DEVELOPMENT OF AN ENHANCED TRUST MANAGEMENT SCHEME FOR SECURED ROUTING IN OPPORTUNISTIC NETWORKS

BY

BASIRA YAHAYA

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

May, 2018

DEVELOPMENT OF AN ENHANCED TRUST MANAGEMENT SCHEME FOR SECURED ROUTING IN OPPORTUNISTIC NETWORKS

BY

Basira YAHAYA, B.Eng. (ABU, 2011), MSc. (ABU, 2015)
P15EGCP9002
basiraee@yahoo.com

A THESIS SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES,

AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF DOCTOR OF A PHILOSOPHY (PhD.) DEGREE IN COMPUTER ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING
FACULTY OF ENGINEERING
AHMADU BELLO UNIVERSITY, ZARIA
NIGERIA.

MAY, 2018

DECLARATION

I Basira YAHAYA declare that the work in this thesis entitled **"Development of an Enhanced Trust Management Scheme for Secured Routing in Opportunistic Networks"** has been carried out by me in the Department of Computer Engineering. The information derived from literatures has been duly acknowledged in the text and a list of references provided. No part of this thesis was previously presented for another degree or diploma at this or any other institution.

Basira YAHAYA (Student)	______________________ Signature	_________________ Date

CERTIFICATION

This thesis entitled “DEVELOPMENT OF AN ENHANCED TRUST MANAGEMENT SCHEME FOR SECURED ROUTING IN OPPORTUNISTIC NETWORKS” by Basira YAHAYA meets the regulations governing the award of the Doctor of Philosophy (Ph.D) Degree in Computer Engineering of the Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

Prof. M. B Mu’azu (Chairman, Supervisory Committee)	____________________ Signature	____________________ Date
Dr. E. A. Adedokun (Member, Supervisory Committee)	____________________ Signature	____________________ Date
Dr. I. J. Umoh (Member, Supervisory Committee)	____________________ Signature	____________________ Date
Prof. M. B Mu’azu (Head of Department)	____________________ Signature	____________________ Date
Prof. S. Z. Abubakar (Dean School of Postgraduate Studies)	____________________ Signature	____________________ Date

DEDICATION

This research work is dedicated to Almighty Allah (S.W.T) The ultimate Sustainer.

ACKNOWLEDGEMENT

All praises are due to Allah (SWT), the most gracious, the most merciful for all the things He has done to my life. Peace and blessings be upon His prophet, Muhammad (SAW) and his companions.

Special thanks to my parents Alhaji Yahaya Musa and Hajia Aishatu Bello for their support and prayers. My endless appreciation goes to the chairman of my supervisory committee, Prof. M. B. Mu’azu, for his tireless effort, support, constant guidance and encouragement throughout this thesis. The richness of this report is attributable to him. I just wish I can be like him. My warm appreciation goes to my other supervisor, Dr. E. A. Adedokun for his constant guidance, support and encouragement, his role in this work cannot be overemphasized, and Dr. Umoh for his observations, contributions and his constructive criticisms. I am grateful to Dr. S. Garba. To all my lecturers, I thank you for the valuable contributions, time created to go through my manuscript and for exposing me to many things that molded me into what I am today.

A profound gratitude goes to my hubby Abdullahi Ahmed Aliyu for his kindness, care, patience and understanding during this research work and always. His encouragement has been helpful to my studies. I must say that people like him are hard to come by, and I cannot stand without him. I say thanks to my lovely children Hameedah, Umar-Farouq and Yahaya for their patience and understanding.

I appreciate my siblings; Rakiya, Aisha, Jamila, Adama, Sa’adatu, Farhana and Mu’ammar, my cousins; Maryam, Fauziya, Aisha and Zainab, my nieces, my nephews, my in-laws especially Yahaya and Aminu for all the care, support and love shown to me.

I appreciate Abdullahi A., the Computer and Control research group and the LNG laboratory for their contributions to the success of this research.

I am grateful to my friends and my colleagues; Ene, Zainab M. A., Maimuna, Khadijah, Rukky, Safiya, Y. Ibrahim, Soja, Tee-jay, B. O. Sadiq, A. S. Abubakar, Y. M. Shehu, E. Obi, I. Ya'u, Ansari, Doguwa, Mal. Abdullahi Tukur, and the entire people of the Departments of Electrical, Communications and Computer Engineering.

ABSTRACT

This research is aimed at the development of an enhanced Trust Management Scheme (eTMS) for secured routing in opportunistic networks (oppnets). Opportunistic routing allows communication to be set up between nodes even without infrastructure in a delay tolerant fashion. Oppnet has become more pertinent now with the proliferation of autonomous mobile devices. However, malicious devices pose potential security threats (packet dropping, denial of service (DoS) attack, black hole attack, identification (ID) spoofing, etc.) to the performance of oppnets. This is due to the inherent characteristics of such networks like ever changing network topology and lack of a definite communication path between nodes amongst others. These characteristics created the issue of a lack of end-to-end connectivity thus making it extremely difficult to establish authentication between the source node and the destination node. In this work, the eTMS is developed by optimally determining, through simulations, the trust threshold value that decides the success probability of building behaviour trust in the standard trust management scheme (TMS) reported in literature. This is implemented in the proposed new variant of the probabilistic routing protocol for intermittently connected mobile ad-hoc network (PRoPICMAN), called the improved PRoPICMAN (iPRoPICMAN), that considered a multi-hop scenario. Simulation was carried out using the Opportunistic Network Environment (ONE) simulator using the benchmark Helsinki simulation area. The iPRoPICMAN when compared with the PRoPICMAN (without any TMS) increased the delivery probability from 0.2806 to 0.4156 (by 48%), increased the delay from 4613s to 5717s (by 23%) and reduced the overhead ratio from 68 to 46 (by 32%). This implied that iPRoPICMAN improved the performance of the oppnet compared with the PRoPICMAN but at the expense of delay. The iPRoPICMAN with the TMS (with threshold value set at 0.6) when compared with the iPRoPICMAN increased the delivery probability from 0.4156 to 0.4976 (19%), increased the delay from 5717s to 6713s (17%) and reduced the overhead ratio from 46 to 38 (17%). The iPRoPICMAN with eTMS (with optimal threshold value of 0.73) is shown to improve the delivery probability by 2% (0.4976 to 0.5065) and reduced the delay by 2% (6713s to 6607s) when compared with the iPRoPICMAN with TMS while maintaining a similar overhead ratio. Comparison between the iPRoPICMAN with TMS and the iPRoPICMAN with eTMS using delivery probability, delay and overhead ratio showed that the eTMS outperformed the TMS by 2% (0.5102 to 0.5221), 10% (2736s to 2464s) and 7% (135 to 126), respectively when run on the developed novel oppnet simulation area called the Ahmadu Bello University (ABU) simulation area. These results indicate that securing a routing protocol through trust-based methods improved the oppnet performance but sometimes with a trade-off in delay and that the threshold value is critical to the performance of such methods (as some values can result in zero network performance).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

Acronyms	Definition
ABU	Ahmadu Bello University
ALAR	Anti-Localization Anonymous Routing
BAN	Body Area Network
BP	Bundle Protocol
CPU	Central Processing Unit
DoS	Denial of Service
DTN	Delay Tolerant Network
eTMS	enhanced Trust Management Scheme
GUI	Graphic User Interface
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
IPN	Inter-Planetry Network
iPRoPICMAN	improved Probabilistic Routing Protocol for Intermittently Connected Mobile Ad-hoc Network
Jdk	java development kit
Jre	java runtime environmen
MANET	Mobile Ad-hoc Network
MITM	Man In the Middle
Oppnet	Opportunistic Network

ONE	Opportunistic Network Environment
PK	Public Key
PRoPICMAN	Probabilistic Routing Protocol for Intermittently
QoS	Quality of Service
RAM	Random Access Memory
SK	Private Key
TCP	Transmission Control Protocol
TMS	Trust Management Scheme
TTL	Time to Live
TTP	Trusted Third Party
VANET	Vehicular Ad-hoc Network
	Connected Mobile Ad-hoc Network
VFP	Verifiable Feedback Packets
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access

CHAPTER ONE

INTRODUCTION

1.1 Background

A delay or disruption tolerant network (DTN) is a type of network that provides communication through mobile nodes in an unstable and stressed environment. The network will normally be subjected to frequent and long lasting disconnections, high end-to-end path latency, limited resources (power, bandwidth etc.) and may comprise of more than one divergent set of protocols (Chen *et al.*, 2011; Fall & Farrell, 2008). DTN finds application in mobile, wireless and terrestrial environments. Typical DTNs include mobile ad-hoc networks (MANET), vehicular ad-hoc networks (VANET), sensor networks (like the acoustic underwater networks), inter-planetary networks (IPN) and opportunistic networks (oppnets) amongst others. The oppnet is an autonomous connection of users that communicate over relatively bandwidth-constrained wireless networks with or without infrastructure. According to Kaur & Kaur (2009) and Verma & Srivastava (2012), oppnet has the following basic features:

1) wirelessly connected nodes that are fixed or mobile (a node is an electronic device that is attached to a network and is capable of sending, receiving or forwarding information over communication channel).
2) A complete path between two nodes aspiring to communicate does not exist.
3) It does not have a fixed communication range.
4) Its routes are dynamically built as any node can opportunistically be used as next hop provided it is likely to bring the message closer to the final destination.
5) Network topology is also flexible as it can change at any time.

With recent advances in their routing technology, application of oppnets is gaining grounds in the following areas (Dinakar, *et al.*, 2013):

1) Tactical networks (like in military operations)
2) Emergency services (rescue operations, disaster recovery and hospital).
3) Education (campus networks, e-learning, virtual classrooms, etc.).
4) Network coverage extension so as to:
    a) provide backup in case of normal network failure,
    b) provide link for portable or temporary workstations,
    c) overcome situations where normal cabling is difficult or financially impractical, and
    d) connect remotely mobile users or networks

Since a complete path between two nodes aspiring to communicate does not exist, there is the problem of lack of end-to-end connectivity in an oppnet, which makes it impossible to make direct initial authentication from a source node to a destination node. The absence of the initial authentication makes malicious devices join the oppnet causing different forms of security threats.

A number of routing protocols exist in oppnets including amongst others (Vahdat & Becker 2000; Lindgren *et al*., 2003; Keranen & Ott, 2009; Verma & Srivastava, 2012; Lin *et al.*, 2008; Islam & Waldvogel 2011; Asgari *et al.*, 2013; Hu *et al.*, 2013):

1) Epidemic
2) Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET)
3) Spray-and-Wait

4) MaxProp
5) Direct delivery
6) Integrated routing protocol
7) Coding in Opportunistic Routing (CodeOR)
8) History Based Routing Protocol for Opportunistic Networks (HIBOp)
9) Practical Opportunistic Routing (POR)
10) Probabilistic Routing Protocol for Intermittently Connected Mobile Ad-hoc Network (PRoPICMAN)

These protocols are mostly used in order to increase forwarding probability and reduce transmission delay with little or no consideration for security. As such, malicious devices readily join the oppnet due to its aforementioned nature. These malicious devices receive and drop packets at will, masquerade themselves and hijack or tamper with messages meant for other nodes, exaggerate the trust value of other malicious devices or lower the trust value of a trusted node, etc. These behaviors could lead to loss of packets, increase in delay of message transmission, breach of privacy, compromising data confidentiality and integrity, and eventually, decrease in performance of the network evident from a decrease in delivery probability (Barai & Bhaumik, 2016). As such, security consideration is a critical issue in oppnet routing protocol.

