; meters)
MovementModel.worldSize = 450, 340
# How long time to move hosts in the world before real simulation

## Message creation parameters
# How many event generators
Events.nrof = 1
# Class of the first event generator
Events1.class = MessageEventGenerator
# (following settings are specific for the MessageEventGenerator class)
# Creation interval in seconds (one new message every 25 to 35 seconds)
Events1.interval = 25,35
# Message sizes (500kB - 1MB)
Events1.size = 500k,1M
# range of message source/destination addresses
Events1.hosts = 0,4
# Message ID prefix
Events1.prefix = M
```

```
## Reports - all report names have to be valid report classes
# how many reports to load
Report.nrofReports = 1
Report.reportDir = reports/HUSSEINA
# Report classes to load
Report.report1 = MessageStatsReport

## GUI settings
# GUI underlay image settings
GUI.UnderlayImage.fileName = data/helsinki_underlay.png
# Image offset in pixels (x, y)
GUI.UnderlayImage.offset = 34, 20
# Scaling factor for the image
GUI.UnderlayImage.scale = 4.75
# Image rotation (radians)
GUI.UnderlayImage.rotate = -0.015

# how many events to show in the log panel (default = 30)
GUI.EventLogPanel.nrofEvents = 100
# Regular Expression log filter (see Pattern-class from the Java API for RE-matching details)
#GUI.EventLogPanel.REfilter = .*p[1-9]<->p[1-9]$

## Optimization settings -- these affect the speed of the simulation
## see World class for details.
Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder = true
```

Appendix A2

The program for PRoPHETrouting node cooperation based for the test

node

Java program for PRoPHETRouting

```
package routing;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import routing.util.RoutingInfo;

import util.Tuple;

import core.Connection;
import core.DTNHost;
import core.Message;
import core.Settings;
import core.SimClock;

/**
 * Implementation of PRoPHET router as described in
 * <I>Probabilistic routing in intermittently connected networks</I> by
 * Anders Lindgren et al.
 */
```

```
public class ProphetRouter extends ActiveRouter {
	/** delivery predictability initialization constant*/
	public static final double P_INIT = 0.75;
	/** delivery predictability transitivity scaling constant default value */
	public static final double DEFAULT_BETA = 0.25;
	/** delivery predictability aging constant */
	public static final double GAMMA = 0.98;

	/** Prophet router's setting namespace ({@value})*/
	public static final String PROPHET_NS = "ProphetRouter";
	/**
	 * Number of seconds in time unit -setting id ({@value}).
	 * How many seconds one time unit is when calculating aging of
	 * delivery predictions. Should be tweaked for the scenario.*/
	public static final String SECONDS_IN_UNIT_S ="secondsInTimeUnit";

	/**
	 * Transitivity scaling constant (beta) -setting id ({@value}).
	 * Default value for setting is {@link #DEFAULT_BETA}.
	 */
	public static final String BETA_S = "beta";

	/** the value of nrof seconds in time unit -setting */
	private int secondsInTimeUnit;
	/** value of beta setting */
	private double beta;

	/** delivery predictabilities */
	private Map<DTNHost, Double> preds;
	/** last delivery predictability update (sim)time */
	private double lastAgeUpdate;
```

```
/**
 * Constructor. Creates a new message router based on the settings in
 * the given Settings object.
 * @param s The settings object
 */
public ProphetRouter(Settings s) {
	super(s);
	Settings prophetSettings = new Settings(PROPHET_NS);
	secondsInTimeUnit = prophetSettings.getInt(SECONDS_IN_UNIT_S);
	if (prophetSettings.contains(BETA_S)) {
		beta = prophetSettings.getDouble(BETA_S);
	}
	else {
		beta = DEFAULT_BETA;
	}

	initPreds();
}

/**
 * Copyconstructor.
 * @param r The router prototype where setting values are copied from
 */
protected ProphetRouter(ProphetRouter r) {
	super(r);
	this.secondsInTimeUnit = r.secondsInTimeUnit;
	this.beta = r.beta;
	initPreds();
}
```

```
/**
 * Initializes predictability hash
 */
private void initPreds() {
	this.preds = new HashMap<DTNHost, Double>();
}