Various mechanisms have been used to address security issues in oppnets, classified as trust-based and privacy-based protocols. The trust-based protocols are further divided into friend-vector based, familiarity-based, reputation based and hybrid-trust based (Barai & Bhaumik, 2016). On the other hand, the privacy-based protocols are divided into cryptography-based and cryptography-free as depicted in Figure 1.1 (Barai & Bhaumik, 2016).

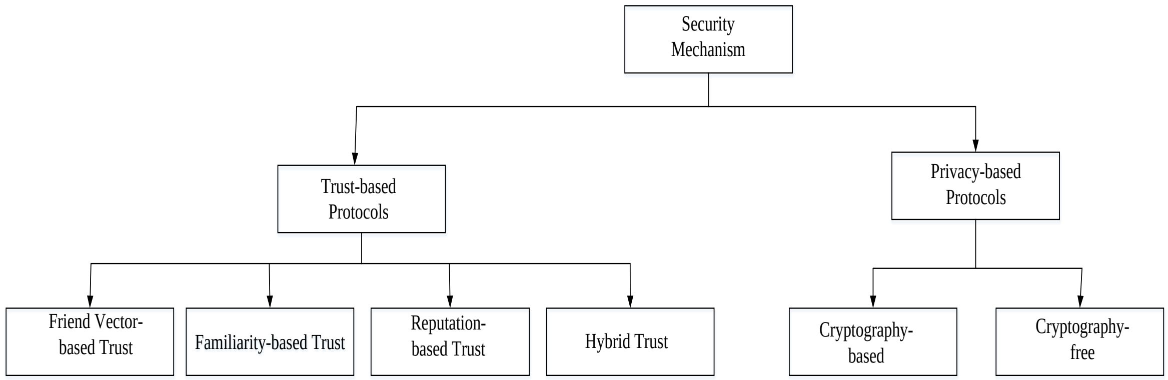

Figure 1.1: Taxonomy of Security Mechanisms in Oppnets (Barai & Bhaumik, 2016)

Another classification for the trust-based protocol has three types: social trust, environmental trust and similarity trust as depicted in Figure 1.2 (Trifunovic & Legendre, 2009). This classification presented the security mechanism as trust-based and cryptography-based.

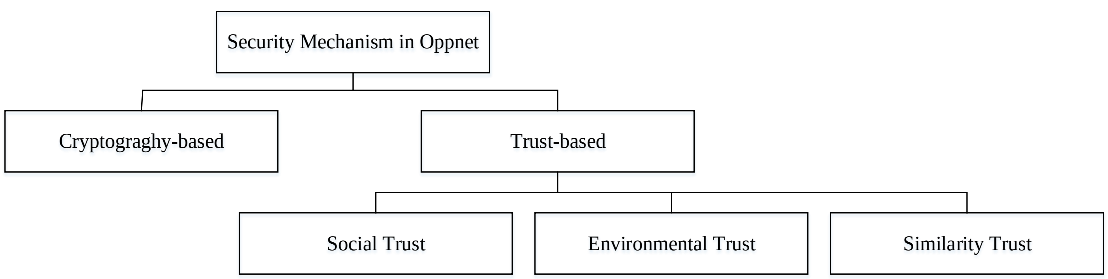

Figure 1.2: Security Mechanisms in Oppnets (Trifunovic & Legendre, 2009)

Due to the characteristics of oppnets (lack of end-to-end connectivity, unstable structure, etc.), cryptography-based schemes are unsuitable because nodes depend mainly on the next hop in order to forward data. This informed the need for a security mechanism which would ensure that the intermediate nodes do not behave maliciously. Cryptography-based algorithms are also not well realizable in oppnets because they require very complex and computationally intensive operations in order to obtain the required level of protection. Most devices in oppnets are made to be portable and energy efficient and as such even have less powerful hardware than those contained in conventional personal computers (PCs) (Xi *et al*., 2015; Trifunovic & Legendre, 2009; Barai & Bhaumik, 2016; Ciobanu *et al.*, 2016). In view of these, the trust-based schemes are the preferred security mechanisms for oppnets.

The trust management schemes (TMS) are dependent of the computation of trust values (which represent the trustworthiness of a node in the network). The establishment of trust can validate a nodes' legitimacy and resist malicious nodes more effectively in oppnets. However, the efficiency of the trust algorithms depends on what basic framework is followed in calculating the trust in oppnets (Xi *et al*., 2015; Trifunovic & Legendre, 2009; Barai & Bhaumik, 2016). The determination of the trust threshold values for the various schemes reported in literature (Xi *et al*., 2015; Yao *et al*., 2016; Trifunovic & Legendre, 2009, etc.) have mostly been assumed, thereby creating some degrees of vagueness.

The determination of an optimal thrust value for the TMS leading to the development of the enhanced TMS (eTMS) using the trust model of Xi *et al*., (2015) is one of the key focus areas of this research. This is because an optimal threshold value is expected to guarantee an increase in network performance in oppnets. An experimental approach is adopted in determining the optimal trust value from a series of values obtained from several experiments. Simulations of the TMS and the proposed eTMS, implemented on the PRoPICMAN and its improved version called the improved PRoPICMAN (iPRoPICMAN) that considers a multi-hop scenario, are then carried out using the opportunistic network environment (ONE) simulator on the benchmark Helsinki simulation area and the Ahmadu Bello University (ABU) (main campus) simulation area. The ABU simulation area is proposed using ONE as a novel simulation area for oppnet research.

1.2 Significance of the Research

Oppnets, due to their flexibility and ease of deployment, are fast gaining ground as networks of choice in emergency services, military operations, network expansion, etc., especially in difficult environments. However, because there is no guaranteed end-to-end connectivity, the possibility of oppnets being joined by malicious nodes, thereby threatening the confidentiality and integrity of data, is a critical one and a limiting factor to their widespread use. The TMS is one of the most preferred schemes for addressing security challenges in the oppnets and their effectiveness is largely dependent of their trust threshold value. This work, is therefore, focused on developing an eTMS in which the trust threshold value is optimally determined via simulation. The eTMS is expected to improve network security and performance in terms of delivery probability and overhead but with a likely trade-off in delay when implemented with the iPRoPICMAN.

1.3 Problem Statement

In oppnets, there is no initial authentication due to the problem of lack of contemporaneous end-to-end connectivity. Lack of end-to-end connectivity implies absence of feedback, which is a critical element of any authentication process. As a result of the absence of initial authentication, malicious devices can join the oppnet and impede the performance and integrity of the network. As such, there is no guaranteed confidentiality or integrity of data in the oppnet. There is therefore the need to device mechanisms to enhance the security of oppnets and invariably improve service delivery. One of the most preferred mechanisms is the TMS but its performance is highly dependent on the trust threshold value. This work focuses on developing an enhanced TMS (eTMS) that utilizes an optimal trust threshold value that is implemented on the proposed new variant of the PRoPICMAN called the iPRoPICMAN and this is expected to show improvement in the security of the oppnet through improved delivery probability and reduced delays. The TMS and eTMS were simulated in ONE using the benchmark Helsinki simulation area and the novel ABU simulation area.

1.4 Scope of the Research

This research deals with security issues specifically in oppnets and not with all categories of DTNs. TMS is the security solution adopted in this research because it had been proven to perform well for oppnets. The routing protocol used is the PRoPICMAN in order to create basis for comparison with the conventional TMS.

1.5 Aim and Objectives

The aim of this research is the development of an enhanced trust management scheme (eTMS) for secured routing in oppnets.

The objectives are as follows:

1) Development of a new variant of the PRoPICMAN called the iPRoPICMAN that considers a multi-hop scenario.
2) Development of the eTMS that uses the trust threshold value which determines the success probability of building behaviour trust in the TMS model of Xi *et al*., (2015).
3) Simulation and comparison of the performance of the TMS and eTMS on the iPRoPICMAN using the ONE simulator on the benchmark Helsinki simulation area

using delivery probability, delays and overhead ratio as performance evaluation metrics.

4) Modeling the Ahmadu Bello University (ABU) main campus as a novel oppnet simulation area called the ABU simulation area using the ONE simulator.
5) Simulation and comparison of the performance of the TMS and eTMS on the iPRoPICMAN using the ONE simulator on the ABU simulation area using delivery probability, delays and overhead ratio as performance evaluation metrics.

1.6 Methodology

The steps of the methodology carried out to achieve the stated objectives include:

1) Setting of the simulation environment using the following steps:
    a) Downloading and installing the java development kit (jdk) and java runtime environment (jre).
    b) Setting the java path
    c) Downloading and installing an integrated development environment (IDE).
    d) Downloading and installing the ONE simulator
    e) Interfacing the ONE simulator with the IDE
2) Modeling the simulation environment using the following:
    a) Defining the types of nodes (pedestrians, car, trams, etc.)
    b) Defining the characteristic of nodes (speed, transmission range, event generation interval, buffer space).
    c) Defining the simulation area
3) Replication of the of PRoPICMAN using the following:

a) Building nodes' profile using set of evidence pair

b) Hashing the evidences

c) Development of header for each node

d) Building the two-hop forwarding strategy

4) Development of the new variant of PRoPICMAN using the following:

a) Building nodes' profile using set of evidence pair

b) Hashing the evidences

c) Development of header for each node

d) Building the multi-hop forwarding strategy

5) Replication of the TMS using the following:

a) Obtaining nodes' profile information for development of certificate

b) Development of the successor selection algorithm

c) Development of the message selection algorithm

d) Development of the VFP propagation algorithm

6) Development of the eTMS using the following:

a) Obtaining nodes' profile information for development of certificate

b) Selection of optimum trust threshold value

c) Development of the successor selection algorithm

d) Development of the message selection algorithm

e) Development of the VFP propagation algorithm

7) Running simulations of the following:

a) TMS using the iPRoPICMAN (multi-hop) on Helsinki simulation area

b) eTMS using the iPRoPICMAN on Helsinki simulation area

c) Comparison between the TMS and the eTMS using delivery probability, delay and overhead ratio as the metrics

8) Modeling the ABU network simulation area using:

    a) Defining the types of nodes (pedestrians, car, etc.)

    b) Defining the characteristic of nodes (speed, transmission range, buffer space)

    c) Defining the simulation area

    d) Extracting line map from the map obtained.

    e) Importing the line map into the ONE simulator

9) Running simulations of the following:

    a) TMS using the iPRoPICMAN on ABU simulation area

    b) eTMS using the iPRoPICMAN on ABU simulation area

    c) Comparison between the TMS and the eTMS using delivery probability, delay and overhead ratio as the metrics.

1.7 Thesis Organization

This thesis is divided into Five Chapters. The general introduction has been presented in Chapter One. This include the significance and scope of the study, statement of the problem, thesis aim and objectives as well as the methodology adopted for the thesis. Chapter Two presents the detailed review of fundamental concept of oppnets, security in oppnet, ONE simulator and a review of similar research works. Chapter three presents the materials and methods for modeling of the scenario, configuration, installation as well as simulations. Result and discussions are presented in Chapter Four. Summary, conclusions and recommendations are discussed in Chapter Five.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This Chapter presents the review of fundamental concepts and the review of similar works. Fundamental concepts and theories relevant to the research are presented in the first segment while literatures on works similar and relevant to the research are presented in the second segment.

2.2 Review of Fundamental Concepts

This review covers the concept of the oppnets, routing in oppnets, security issues and mitigation schemes as well as the tools for evaluating the performance of delay tolerant networks (DTN).

2.2.1 Delay tolerant networks

DTN is a type of network that is intermittently connected and may suffer from frequent partitioning (disruption may occur due to limits of wireless radio range, energy resources, sparsity of mobile nodes, noise, attacks, etc.). It connects heterogeneous networks operating on different transmission media. Devices are connected in a decentralized manner with no central administration (Aneja & Vishal, 2015; Balaganesh & Nalini 2014). Communication in DTN is not limited to the internet and an end-to-end connectivity may not be guaranteed leading to the existence of long and variable delay in the network. Hence, routing protocol use the store and forward approach (Fall & Farrell, 2008; Cerf *et al.*, 2007; Singh *et al.*, 2015: Gao *et al*., 2013). Routing protocols in DTN are expected to be fault tolerant and have minimal latency even from unreliable routers. Internet protocols (IP) fails on most DTN because DTN violates many of the basic assumptions of the internet like (Cerf *et al*., 2007):

1) Availability of an end-to-end path between a source and a destination for the duration of a communication session.
2) All routers and end station support the transmission control protocol, TCP/IP protocol that repair error using retransmission based on timely and stable feedback from receivers.
3) An end-to-end loss is relatively low.
4) Selecting a single route between two nodes is sufficient for achieving acceptable communication performance.
5) Packet switching is the more appropriate for better performance and interoperability.

DTN is based on a new experimental protocol called the bundle protocol (BP). Bundle layer is an end-to-end message-oriented overlay situated between the transport layer and the application layer (as depicted in Figure 2.1) and it is responsible for the store-and-forward service in DTN. Devices capable of implementing the bundle layer are called DTN nodes.

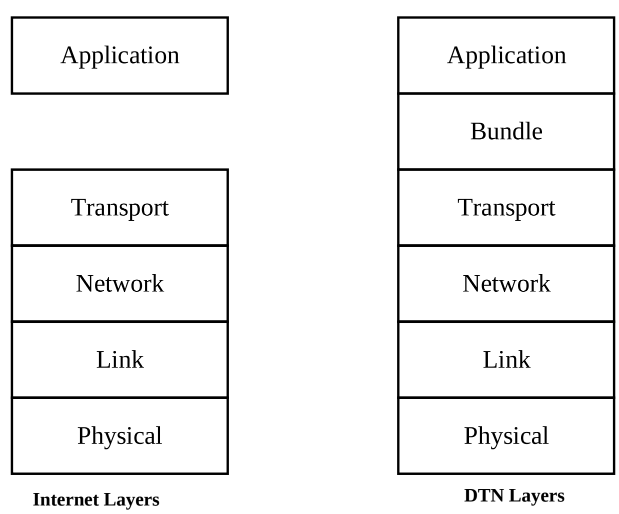

Figure 2.1: Comparison of IP Layer with DTN Protocol Layer (Cerf *et al*., 2007)

The bundle layer overlay use persistent storage in combatting network interruption. It contains several management and diagnostic features. Its uses a flexible naming scheme that considers different naming and addressing scheme in its naming syntax. Routing, congestion, and Security are the major concern in DTN (Fall & Farrell, 2008; Cerf *et al*., 2007). Types of DTN include (Cerf *et al*., 2007):

1) Mobile ad-hoc networks (MANET) for example, military tactical networks.
2) Vehicular ad-hoc networks (VANET) which finds application in DakNet, Message ferry, Village network, etc.
3) Sensor networks for example, the acoustic underwater networks.
4) Satellite network with moderate delay and periodic connectivity.
5) Inter-planetary networks (IPN) which find application in deep space networks.
6) Mule networks like the Zebranet (tracking of zebras in wildlife)
7) Opportunistic networks (oppnet) which is found in mobile or extreme terrestrial environment. The oppnet has the ability to extend communication across most of the aforementioned DTN types. This research is focused on the oppnet.

2.2.2 Concept of the opportunistic networks

Oppnet is a type of self-organizing DTN with a number of wireless nodes opportunistically communicating with each other in a "Store–Carry–Forward" fashion (Liu, 2014). Oppnets use wireless technologies such as Bluetooth, IEEE 802.11, Worldwide Interoperability for Microwave Access (WiMAX), and other short-range radio communication technologies. Nodes in oppnets usually have high mobility, low density, limited power, short radio range (Alajeely *et al*., 2015). It can be formed by small devices (fixed or mobile) communicating with each other over a wireless link and network contacts are intermittent or link performance is highly variable. Hence, in order to make communication possible, the intermediate node may take custody of data when there is no connection and forward it when connectivity resumes (Huang, *et al*., 2008). These networks are usually built around people, typically pedestrians. They are usually limited in speed, and have much slower propagation than wired or wireless links making delays in such networks remain (Ristanovic, 2012).

Some of the characteristics of oppnets include (Dinakar *et al.*, 2013):

1) When the nodes move away or turn off their power, the link may shutdown resulting in unstable connectivity.
2) Network partitioning may occur when no paths exist between the source and the destination.
3) The node communicates in a store-and-forward mechanism, etc.

The store-and-forward mechanism is alternatively called packet switching. Its procedure is as follows (Lindgren *et al.*, 2003; Zhao *et al.*, 2012):

1) The message is divided into fixed-length packet
2) Every channel has input and output buffers for one entire packet
3) Every packet is individually routed from the source to the destination
4) One step of the store-and-forward switching is called hop. In a hop, the whole packet from one output buffer is copied to the next input buffer.
5) Routing decisions are made by each intermediate router until the message reaches its final destination.

Due to the nature of this network, it has to trade a lower delay for a higher delivery ratio. It faces some challenges which include (Dinakar *et al.*, 2013; Huang, *et al.*, 2008; Shikfa, 2010; Verma & Srivastava, 2012; Ristanovic, 2012):

1) **Lack of end-to-end connectivity**: As a result of the lack of end-to-end connectivity, initial authentication becomes difficult. Without initial authentication, many malicious devices join this network causing security threats.
2) **Susceptibility to Failure:** Many protocols (like the transmission control protocol (TCP)) designed to assume quick return of acknowledgement and data are susceptible to failure. The lack of end-to-end path between the source and the destination coupled with limited resources and high delays effect TCP/IP protocols in opportunistic environments.
3) **Lack of Context Information:** Context information is the users' working address and institution; working environment and behavior; probability of meeting another user or visiting a particular place. Information about the context in which users communicate is a key piece of knowledge required to design an efficient routing protocol. It is important to identify suitable forwarders based on context information about the destination. Lack of this information makes it difficult to design routing protocols in oppnets.
4) **Unstable Structure:** The oppnet is challenging as its structure or connection is changing continuously. Nodes are highly mobile. The unstructured nature of oppnets makes them extremely difficult to put any performance guarantees on data dissemination. Also, routing and security mechanisms have to be highly dynamic and flexible, and should not depend on any particular route.
5) **Heterogeneity: N**odes do not rely on global infrastructure as they rely on various communication technologies. This makes naming a big issue because nodes do not

have a unique address that crosses different networks. This further makes authentication and trust establishment difficult.

6) **Storage Constraint**: Most devices involved in opportunistic routing have limited storage space. Nodes need to store messages in their buffer when they are not in communication range with other nodes. Since congestion occurs when a node's buffers become full, limited storage space makes congestion unpreventable in this network.

7) **Energy constraint:** Since the nodes are mobile, they cannot be kept to a permanent power source. As, such they run out of energy when they are taken away from the energy source for a long time. This further affects the stability of this network.

In order to resume communication after a break in connectivity due to unstable or incomplete path, unstable nodes (that keep the data during collapse periods) are used to forward the data when connectivity resumes (Dinakar, *et al.*, 2013). The oppnet is a self-configuring network that looks like an ad-hoc network and it uses all types of communication media like wireless fidelity (Wi-Fi), Bluetooth, satellites etc.

2.2.3 Opportunistic network node definition

In oppnet, a node is a device with short-range wireless communication capabilities. Communication range between two connected nodes is within walking distance (between 100-300 meters) (Yogi & Chinthala, 2014). A node can be a mobile device carried by people or a fixed device. A mobile node consists of a user carrying a mobile device that acts as an opportunistic network node (for example, personal digital assistant, mobile phones etc.). A fixed node is a device kept at a dedicated location, which is not under the direct control of a user. An example of a fixed node is the information sprinkler that is connected to the backbone network. The backbone network may be a wired network that joins a set of information sprinklers and synchronizes their operation. The information sprinkler can operate in a sprinkler mode (that is information is only dispersed), or in a sink mode (that is information only collected), or in both modes together (Yogi & Chinthiala, 2014). Figure 2.2 shows three information sprinklers and an optional sprinkler backbone, the connection link shows two adjacent nodes close enough to exchange message with one and other. The communication range indicated as dashed spheres is not an ideal sphere in practice due to communication signal interference with the surroundings.

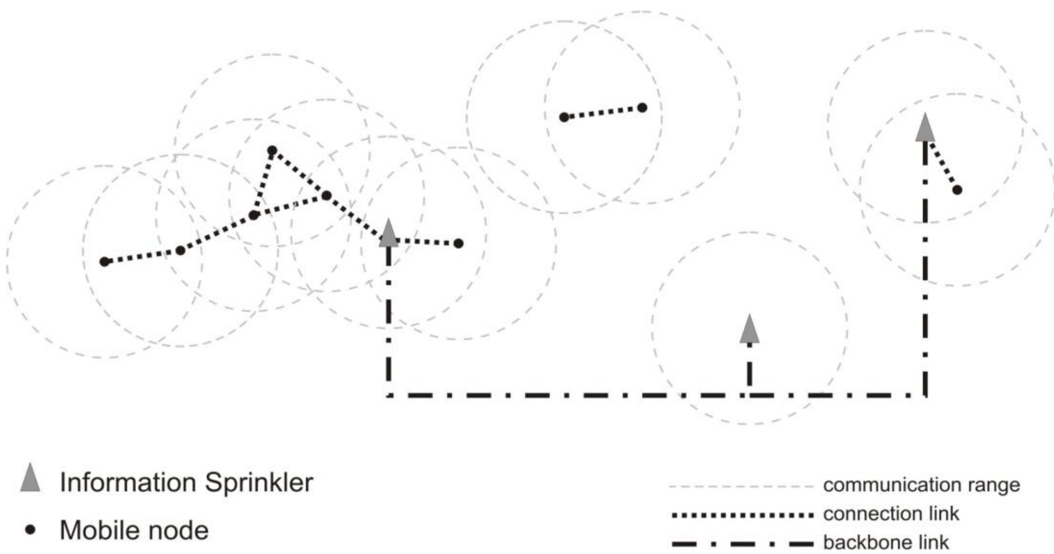

Figure 2.2: Typical Features of Oppnets (Yogi & Chinthala, 2014)

2.2.4 Seed and expanded opportunistic networks

In mobile networks or ad-hoc networks, the size of the network and locations of its nodes is pre-designed either in a fully deterministic way, or with a certain degree of randomness. Contrarily, the size of an oppnet and locations of nodes cannot be approximately predicted. The seed nodes are the initial set of nodes. They are deployed like typical ad-hoc networks.

Different devices (ranging from sensing and monitoring, to computing and communication systems), not originally employed as network nodes are detected and invited to join the seed nodes in order to perform certain tasks with tolerance in delay. These devices are the oppnet helpers. Helper nodes are used as intermediate nodes for forwarding data from the source to the destination. By integrating helpers into its fold, a seed oppnet grows into an expanded oppnet with vast sensing, communication and computation capabilities (Poonguzharselvi & Vetriselvi, 2012; Lilien *et al.*, 2006b).

The seed oppnet in Figure 2.3 grew into an expanded network by admitting the following helpers:

1) a microwave relay providing access to a microwave network
2) a computer network from a nearby college campus
3) a satellite, a smart appliance (e.g. a smart refrigerator) providing access to a home network
4) a cellphone tower representing cellphone infrastructure
5) a vehicular computer network, connected with wearable computer networks on and possibly within, the bodies of the occupants of a car

In general, the set of potential helpers for oppnets is very diverse, including communication, computing and sensor systems; wired and wireless; freestanding and embedded. As computing devices continue to become more and more pervasive, the pool of candidates will continue increasing dramatically. More densely populated areas will have a denser coverage by potential helpers. As a result, it will be easier to leverage capabilities of an oppnet in more densely populated areas since more resources become available in areas with a possibility of more humans (Lilien *et al.*, 2006b).