@Override
public void changedConnection(Connection con) {
	super.changedConnection(con);

	if (con.isUp()) {
		DTNHost otherHost = con.getOtherNode(getHost());
		updateDeliveryPredFor(otherHost);
		updateTransitivePreds(otherHost);
	}
}

/**
 * Updates delivery predictions for a host.
 * <CODE>P(a,b) = P(a,b)_old + (1 - P(a,b)_old) * P_INIT</CODE>
 * @param host The host we just met
 */
private void updateDeliveryPredFor(DTNHost host) {
	double oldValue = getPredFor(host);
	double newValue = oldValue + (1 - oldValue) * P_INIT;
	preds.put(host, newValue);
}

/**
 * Returns the current prediction (P) value for a host or 0 if entry for
```

```
 * the host doesn't exist.
 * @param host The host to look the P for
 * @return the current P value
 */
public double getPredFor(DTNHost host) {
	ageDeliveryPreds(); // make sure preds are updated before getting
	if (preds.containsKey(host)) {
		return preds.get(host);
	}
	else {
		return 0;
	}
}

/**
 * Updates transitive (A->B->C) delivery predictions.
 * <CODE>P(a,c) = P(a,c)_old + (1 - P(a,c)_old) * P(a,b) * P(b,c) * BETA
 * </CODE>
 * @param host The B host who we just met
 */
private void updateTransitivePreds(DTNHost host) {
	MessageRouter otherRouter = host.getRouter();
	assert otherRouter instanceof ProphetRouter : "PRoPHET only works " +
		" with other routers of same type";

	double pForHost = getPredFor(host); // P(a,b)
	Map<DTNHost, Double> othersPreds =
		((ProphetRouter)otherRouter).getDeliveryPreds();

	for (Map.Entry<DTNHost, Double> e : othersPreds.entrySet()) {
		if (e.getKey() == getHost()) {
```

```
                continue; // don't add yourself
            }

            double pOld = getPredFor(e.getKey()); // P(a,c)_old
            double pNew = pOld + ( 1 - pOld) * pForHost * e.getValue() * beta;
            preds.put(e.getKey(), pNew);
        }
    }

    /**
     * Ages all entries in the delivery predictions.
     * <CODE>P(a,b) = P(a,b)_old * (GAMMA ^ k)</CODE>, where k is number of
     * time units that have elapsed since the last time the metric was aged.
     * @see #SECONDS_IN_UNIT_S
     */
    private void ageDeliveryPreds() {
        double timeDiff = (SimClock.getTime() - this.lastAgeUpdate) /
            secondsInTimeUnit;

        if (timeDiff == 0) {
            return;
        }

        double mult = Math.pow(GAMMA, timeDiff);
        for (Map.Entry<DTNHost, Double> e : preds.entrySet()) {
            e.setValue(e.getValue()*mult);
        }

        this.lastAgeUpdate = SimClock.getTime();
    }
```

```
/**
 * Returns a map of this router's delivery predictions
 * @return a map of this router's delivery predictions
 */
private Map<DTNHost, Double> getDeliveryPreds() {
	ageDeliveryPreds(); // make sure the aging is done
	return this.preds;
}

@Override
public void update() {
	super.update();
	if (!canStartTransfer() ||isTransferring()) {
		return; // nothing to transfer or is currently transferring
	}

	// try messages that could be delivered to final recipient
	if (exchangeDeliverableMessages() != null) {
		return;
	}

	tryOtherMessages();
}

/**
 * Tries to send all other messages to all connected hosts ordered by
 * their delivery probability
 * @return The return value of {@link #tryMessagesForConnected(List)}
 */
private Tuple<Message, Connection> tryOtherMessages() {
	List<Tuple<Message, Connection>> messages =
```

```
        new ArrayList<Tuple<Message, Connection>>();

Collection<Message> msgCollection = getMessageCollection();