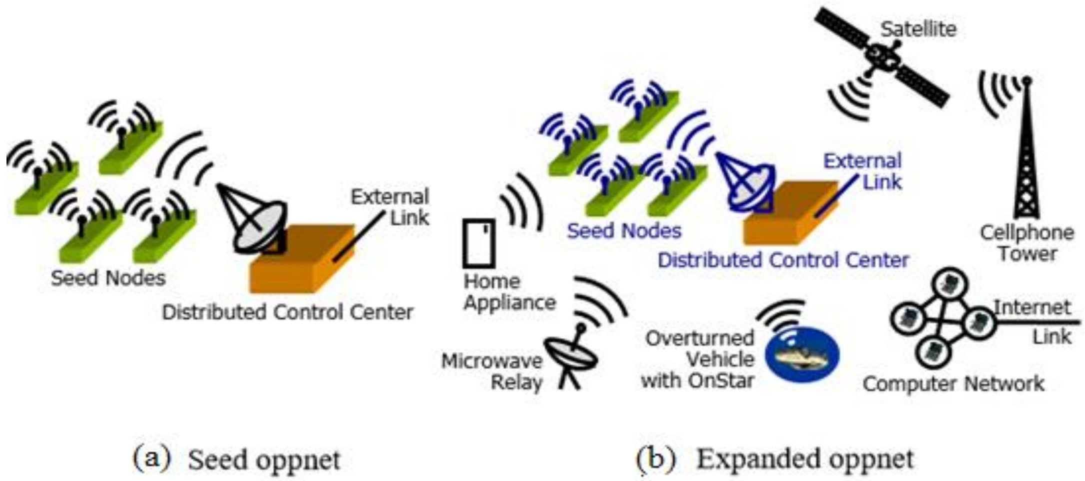

Figure 2.3: Seed and Expanded Oppnets (Lilien *et al*., 2006b)

In Figure 2.3, a seed oppnet is deployed in a metropolitan area after an earthquake. It finds many potential helpers, and integrates some of them into an expanded oppnet. One of the nodes of the expanded oppnet, a surveillance system, inspects the public scene with many objects. The image is passed to an oppnet node that analyzes it, and recognizes one of the objects as an overturned car as shown in Figure 2.3. Another node decides that the license plate of the car should be read. As the oppnet currently includes no image analysis specialist, a helper with these capabilities is found and integrated into the oppnet to read the license plate number. The license plate number is used by another newly integrated helper to check in a vehicle database whether the car is equipped with the OnStar communication system or not. If the appropriate OnStar center facility is contacted, it becomes a helper, and obtains a connection with the OnStar device in the car. The OnStar device in the car becomes a helper and is asked to contact body area networks (BANs) on and within bodies of car occupants. Each BAN available in the car turns to a helper and reports on the important signs of its owner. The reports from BANs are analyzed by scheduling nodes that handles the responder teams to ensure that people in the most serious condition are rescued earlier than the ones in less critical conditions (Lilien *et al*., 2006b).

2.2.5 Opportunistic Networks and Mobile Ad-hoc Network (MANETs)

Oppnet is one of the most interesting evolution of MANET. MANETS are autonomous, infrastructure-less, self-configuring networks that require knowledge of the network topology while oppnets can be both infrastructure-less or infrastructure-based and do not necessarily require knowledge of the network topology. MANETS usually focus on end-to-end routing between mobile nodes but oppnets go beyond MANETs as end-to-end routes can rarely exist. As a result, an oppnet is typically considered in environments with intermittently connected wireless nodes (for example, oppnets have been deployed as means of bridging between isolated rural areas, wildlife monitoring, and networking using buses that follow predictable routes) (Pelusi *et al*., 2006). An oppnet can resume link failure and forward data in a store-and-carry mechanism which is not possible in MANETs. As a result, oppnets overcome the technical challenges of MANETs to a better extent which parallels the increase in performance and can satisfy the user in a better way (Dinakar, *et al.*, 2013; Ristanovic, 2012).

Oppnets solely expose the one-hop communication scheme, where only directly connected nodes exchange message (for example, by learning about the information they were looking for), this implies that the oppnet reside on the application layer. MANETs resides on the network layer (Yogi and Chinthiala, 2014).

2.2.6 Routing in opportunistic network

Routing in oppnet shows different trend based on distinct goal. Different classifications exist considering different factors and interests (for example, using level of knowledge, copy of messages in the network, instance of information coding, methods to control movement of nodes and so on). Some classifications include:

1) Level of randomness: Zhang (2006) classified it as deterministic and stochastic routing. The deterministic approach includes space time routing, tree approach and modified shortest path while the stochastic approach consists of epidemic, coding-based and model-based.
2) Knowledge-based: This is classified as zero knowledge, partial knowledge and complete knowledge (Jain *et al*., 2004).
3) Based on routing type: This is classified as proactive and reactive routing protocols (Jain *et al*., 2004).
4) Nelson *et al*., 2009 classified it as forwarding-based and replication-based. The replication-based is further split into flooding-based and quota-based.
5) D'Souza and Jose, 2010 classified it as flooding-based, history-based and social device-based. The flooding-based is further broken into 'bounded copies' and 'additional information usage', while social device-based is broken into stationary, mobile and social context.
6) Other classification methods are forwarding utility-based, replication-based and coding-based. The coding-based is further classified as source coding and network coding and the replication-based as greedy, controlled, utility-based and resources allocation (Spyropoulos *et al*., 2010).
7) Another classification of the opportunistic routing is based on presence or absence of context information (Kaur & Kaur, 2009; Pelusi, *et al.*, 2006).

8) Oppnets are classified at a very top level into two categories: infrastructure-less and infrastructure-based networks. The infrastructure-less approach can be further divided into dissemination-based and context-based. In the infrastructure-based case, it is divided into the mobile infrastructure and fixed infrastructure (Kaur & Kaur, 2009; Pelusi, *et al.*, 2006). A typical taxonomy of oppnet is shown in Figure 2.4.

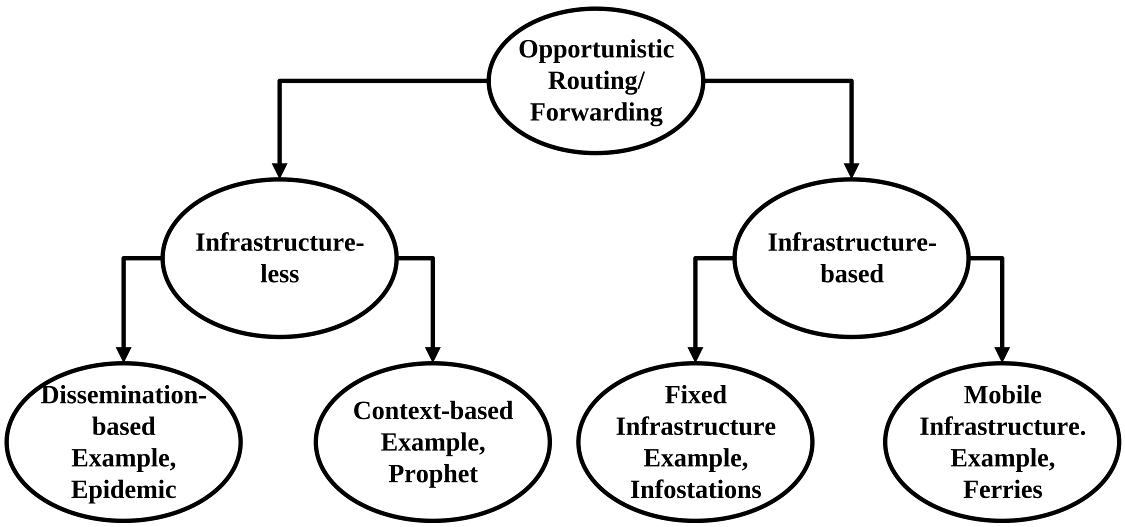

Figure 2.4: Taxonomy of Routing/Forwarding Techniques for Oppnets (Pelusi, *et al.*, 2006)

a) Dissemination-based Routing: It is also called the context oblivious routing. It provides a final delivery of messages to random destinations with minimal assumption of topology and connectivity of the network (Pelusi *et al.*, 2006). The heuristic behind this policy is that, the message should be broadcasted all over the network (flooding) expecting that it will eventually reach its destination. This technique works well in a highly dense mobile network where the contact opportunities needed for data diffusion are common. They limit message delay but consume resources (Pelusi *et al.*, 2006; Verma & Srivastava, 2012).

b) **Context-based Routing:** Mobility and contact pattern in the real world scenarios are likely to be predicted (for example, human mobility has a regular traffic pattern based on repeating behavioral pattern (like going to work, returning home, going to the market, and even social interaction)). It implies that if a node has visited a location or come in contact with a certain node several times before, it is likely to visit that location or meet that node again. Context-based routing utilizes this information to route data. In order to accomplish this, the protocol maintains a delivery predictability metric (message delivery probability between two nodes) at every node (Pelusi *et al.*, 2006).

   Hence, it is able to learn the past communication opportunities determined by users' mobility pattern and uses them efficiently in future. Context-based routing provides congestion control, acceptable quality of service (QoS) with lower overhead. However, in the absence of context information, it provides high overheads and message delay (Pelusi *et al.*, 2006; Verma & Srivastava, 2012)

c) **Routing Based on Fixed Infrastructure:** With infrastructure-based routing, a node intending to deliver a message keeps it until it comes within reach of an access point belonging to it. Access points are gateways which can provide Internet access or can be connected to a LAN (Kaur & Kaur, 2009). Two protocol variations exist, one in which only node-to-base station communication is possible while the second allows both node-to-base station and node-to-node communication, (that is, a node intending to send a message to a destination node delivers the message to the base station directly if they are within communication range, else it delivers the message opportunistically to a near node that will forward it to the base station when encountered). Based on these two examples, fixed base stations play passive roles in opportunistic forwarding strategy because they act as information sinks (Kaur & Kaur, 2009).

d) **Routing Based on Mobile Infrastructure:** This technique is also called the Carrier-Based routing. Nodes of the infrastructure are mobile data collectors, as they move in either predetermined or arbitrary routes around the network area gathering messages from nodes they pass by. When only node-to-carrier communications are allowed, these mobile nodes can be the only entities responsible for message delivery, thereby increasing connectivity in a sparse network by assuring that isolated nodes can also be reached (Pelusi, *et al.*, 2006).

2.2.6.1 Epidemic routing protocol

The epidemic protocol simply forwards all data to nodes in the network with minimal information of network connection or organization. Higher delivery probability is guaranteed even though it saturates the network with many copies of the same data. In the epidemic routing protocol, each host maintains two buffers, one for storing the message it originated and the other for messages it is buffering for other host. Each mobile device stores a summary vector which contains a summary of the messages stored in its buffer. When two nodes come into communication range, they exchange their summary vector, then each node check which message is missing and then request for the missing message. By so doing, the messages get flooded in the network, and eventually, the message reaches its destination.

The epidemic routing protocols are the only solution when context information about the user is absent, but they generate high overhead, suffer from network contention and may potentially lead to network congestion (Vahdat & Becker, 2000; Jones & Ward, 2006; Pelusi *et al.*, 2006; Verma & Srivastava, 2012).

2.2.6.2 PRoPHET routing protocol

PRoPHET is a variant of the epidemic routing protocol that uses past history of encounters and transitivity. It minimizes resource usage while still attempting to achieve the best case routing capabilities of the epidemic routing in intermittently connected networks. That is, when properly configured, prophet gives better performance in cases where resources (buffer space and bandwidth) are limited. Mobility and contact pattern in the real world scenarios are likely to be predicted (for example, human mobility have a regular traffic pattern based on repeating behavioral pattern (like going to work, returning home, going to the market, and even social interaction)). It implies that if a node has visited a location or come in contact with a certain node several times before, it is likely to visit that location or meet that node again. Prophet utilizes this information to route data. In order to accomplish this, prophet maintains a delivery predictability metric at every node. (The delivery predictability metric refers to message delivery probability between two nodes) (Pelusi *et al.*, 2006; Lindgren *et al.*, 2004).

Hence, it is able to learn the past communication opportunities determined by users' mobility pattern and uses them efficiently in future. Prophet as any other context-based routing, provides congestion control, acceptable QoS with lower overhead. In the absence of context information, it provides high overheads and message delay (Lindgren *et al.*, 2003; Pelusi *et al.*, 2006; Verma & Srivastava, 2012)

It allows messages to be buffered at intermediate nodes thereby exploiting the mobility pattern of nodes to bring messages closer to their final destinations by transferring them to other nodes as they meet. Figure 2.5 shows the mobility of nodes in a scenario that can be used to deliver a message to its final destination. The physical position of four nodes (A, B, C, and D) at four time instants is represented in the four sub-figures (a) to (d). At the beginning, node A has a message intended to be delivered to node D, but there is no direct path between nodes A and D due to limited range of wireless connections. The mobility of nodes allows the message to first be transferred to node B, and then to node C when they come in communication range, and eventually to node D. This is known as transitive networking.

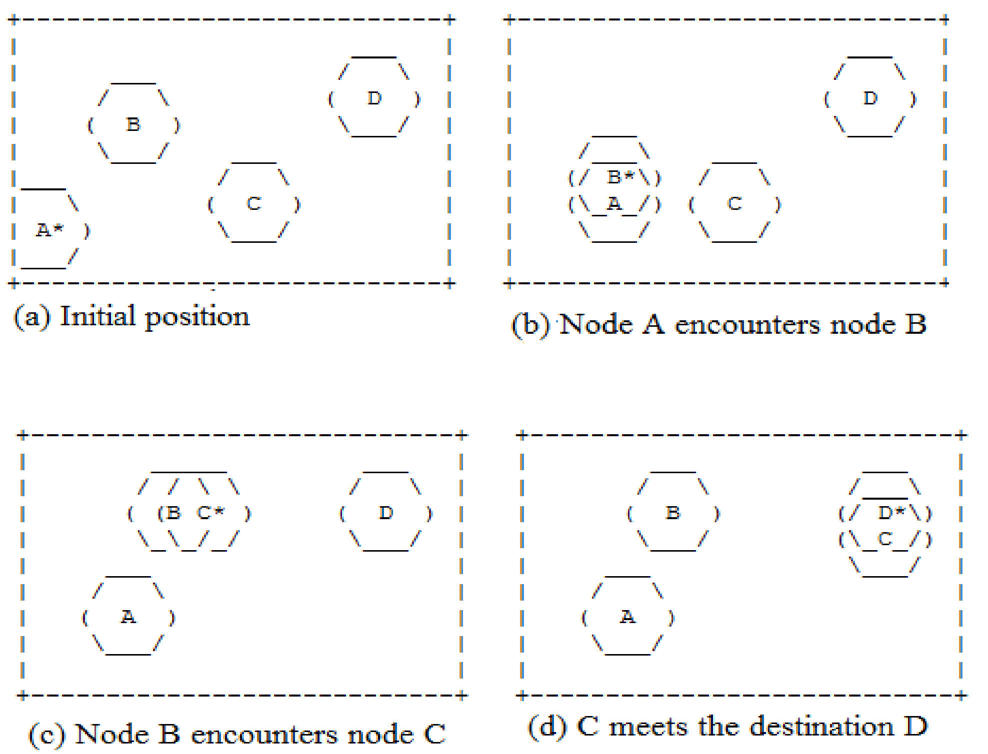

Figure 2.5: Transitive Communication Process (Lindgren *et al*., 2003)

Nodes need to know the contact probabilities to all other nodes in the network, every node maintains a probability matrix. A node then updates its metric with that of other nodes if the other node has a more recent update time attribute so that the two nodes will have identical contact probability matrices after communication. The integrated routing protocol was observed to overcome the challenges of other routing schemes with respect to oppnets (Verma & Srivastava, 2012).

2.2.6.3 PRoPHET routing protocol probabilities

The metric is updated whenever a node meets with other nodes, hence, nodes that meet often have a high message delivery probability. When node x meets node y, the delivery probability of node x for y is updated by (Verma & Srivastava, 2012):

$$P'_{xy} = P_{xy} + (1+P_{xy})\, P_0 \qquad (2.1)$$

where P_0 is an initial probability (P_0 ranges between 0 and 1), P'_{xy} is the current delivery probability of node x for y and P_{xy} is former delivery probability of node x for y.

When node x does not meet with node y for some predefined time, the delivery probability decreases by (Verma & Srivastava, 2012):

$$P'_{xy} = \gamma^k P_{xy} \qquad (2.2)$$

where γ is the aging factor (γ < 1), and k is the number of time units since the last update. When node x receives node y's delivery probabilities, node x may compute the transitive delivery probability through y to z by (Verma & Srivastava, 2012):

$$P'_{XZ} = P_{XZ} + (1 + P_{XZ}) P_{xy}\, P_{YZ}\, \beta \qquad (2.3)$$

where β is a design parameter for the impact of transitivity and according to Verma & Srivastava (2012), $\beta \in [0,1]$.

2.2.6.4 Routing strategies of the PRoPHET routing protocol

When a message arrives at a node, there might not be an available path to the destination. As such, the node might have to buffer the message and upon each encounter with another node, a decision must be made on whether or not to transfer a particular message. Furthermore, it may also be sensible to forward a message to multiple nodes to increase the probability that a message is really delivered to its destination (Verma & Srivastava, 2012). Whenever a node meets with other nodes, they all exchange their messages (or their probability matrices). If the destination of a message is the receiver itself, the message is delivered. Otherwise, if the probability of delivering the message to its destination through this receiver node is greater than or equal to a certain threshold, the message is stored in the receiver's storage to forward to the destination. If the probability is less than the threshold, the receiver discards the message. If all neighbors of sender node have no knowledge about destination of message and sender has waited more than a configured time, sender will broadcast it to all its current neighbors. This process will be repeated at each node until it reaches its final destination (Lindgren *et al*., 2004).

2.2.6.5 Probabilistic routing protocol for intermittently connected mobile ad-hoc network (PRoPICMAN)

PRoPICMAN is a probabilistic routing protocol that was developed to distribute messages through opportunistic mobile ad-hoc networks. The PRoPICMAN modified the delivery predictability mechanism of the prophet to consider more social context rather than history of encounter in order to make it realistic. The aim is to reduce the amount of resources consumed in message delivery and to increase the message delivery rate (Nguyen *et al*., 2007).

In PRoPICMAN, when a source node (say S) intends to send a message to a destination node (say D). S first sends the header of the message (which contains the information S knows about D). All neighbors use the header to compute their delivery probability. The message will be forwarded to the route with the highest delivery probability using two hop count to the destination. In order to select the best forwarder, S builds a message header H_m which is a concatenation of all hashed pair evidences. S sends H_m to all its neighbors, neighbors compare each of their hashed values of evidences to that of H_m. A weight is recorded for each match in attribute. Neighbors use this to calculate their delivery probabilities according to (Nguyen *et al*., 2007):

$$P_A = \frac{\sum Matched(W_i)}{\sum W_{Mi}} \quad (2.4a)$$

$$0 \leq P_A = \ \leq 1 \quad (2.4b)$$

where $\sum$ Matched (W_i) represents the total weights of evidences whose hashed values matched that of H_m, P_A represents the delivery probability, $\sum W_{Mi}$ is the total weights of all evidences that S knows about the destination.

For the next hop, the neighbor in turn sends H_m to their neighbors to further calculate the delivery probability. The delivery probability is then sent to S. S forwards the message through the two-hop nodes with the highest delivery probability.

2.2.7 Security issues in oppnets

When opportunistic routing started, the intension of every routing scheme is to increase the opportunity in forwarding information in this highly challenged network (absence of end-to-end connectivity, lack of context information, unstable structure, etc.). Most schemes welcome any node in the forwarding process with the assumption that; the higher the number of nodes, the higher the probability of getting a message to the destination. Most work focused on developing and improving the routing schemes for opportunistic network without considering trust and security. It is true that when more nodes join the network, more opportunities will be available to continue the routing process. This is because some devices will be available to sustain the connection when some are down and this will lead to an increase in delivery probability and decrease in delay. However, some selfish and malicious nodes may join this network and perform notorious activities which will eventually reduce the integrity of data, delivery probability and the importance of opportunistic network especially in emergency situation (Kaur & Kaur, 2009).

Security is a crucial issue in opportunistic routing. It should be implemented in the oppnet to encourage more users to join this network. In this network, there is no fixed infrastructure, so message is forwarded through many intermediate nodes. Some users may not want to disclose their identity (context information) before joining the network when security is not guaranteed. Helpers will refuse to join the network when proper privacy protection is not built into it because they may not want to disclose their sensitive resources (private data) to opportunistic network. Secured opportunistic routing will attract honest nodes to the network as well as isolate malicious node from the networks. This will in the long run improve the routing performance. For example, in Figure 2.6, groups A and B are mobile nodes and they easily come in communication range with other devices. If groups A and B refuse to join the network because of security reason, it will take a longer time for the initiator to reach the responder. The more the number of honest node in the network the faster the networking.

Also, when groups A and B decide to be malicious, their presence in the network will retard the routing process. The increase in number of such nodes will not guarantee higher delivery probability as they may accept packets and drop the packet deliberately. This can make the initiator wait infinitely for the responder (Le *et al*., 2009).