/* for all connected hosts collect all messages that have a higher
   probability of delivery by the other host */
for (Connection con : getConnections()) {
        DTNHost other = con.getOtherNode(getHost());
        ProphetRouter othRouter = (ProphetRouter)other.getRouter();

        if (othRouter.isTransferring()) {
                continue; // skip hosts that are transferring
        }

        for (Message m : msgCollection) {
                if (othRouter.hasMessage(m.getId())) {
                        continue; // skip messages that the other one has
                }
                if (othRouter.getPredFor(m.getTo()) > getPredFor(m.getTo())) {
                        // the other node has higher probability of delivery
                        messages.add(new Tuple<Message, Connection>(m,con));
                }
        }
}

if (messages.size() == 0) {
        return null;
}

// sort the message-connection tuples
Collections.sort(messages, new TupleComparator());
```

```
        return tryMessagesForConnected(messages);        // try to send messages
}

/**
 * Comparator for Message-Connection-Tuples that orders the tuples by
 * their delivery probability by the host on the other side of the
 * connection (GRTRMax)
 */
private class TupleComparator implements Comparator
        <Tuple<Message, Connection>> {

        public int compare(Tuple<Message, Connection> tuple1,
                        Tuple<Message, Connection> tuple2) {
                // delivery probability of tuple1's message with tuple1's connection
                double p1 = ((ProphetRouter)tuple1.getValue().
                                getOtherNode(getHost()).getRouter()).getPredFor(
                                tuple1.getKey().getTo());
                // -"- tuple2...
                double p2 = ((ProphetRouter)tuple2.getValue().
                                getOtherNode(getHost()).getRouter()).getPredFor(
                                tuple2.getKey().getTo());

                // bigger probability should come first
                if (p2-p1 == 0) {
                        /* equal probabilities -> let queue mode decide */
                        return compareByQueueMode(tuple1.getKey(), tuple2.getKey());
                }
                else if (p2-p1 < 0) {
                        return -1;
                }
                else {
```

```
                        return 1;
                    }
                }
            }

            @Override
            public RoutingInfo getRoutingInfo() {
                ageDeliveryPreds();
                RoutingInfo top = super.getRoutingInfo();
                RoutingInfo ri = new RoutingInfo(preds.size() +
                        " delivery prediction(s)");

                for (Map.Entry<DTNHost, Double> e : preds.entrySet()) {
                    DTNHost host = e.getKey();
                    Double value = e.getValue();

                    ri.addMoreInfo(new RoutingInfo(String.format("%s : %.6f",
                            host, value)));
                }

                top.addMoreInfo(ri);
                return top;
            }

            @Override
            public MessageRouter replicate() {
                ProphetRouter r = new ProphetRouter(this);
                return r;
            }

}
```

Appendix B1

Detail Results Obtained for the Test Node without node cooperation

SIM T	TIME SEC	DP	Buffer time	Hop count	
4000		0.0953	726.1947	2.0000	
8000	212.39	0.1719	851.5692	2.1739	
12000	272.23	0.1711	976.6067	2.4706	
16000	109.89	0.1759	1062.5296	2.7083	
20000	251.25	0.1792	1122.1521	2.6935	
24000	226.27	0.1813	1123.1904	2.7500	
28000	192.40	0.1858	1151.2500	2.7097	
32000	220.17	0.1992	1156.0307	2.6612	
36000		0.2034	1174.1293	2.7801	
40000	209.69	0.2023	1147.4536	2.7613	
44000	212.89	0.2148	1135.6108	2.6720	

Appendix B2

Detail Result Obtained for the Test Node with node cooperation

SIM T	TIME SEC	DP	Buffer time	Hop count	
4000	119.70	0.0953	1925.3824	1.4321	
8000	161.21	0.2121	1264.6158	1.8529	
12000	230.84	0.2151	1411.5372	2.0577	
16000		0.2235	1541.6156	2.4324	
20000	255.87	0.2233	1605.6142	2.6304	
24000	222.39	0.2291	1647.2758	2.7565	
28000	247.25	0.2479	1678.4262	2.6092	
32000		0.2479	1678.4262	2.6092	
36000	171.83	0.2508	1709.5200	2.6281	
40000	420.82	0.2517	1672.0790	2.6712	
44000	141.93	0.2657	1657.7372	2.6340	

Appendix B3

Detail Result Obtained for Helsinki without node cooperation

SIM T	TIME SEC	DP	Buffer time	Hop count	
4000	47.83	0.2671	926.6589	3.6667	
8000	70.61	0.2887	1004.3477	3.8333	
12000	55.14	0.3159	1086.4936	4.1935	
16000	61.13	0.3076	1204.4092	4.0252	
20000	39.71	0.3132	1300.9418	4.2039	
24000	56.81	0.3158	1357.3559	4.1622	
28000	56.62	0.3150	1390.9951	4.1395	
32000	59.62	0.3151	1386.8011	4.2114	
36000	116.80	0.3161	1403.1972	4.2554	
40000	102.80	0.3208	1406.2814	4.3302	
44000	117.46	0.3200	1425.7774	4.4551	

Appendix B4

Detail Result Obtained for Helsinki with node cooperation

SIM T	TIME SEC	DP	Buffer time	Hop count	
4000	119.70	0.2746	1925.3824	2.4706	
8000	108.23	0.3854	2364.4310	2.5125	
12000	238.14	0.4213	2795.1711	2.4412	
16000	109.79	0.4146	2962.0377	2.4463	
20000		0.4095	3029.6997	2.5023	
24000	117.70	0.4279	2996.9269	2.5942	
28000	114.23	0.4270	3020.8477	2.5981	
32000	102.61	0.4500	3024.1300	2.5944	
36000		0.4481	3030.3842	2.5786	
40000	54.53	0.4568	3032.2307	2.5892	
44000	122.34	0.4518	3067.2979	2.5970	

Appendix C1

The matlab program used to generate the response of the metrics