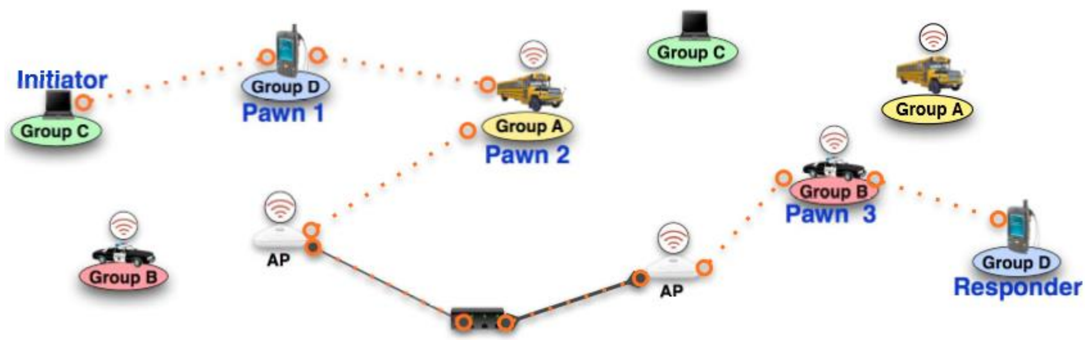

Figure 2.6: Opportunistic Networking (Le *et al*., 2009)

2.2.8 Attacks in opportunistic network

There are varieties of attacks in the oppnet. The attacks can be classified as active or passive attacks. A passive attack only snoops the message exchanged in the network without altering it, for example, eavesdropping, jamming and traffic analysis and monitoring. Active attacks attempt to alter or destroy the data being exchanged in the network, thereby disrupting the normal functioning of the network. Active attacks can be internal or external. External attacks are triggered by nodes that do not belong to the network. Internal attacks are from compromised nodes that are part of the network. Internal attacks are more severe and hard to detect than external attacks because the attackers are already part of the network (Kaur, 2015). Some of the possible attacks in oppnets include (Lilien *et al.*, 2006a: Wang *et al.*, 2014):

1) **Packet dropping**: this is a situation where malicious devices drop some or all packets sent between two nodes (say the victim and the responder). It can collect packet at random, drop packet containing important information and forward packets that are not important. Packet dropping might be due to selfishness, where a node drops any packet it is not interested in, in order to conserve its battery or buffer space, or to protect its private information such as location, identity and message content (Jingwei *et al.*, 2013). Some selfish nodes might not even accept any packet(s) not meant for them. Some literatures considered selfishness as another attack on its own (for example, Alajeely *et al*., 2015), where selfish nodes refuse to cooperate with other nodes but use network services.
2) **Eavesdropping:** An attacker in the vicinity of the user has the ability to eavesdrop on the user's wireless signals. This attacker could be a curious neighbor. Such an attacker requires little in the way of equipment a simple wireless receiver is sufficient to observe messages in the system. This form of attack is easily defeated by simply encrypting messages and the destination with a key that the attacker does not have. Eavesdropping is an example of man in the middle (MITM) attack (Le *et al*., 2009).
3) **Denial of Service (DoS) Attack**: malicious devices generate a number of false messages (for example, false request for help) just to keep an important node (say rescue team) busy and unavailable for real emergencies. The large number of messages flowing in the network can lead to congestion. DoS attack can be on weak links where malicious devices target weak devices that are critical to opportunistic

network operation (for example, a cellphone connecting two cluster of users). They send enough packets to run down the battery of the cell phone, whereas the battery should be used carefully until another inter-cluster connection is found.

4) **Worm hole attack**: in worm hole attack, malicious nodes record packets at specific location on the network and take them to another location on the network. They later retransmit them from that location (Alajeely *et al*., 2015).

5) **Black hole attack:** malicious devices use routing protocol to advertise itself as having the shortest path or higher delivery probability to the node whose packets it wants to intercept. When the attacker receives a request for a route to the destination node, it creates a reply message which advertises itself as a valid path to destination. The attacker consumes the intercepted packets without any forwarding (Kaur, 2015).

6) **Identification (ID) spoofing**: ID spoofing is also known as **Sybil attack**. This involves a masquerading malicious device generating requests with multiple identities (this may result in many false alarms for a rescue team). A device capable of spoofing identities of trusted nodes may pose many kinds of attacks. Some of which includes (Yao *et al*., 2016):

   a) Stealing messages meant for other nodes.

   b) Trust-Boosting: malicious devices exaggerate the trust value of other malicious nodes by submitting biased recommendation about them. With a higher trust value, they tend to attract more packets to drop.

   c) Defamation: malicious devices may lower the trust value of trusted node by submitting bad recommendation against it.

d) Promise-Then-Drop: the attacker promises to forward packets to other nodes, then it silently drops the packet that it received.

It is worthy of note that an attacker may not be limited to a single type of attack. For example, a local eavesdropper can easily sniff and modify information at will. Such attacker types must also be considered in designing the defense system.

2.2.9 Security/privacy solution in oppnets

Defense strategy in opportunistic routing takes two folds: preventive defense and reactive defense. Preventive defense involves blocking malicious helpers from joining the network while reactive defense involves detecting malicious devices in the network through their notorious behavior and isolating them. The trust management scheme has the ability to defend both folds. Security requirements in oppnets include (Wu *et al*., 2015):

1) **Authentication, authorization and access control:** centralized servers or online central certificate cannot be used in oppnet due to the absence of end-to-end secured connection. As such, effective authentication, authorization and access control is needed to prevent malicious devices intruding into the network.
2) **Data confidentiality and integrity:** integrity implies that data should not be tempered with in the process of forwarding, while confidentiality implies that sensitive data are not disclosed to unauthorized third party. Integrity and confidentiality should be ensured since messages forwarded by oppnet nodes are mostly wireless signals that are vulnerable to eavesdropping. Also, more helpers will readily join the network.

3) **Privacy protection:** both identity and location privacy protection are important. There should be an appropriate means to balance between users' privacy protection and network information requirement when designing oppnet routing protocols.
4) **Cooperation:** cooperative behavior of nodes will affect the performance of oppnets to a large extent. A situation where nodes refuse to cooperate and relay messages to other node leads to a considerable delay in the network.
5) **Trust management:** trust management should be designed to calculate the level of trust between nodes in order to prevent malicious nodes from intruding into the network. The different forms of trust should be considered in order to suit different scenarios.

2.2.10 Trust management

Trust is a measure of a device's faithfulness and an indicator of quality of service that the device can provide, which is sometimes interchanged with reputation. However, reputation has to do with the perception regarding an agents' behavior norms, which is held by other agents, based on experience and observation of its past actions. Trust management is a mechanism that deals with the evaluation, collection and propagation of trust in a network. It evaluates the level of trust between nodes in order to obtain the availability of oppnet and to resist malicious dropping. Trust management in DTN is challenging due to the frequently changing network topology. This make DTNs have high delay (Kattimani & Indikar, 2015). Different forms of trust exist in order to fit different scenarios (Wu *et al.*, 2015; Trifunovic & Legendre, 2009). These include (Trifunovic & Legendre, 2009):

1) **Social Trust:** a secured and reliable friend ties (common friends, common interest) can be achieved from a proper understanding of the mobility pattern of devices. Friends list are exchanged and saved each time a node is encountered. This is used in creating a friendship graph (friendship graph is a graph that is organized in levels constituted by a set of nodes having the same distance from a local node, where edges only exist between nodes in sequenced levels). Trust is evaluated as function of hop distance (they share an inverse relationship) and positive recommendations from other nodes (they share a direct relationship). A transitivity of trust is assumed over a reasonable distance (up to six hops). A peer first asks its friends for opinion before working with an unknown peer. Friends inform each other when a malicious peer is found in the network in order to eliminate the malicious peer. Trust and recommendation values are defined by probability. There are advantages and disadvantages of taking friends connections as a basis of establishing trust. Sybil attacks is prevented by secured pairing process where the human entity behind the identity is verified. On the other hand, the transitivity of trust allows a Sybil user to only need to establish one trusted relation to acquire full trust other nodes. Also, the friendship graph is loosely connected and does not guarantee a regular interaction. As such, trust assessment of nodes that are met regularly is not guaranteed.
2) **Environmental Trust:** this is called **familiarity based models** in some literatures (Barai & Bhaumik, 2016). There are certain individuals people regularly share the same activity or the same space in everyday life. (for example, the same coworker sharing the same office every day, people living in the same building apartment or estate, people who regularly go to the same mosque, attend the same church or regular

visit a particular location (say gymnasium) at a particular time). These communities can be identified using a community detection algorithm that carefully observe the environment and analyze the peers in the area over time. Social notion is not guaranteed here since member of the community may not know or be friends with each other but only share the same space/environment for a significant amount of time. This method has an advantage of not relying on users' interaction over the friends-based social trust. Also, a given amount of trust in a familiar stranger can be justified which is usefully for preventing Sybil attack. However, this method is not as secured as basing trust on friend ties since community detection cannot assure for a certain entity to be behind the proclaimed identity. Also, special care has to be taken in making the current community detection mechanisms resilient to attacks since they are easily tempered with.

3) **Similarity Trust:** the comparison of the recommendation of other users with direct experience is used to assess trust on similarity interest. It is seen as part of a reputation system and it is called the reputation based trust in some literatures (Alajeely *et al*., 2015; Barai & Bhaumik, 2015). This trust might identify unwanted content or select peers which for their similar taste are more relevant in a given context. Making similarity trust a secured way of assessing trust is difficult in oppnet because:

   It requires a reputation system and users need to have experience. In a decentralized environment, recommendations can easily be forged and manipulated to be similar with targeted peer.

4) **Data-centric Trust:** the aforementioned trust notions are entity-centric (trust as a relation among entities) and slow to change. Oppnet systems are data-centric in their functionality and operate in an unstable manner. As such, it is worthy to establish trust in data rather than the entity reporting them. Also, nodes can be unreliable, faulty and insufficiently equipped for data in oppnets (Wu *et al*., 2015).

5) **Hybrid Trust**: this model is a combination of two or more of the aforementioned trusts. For example, social studies have shown that people in close proximity tend to have similarities with others. People tend to communicate, cooperate, socialize and potentially, trust each other if they belong to the same community of interest or activity. Table 2.1 summarizes the comparison of different forms of trust in oppnets.

Table 2.1: Comparison of Different Forms of Trust in Oppnets (Wu *et al*., 2015)

Trust type	Reputation-based trust	Social trust	Environmental trust	Data-Centric trust
Typical scheme	Reputation-assisted data forwarding in Oppnets	Explicit and implicit social trust	Community detection	Trust based on Dempster-Shafer theory
Core idea	Positive feedback message	Common friends, common interest and the distance in the social graph	Frequent encounter of familiarity	Evaluation of the trustworthiness of data rather than entities themselves
Anti-attack type	Anti-blackhole attack and anti-spoofing attack	Anti-Sybil attack	Certain amount of anti-sybil attack	Anti-spoofing attack
Overhead generated	Large	Medium	Small	Medium
Disadvantages	Large network overhead	Requiring interaction of the user and regular interactions not guaranteed	Easily tampered and weak trust in a familiar stranger	Fewer research in opportunistic networks

The Dempter-Shafer theory indicated in Table 2.1 is a theory of evidence which is a general framework for reasoning with uncertainty. It is used to combine separate pieces of evidences (or information) to calculate the probability of an event (Wu *et al*., 2015).

2.2.11 Trust management scheme based on behavior feedback for oppnets

Human behavior has the feature of repeatability and periodicity when carefully studied. This is because people do not move randomly but have several specific predictable pattern which is concluded on social attributes. A person social attributes constitute his social context. People with similar social attribute will have a high probability of meeting each other. Based on this, Xi *et al*., (2015) presented a trust management scheme based on feedback information propagated by other nodes, which uses a social context-based key management algorithm (as such, it can be applied to social context routing protocol like the PRoPICMAN, Bubble rab, etc.). The following assumptions are observed in the scheme:

1) The social context information of a node is known by other nodes
2) Nodes have limited buffer for message and certificate graph
3) Nodes communicate with each other when they are within a given communication range.

By using the social context information, mobile nodes issue and exchange certificate for each other. This gives the mobile node the opportunity to query valid certificate path for the nodes they encounter in less time. Certificate is issued only between nodes with reasonably high similar social attributes as portrayed in Figure. 2.7. This is so in order to reduce the storage needed by each node certificate sub-graph.

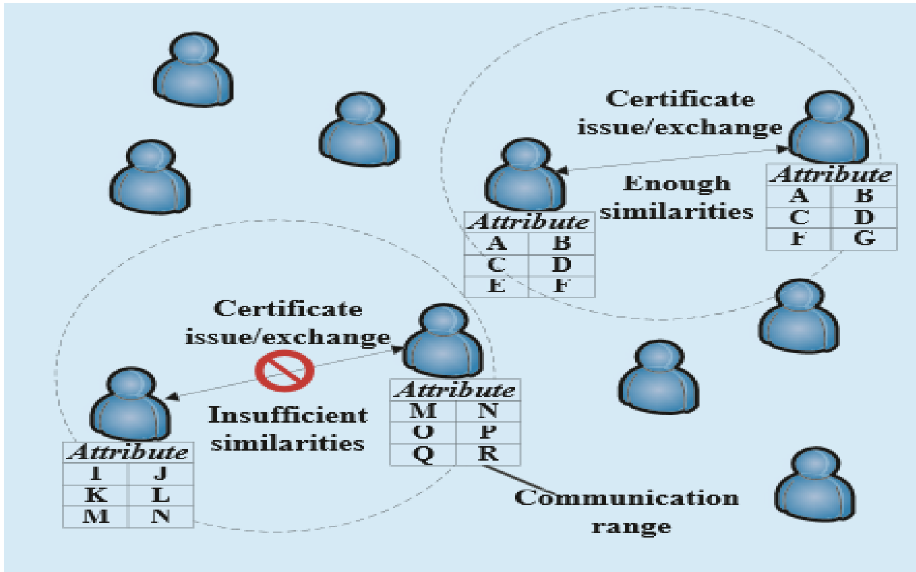

Figure 2.7: Social Context-Based Certificate Exchange (Xi *et al*., 2015).

The trust model considered four basic components (Oppnet trust model, social context-based key management, secured forwarding and feedback system) as depicted in Figure 2.8 (Xi *et al*., 2015).

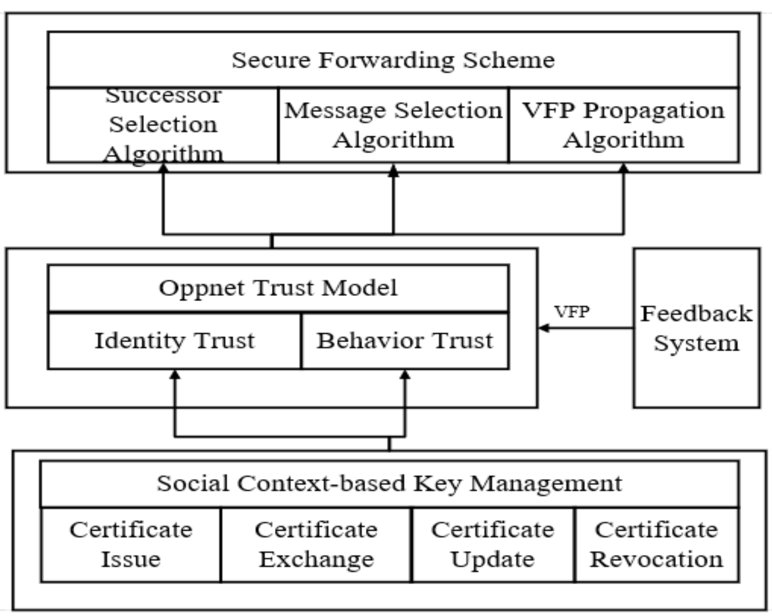

Figure 2.8: Trust Management Scheme Based on Behavior Feedback (Xi *et al*., 2015).

The key management system issues, exchange, update and revoke certificates are presented as follows (Xi *et al*., 2015):

1) **Certificate issue:**

a) Mobile nodes generate their public key (PK) and private key (SK) pairs.

b) When a node (say u) comes in contact with another node (say v), u will check if it owns a valid certificate part to v,

c) If no, u will query the attribute list of v and calculate the match of social attributes Match(u,v) between itself and v.

d) If match(u,v) is less than a given threshold Thres$_{attri}$, u refuses to make connection with v.

e) Else if match(u,v) is greater than the given threshold Thres$_{attri}$, u will use some local method to inspect the integrity and security of v in order to ensure that v's identity and security can be trusted.

f) If the identity ID_v and security of v can be trusted, u will issue a new certificate $Cert_{u\_v}$ to v, which is signed by u.

g) The certificate's reliability $Relia_{u\_v}$ and period of validity T_{u_v} are functions of match(u,v) as shown in the equations (2.5) and (2.6) (Xi *et al*., 2015):

$$Relia_{u\_v} = Relia_{max} * \text{match(u,v)} \tag{2.5}$$

$$T_{u\_v} = T_{max} * \text{match(u,v)} \tag{2.6}$$

where $Relia_{max}$ and T_{max} are maximum reliability and maximum period of validity and both of them are set up specific to a network environment. The issued certificate $Cert_{u\_v}$ is given as (Xi *et al*., 2015):

$$Cert_{u\_v} = Sig_u(ID_u, \quad PK_v \ \ Relia_{u\_v} \ \ T_{u\_v}) \tag{2.7}$$

$Cert_{u\_v}$ is an indication that u trusts the authenticity of v.

2) **Certificate exchange:** During the warm up period, nodes exchange the certificate database with every qualified encountered node in order to collect as many certificates as possible. At on-stream period, nodes will only exchange certificate information with encountered nodes whose match of social attribute is greater than Thres_{attri}.

3) **Certificate update:** when the validity period of a certificate expires, the issuer will have to wait for another encounter to repeat the process of certificate issue. Also, node u will maintain a counter $Counter_E$ for the related edge E which is in its certificate graph G_u. $Counter_E$ will add 1 whenever the E is used to authenticate the validity of another node.

4) **Certificate revocation:** certificates are revoked whenever the private key of a mobile node leaks, or the binding of a nodes' private key and public key has changed. A certificate can be revoked by both the issuer and the owner of the certificate. The certificate revocation information has the highest priority in the network traffic in order to notify other node who may also possess the certificate.

Mobile nodes can share their public keys with others efficiently based on the social-context key management explained. This guarantees users' mutual authentication and message confidentiality. In order to check packet dropping by malicious node, secured forwarding based on verifiable feedback packets (VFP) was introduced.

The VFP contains feedback information (identity of the source node, list of positive forwarders and message sequence number) and the signature of the VPF creator. The signature is generated using the private key of the creator. The validity of the VFP can be tested by anyone who has the public key of the creator. This prevents raising of reputation by malicious nodes.

Assuming the best transmission path from a source node n_a to a destination n_e is $n_a \rightarrow n_b \rightarrow n_c \rightarrow n_d \rightarrow n_e$ where $n_{b,}$ n_c and n_d are intermediate nodes. n_b creates a VFP for n_a's positive action when n_a forwards the message to n_b which is signed by n_b's private key. Similarly, n_c to n_b and n_d to n_c. Then n_b propagates the VFP for n_a to every node it meets. Network overhead due to VFP propagation is limited due to the small size of the VFP. When the VFP of n_a is received by another node from n_b, the receiver will use n_a's public key to check if the VFP is valid. If valid, the receiver will increase the trust value of n_a for its positive forwarding behavior and forward this VFP to other nodes when they meet (Xi *et al*., 2015). Based on the social context-based key management and the VFP, two kinds of trust models are employed (Xi *et al*., 2015):

1) Identity Trust: this is based on the social context-based key management scheme. Identity trust between two node (say n_a and n_b) implies that they can authenticate each other by their public and private key pairs. This can protect the confidentiality, integrity and privacy of the message. That is, the existence of identity trust between two nodes ensures security of transmitting and forwarding message.
2) Behaviour Trust: behavior trust is based on VFP propagation. If n_b has enough valid and verifiable VFP's of n_a, then n_b behavior trust n_a. Large VFP's is an indication that nodes help in forwarding a lot of messages, implying that it is not a malicious node that drops packets.

Bahaviour trust is a low-level trust when compared to identity trust but it is still important since communication in oppnet is via forwarding process. If only identity trust is used, a lot of honest node whose identity cannot be verified will be excluded in the forwarding process. However, with behavior trust, the security for forwarding between unacquainted nodes can be ensured even if their identity cannot be verified. Based on the two trust models, a self-organizing protocol was presented using three algorithms (Xi *et al.*, 2015):

1) Successor Selection Algorithm: this is used to select which trusted node has more encounter probability with the destination. Using this scenario again, $n_a \rightarrow\ n_b \rightarrow n_c \rightarrow\ n_d \rightarrow n_e$, when n_b meets n_c intends to forward the message to n_c using any of the social context-based routing protocol (PRoPICMAN in this case), n_b will first check for n_c identity by checking if it has a local certificate graph to n_c else, n_b will try to query the behaviour of n_c by checking how many VFPs of n_c have been propagated by trusted nodes in the network. That is, whenever identity trust cannot be achieved, behaviour trust is used to decide if the successor is qualified for secure forwarding. The successor selection algorithm is presented as in Figure 2.9 (Xi *et al.*, 2015):

1: n_i: mobile node
2: PK_i: n_i's public key; SK_i: n_i's private key;
3: G_i: n_i's local certificate graph
4: V_i: the binding of $(PK_i \| n_i)$
5: $C_{i,j}$: certificate chain from V_i to V_j
6: $VFP_{i,j}$: the verifiable feedback packet (VFP) which is issued by n_i for n_j's positive behaviors.
7: M: message which need to be transmitted from source node n_S to destination node n_D
8: $TC_{i,j}$: each node n_i holds a trust counter for the other node n_j's positive behaviors.
9: th_i: When n_i get more than th_i verifiable feedback packets $VFP_{i,j}$ for a stranger node n_j, n_i will trust n_j's behaviors.
10: Max_i: n_i's maximum copy for the propagation of VFP
11: $Ex_{i,j}$: the expiry date for $VFP_{i,j}$
12: Message M is forwarded though the path $n_S \rightarrow n_a \rightarrow n_b \rightarrow n_c \rightarrow n_D$ based social context routing protocols, When n_b encounter n_c which has more delivery competency:
13: **If** n_b can find a valid $C_{b,c}$ in G_b **then**
14: n_b forwards the M to n_c
15: **Else if** $TC_{b,c} \geq th_b$ **then**
16: n_b forwards M to n_c
17: **Else** n_b stores and carries M until it encounters another proper node.

Figure 2.9: Successor Selection Algorithm (Xi *et al.*, 2015)

2) Message Selection Algorithm: here also, the identity trust is tested first in order to decide whether a node should accept a message from other nodes for relaying. If n_c receives a forwarding request from n_b, it will first try to verify n_b identity. n_c will collect the message from n_b if $n_b's$ identity can be testified based on the certificate graph. Else, if a valid certificate chain from n_c to the source node can be found in n_c's certificate graph, then n_c will forward the message in order to assist the source.

Else n_c ignores the message. The message selection algorithm is presented as in Figure 2.10 (Xi *et al*., 2015):

1: When n_c receive forwarding request of M from n_b
2: **If** n_c can find a valid $C_{c,b}$ in G_c or $TC_{c,b} \geq th_c$
then
3: n_c stores and carries M hoping to encounter another proper node.
4: **Else if** n_c can find a valid $C_{c,S}$ in G_c , and the validity of M can be testified
then
5: n_c stores and carries hoping to encounter another proper node.
6: **Else if** n_c can find a valid $C_{c,D}$ in G_c
then
7: n_c temporarily holds M for a certain period P_{temp}.
8: **If** n_c can get obtain a valid $C_{c,b}$ or $C_{c,S}$ in P_{temp} after several certificate graph exchanging
then
9: n_c stores and carries M hoping to encounter another proper node.
10: **Else** n_c drop M.
11: **End if**
12: **Else** n_c rejects the related forwarding request from n_b
13: **End if**

Figure 2.10: Message Selection Algorithm (Xi *et al*., 2015)

3) VFP Propagation Algorithm: this is used to ensure that VFP has not expired so as to reflect the recent behavior of nodes. When a node receives a VFP, it will check its validity by examining its expiration date. If it is still valid, then it helps in propagating the VFP. The VFP propagation algorithm is presented as in Figure 2.11 (Xi *et al*., 2015):

1: **For** each node n_i in opportunistic networks

2: **Do**

3: **If** n_i get a forwarding request from a relay node n_j, and n_i accept it by Algorithm 2

4: **then**

n_i generates $VFP_{i,j}$ and propagates it to the following Max_i encounter nodes.