```
%Test Program Standard
clc
close all
T=[4000,8000,12000,16000,20000,24000,28000,32000,36000,40000,44000];
Del%THE REPLICATED PROGRAME ON TEST BED AND HELSINKI
_Prob=[0.0953,0.1719,0.1711,0.1759,0.1792,0.1813,0.1858,0.1992,...
   0.2034,0.2023,0.2148];
Btime=[726.1947,851.5692,976.6067,1062.5296,1122.1521,1123.1904,...
   1151.2500,1156.0307,1174.1293,1147.4536,1135.6108];
Hcount=[2.0000,2.1739,2.4706,2.7083,2.6935,2.7500,2.7097,2.6612,...
2.7801,2.7613,2.6720];
% plot(T,Del_Prob,'-*','LineWidth',2)
% xlabel('Simulation Period (sec)')
% ylabel('Delivery Probility')
% grid
%Helsinki Program standard

Test_Hprob=[0.2671,0.2887,0.3159,0.3076,0.3132,0.3158,0.3150,0.3151,...
0.3161,0.3208,0.3200];
BtimeH=[926.6589,1004.3477,1086.4936,1204.4092,1300.9418,1357.3559,...
   1390.9951,1386.8011,1403.1972,1406.2814,1425.7774];
HcountH=[3.6667,3.8333,4.1935,4.0252,4.2039,4.1622,4.1395,4.2114,...
4.2554,4.3302,4.4551];
% plot(T,Test_Hprob,'-r*','LineWidth',2)
% xlabel('Simulation Period (sec)')
% ylabel('Delivery Probility')
% grid

%NODE COORPERATION TECHNIQUE
```

```
%TEST BED
ntDel_Pro=[0.0953,0.2121,0.2151,0.2235,0.2233,0.2291,0.2479,0.2479...
    0.2508,0.2517,0.2657];
ntLat=[1925.3824,1580.9412,2048.9615,2654.3378,2894.8261,3195.3478,...
3135.8678,3135.8678,3233.1106,3249.8739,3139.9057];
ntBtime=[1925.3824,1264.6158,1411.5372,1541.6156,1605.6142,1647.2758,...
1678.4262,1678.4262,1709.5200,1672.0790,1657.7372];
ntHcout=[1.4321,1.8529,2.0577,2.4324,2.6304,2.7565,2.6092,2.6092,...
2.6281,2.6712,2.6340];

%HELSINKI
HDel_Probnt=[0.2746,0.3854,0.4213,0.4146,0.4095,0.4279,0.4270,0.4500,...
0.4481,0.4568,0.4518];
ntLatH=[1925.3824,2831.0300,3588.8757,4085.4073,4273.5032,4532.7804,...
4799.5287,5104.6622,5245.6825,5272.9118,5396.9921];
ntBtimeH=[1925.3824,2364.4310,2795.1711,2962.0377,3029.6997,2996.9269,...
3020.8477,3024.1300,3030.3842,3032.2307,3067.2979];
ntHcoutH=[2.4706,2.5125,2.4412,2.4463,2.5023,2.5942,2.5981,2.5944,...
2.5786,2.5892,2.5970];
%%%%%%%%%%%%%%%%    EVALUATING THE
%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%    DELIVERY PROBABILITY %%%%%%%%%%%%%
%Delivery Probability Evaluation Based on Test Bed
figure(1)
plot(T,Del_Prob,'-*r','LineWidth',2)
hold on
plot(T,ntDel_Pro,'-o','LineWidth',2)
title('Delivery Probability for Test Node')
legend('Without Node Coorporation','With Node Coorporation')
xlabel('Simulation Period (sec)')
ylabel('Delivery Probility')
```

```
grid
%Delivery Probability for Helsinki on Test Bed
figure(2)
plot(T,Test_Hprob,'-*r','LineWidth',2)
hold on
plot(T,HDel_Probnt,'-o','LineWidth',2)
title('Delivery Probability for Helsinki')
legend('Without Node Coorporation','With Node Coorporation')
xlabel('Simulation Period (sec)')
ylabel('Delivery Probility')
grid
%%%%%%%%%%%%%%%%    EVALUATING THE  %%%%%%%%%%%%
%%%%%%%%%%%%%%%%    LATENCY AVERAGE  %%%%%%%%%%%%
%Latency Average for Test System
figure(3)
plot(T,Lat,'-*r','LineWidth',2)
hold on
plot(T,ntLat,'-o','LineWidth',2)
title('Latency Average for Test Node')
legend('Without Node Coorporation','With Node Coorporation')
xlabel('Simulation Period (sec)')
ylabel('Latency')
grid
%Latency Average for Helsinki
figure(4)
plot(T,LatH,'-*r','LineWidth',2)
hold on
plot(T,ntLatH,'-o','LineWidth',2)
title('Latency Average for Helsinki')
legend('Without Node Coorporation','With Node Coorporation')
xlabel('Simulation Period (sec)')
```

```
ylabel('Latency')
grid
%%%%%%%%%%%%%%%%    EVALUATING THE  %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%    HOP COUNT       %%%%%%%%%%%%
%Hop count for test node
figure(5)
plot(T,Hcount,'-*r','LineWidth',2)
hold on
plot(T,ntHcout,'-o','LineWidth',2)
title('Hop Count for Test Nodes')
legend('Without Node Coorporation','With Node Coorporation')
xlabel('Simulation Period (sec)')
ylabel('Hop Count')
grid
%Hop count for Helsinki
figure(6)
plot(T,HcountH,'-*r','LineWidth',2)
hold on
plot(T,ntHcoutH,'-o','LineWidth',2)
title('Hop Count for Helsinki')
legend('Without Node Coorporation','With Node Coorporation')
xlabel('Simulation Period (sec)')
ylabel('Hop Count')
grid

%%%%%%%%%%%%%%%%    EVALUATING THE  %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%    BUFFER TIME     %%%%%%%%%%%%
%Hop count for test node


figure(7)
```

```
plot(T,Btime,'-*r','LineWidth',2)
hold on
plot(T,BtimeH,'-o','LineWidth',2)
title('Buffer Time for Test Node')
legend('Without Node Coorporation','With Node Coorporation')
xlabel('Simulation Period (sec)')
ylabel('Buffer Time')
grid
%Hop count for Helsinki
figure(8)
plot(T,ntBtime,'-*r','LineWidth',2)
hold on
plot(T,ntBtimeH,'-o','LineWidth',2)
title('Buffer Time for Helsinki')
legend('Without Node Coorporation','With Node Coorporation')
xlabel('Simulation Period (sec)')
ylabel('Buffer Time')
grid
```