5: **End if**

6: **End for**

7: **While** node n_x receives a $VFP_{i,j}$

8: **Do**

9: **If** $C_{x,i}$ exists in G_i and $LocalTime < Ex_{i,j}$

then

10: n_x testifies the validity of $VFP_{i,j}$ by using the PK_i which can be queried in G_x

11: **If** $VFP_{i,j}$ is valid

then

12: $TC_{x,j}$+1, and n_x forward $VFP_{i,j}$ to following Max_x encounter nodes

13: **End if**

14: **Else** n_x drops $VFP_{i,j}$

15: **End if**

16: **End while**

Figure 2.11: VFP Propagation Algorithm (Xi *et al*., 2015)

Using the three trust algorithms, message confidentiality, integrity and privacy can be ensured. Figure 2.12 is the flowchart of the TMS based on the behavioral feedback for oppnets.

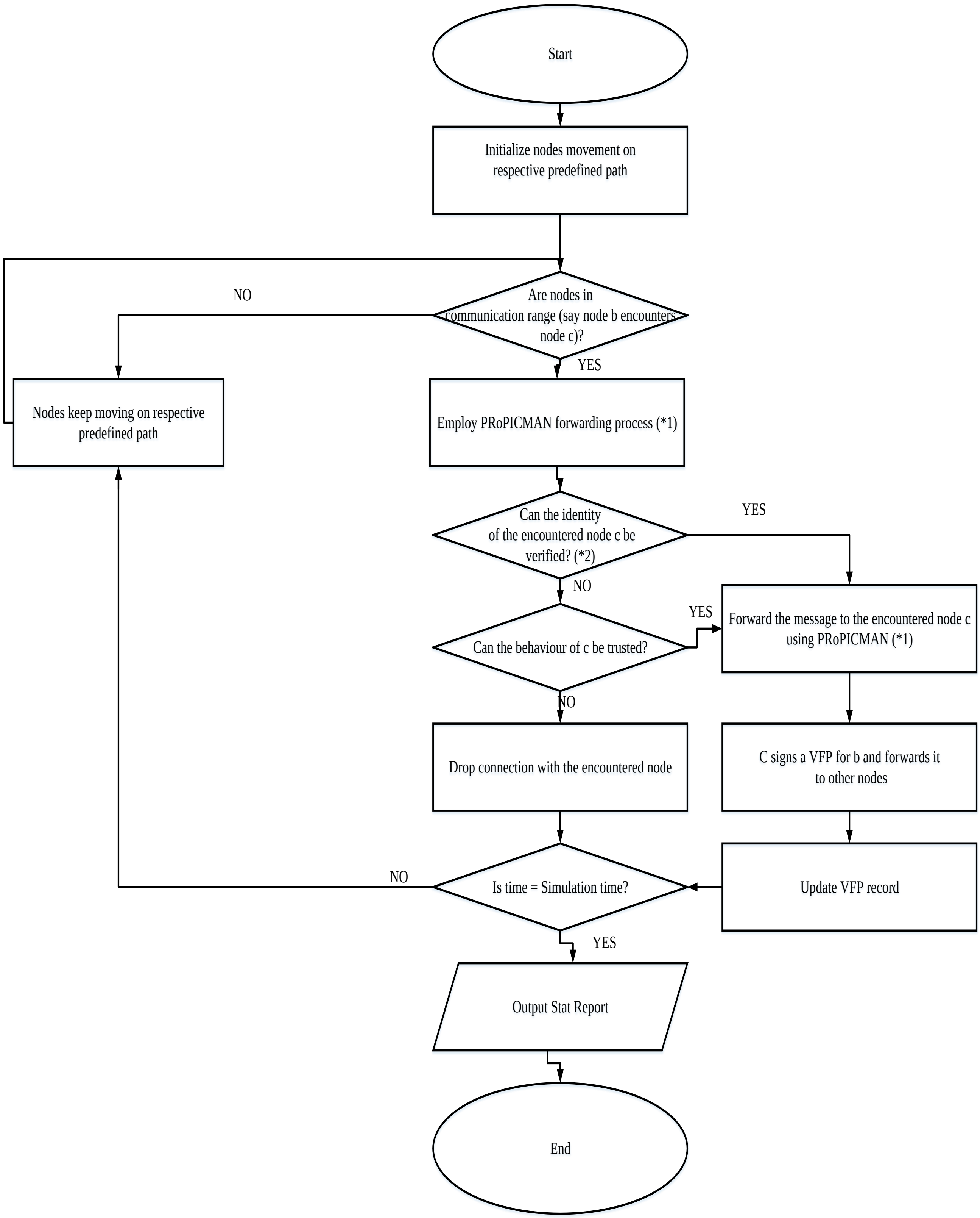

Figure 2.12: Flowchart of the Routing Process with the TMS (Xi *et al*., 2015)

The process labeled *2 (that is, verifying a nodes' identity) is critical to this research as such it is singled and expanded as shown in the flow chart of Figure 2.13.

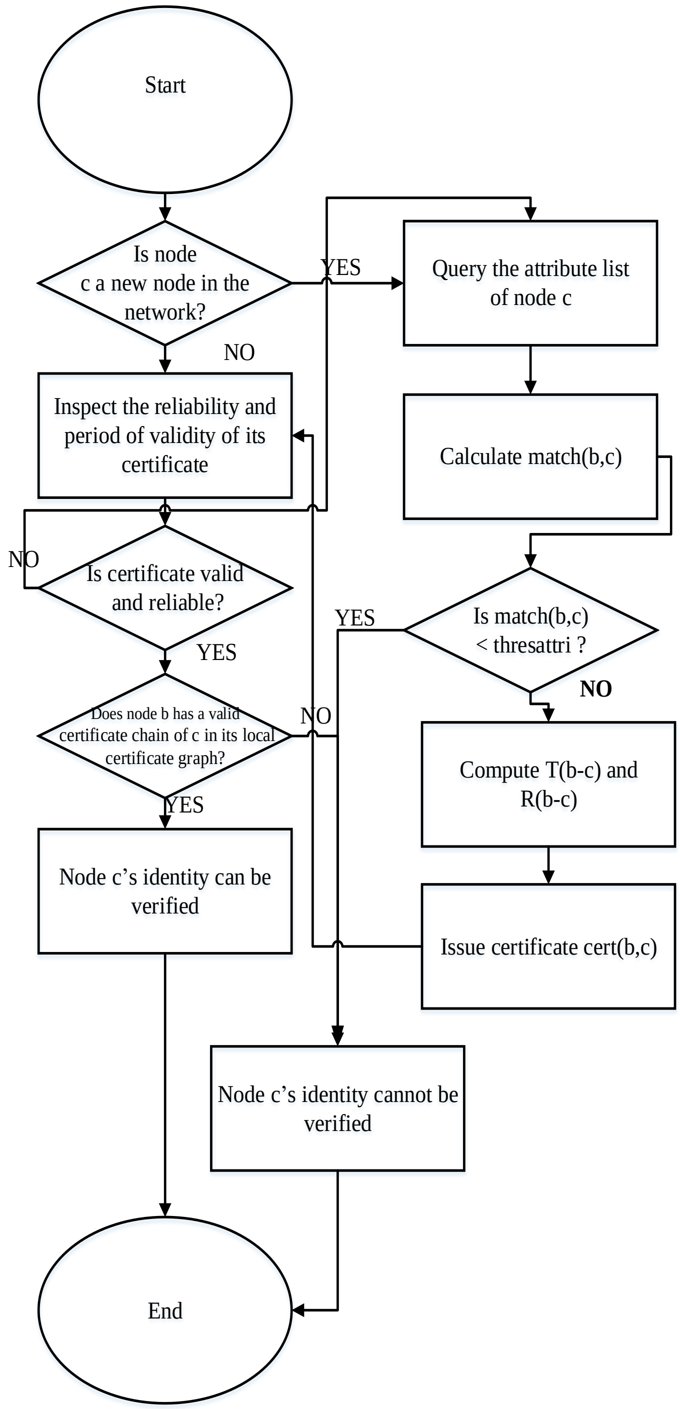

Figure 2.13: Flowchart of the Process of Verifying Nodes' Identity (Xi *et al.*, 2015)

The problems associated with the trust management scheme can be stated as (Trifunovic & Legendre, 2009; Xi *et al*., 2015; Barai & Bhaumik, 2016):

1) Conventional PRoPICMAN was used. The conventional PRoPICMAN forwards message using a maximum of two hop route from the source node to the destination node. Limiting the intermediate nodes to two hops does not scale well in oppnet due to its nature (having ephemeral connectivity, unstable topology, unavailability of context information among others) as a result, this work developed a new variant of the PRoPICMAN called the iPRoPICMAN which would be multi-hop is proposed. This will perform better in oppnet.
2) The threshold value th_i of any given node i was arbitrarily selected. The manner of threshold selection greatly affects the forwarding process since it shares a direct relationship with the forwarding probability of the identity trust. Therefore, selecting an optimal threshold value will improve the privacy, integrity, and confidentiality of the message being forwarded. This guarantees a better security in the network. This work will use the optimum threshold value for the successor selection algorithm.

2.2.12 Evaluating the performance of DTN routing protocols

Many tools have been used to evaluate the performance of DTN routing protocols, mostly including some kind of mobility models built from synthetic mobility model taken from traces of real-life models. Some of which includes (Keranen & Ott, 2009):

1) Diesel Net: this is used to obtain real-world traces which can be made available to other researchers. Its flexibility is limited and large-scale operation is expensive. Diesel net is typically limited to those running test-beds.

2) The CRAWDAD project: this provides a repository where numerous real-world traces are available. These traces give insight into the real-world interaction between mobile users from different angles and make a valuable data source for mobility and connectivity characteristics obtained from synthetic model. However, real-life traces have their own limitations since the population analyzed in the traces is limited, hence, the result is biased. Also the time granularity is small in order not to drain the mobile device battery quickly and results cannot be arbitrarily scaled-up thus, limiting what can be evaluated.
3) TRANSIMS allows modeling a population and how they interact at certain locations or in public vehicles. It does not include details on the path between such locations thus, limiting its suitability for generating mobility data for pedestrians.
4) ONE Simulator: a number of test-beds, real-world traces, tools or simulation models address only a particular subset of users. Traces and test-beds are limited in their coverage and granularity. Vehicular mobility models do not consider pedestrians, and constraints on maps are often not taken into account. Also the models do not allow the incremental addition of further features to stepwise increase the degree of reality of the simulation. The ONE simulator contributes an environment for DTN protocol evaluation, which has internal and external mobility models, different DTN routing schemes, interactive inspection as well as post-processing. ONE allows the incremental addition of further features to stepwise increase the degree of reality of the simulation.

   Another useful tool in the research is MATLAB.

2.2.13 ONE simulator

ONE is a customizable java-based simulator targeted for research in DTN. It can import mobility data from real world traces or other mobility generators. It can also produce a variety of reports from node movement to messages passing. The simulator is written in Java (1.6), it allows a user to simulate different scenario quickly and in a flexible manner. It also has the provision to be able to generate statistical results (for example, delivery probability, overhead ratio, number of hops, delay, simulation time, packets created, dropped and delivered and so on.) from the simulation performed. It can be run on Linux, Windows or any other platform supporting java. ONE is capable of (Keranen & Ott, 2009; Benamar *et al*., 2014):

1) Generating node movement using different models.
2) Routing messages between nodes with various DTN routing algorithm.
3) Visualizing both mobility and message passing in real time in its graphical user interface (GUI). Figure 2.14 shows an overview of ONE simulator GUI.

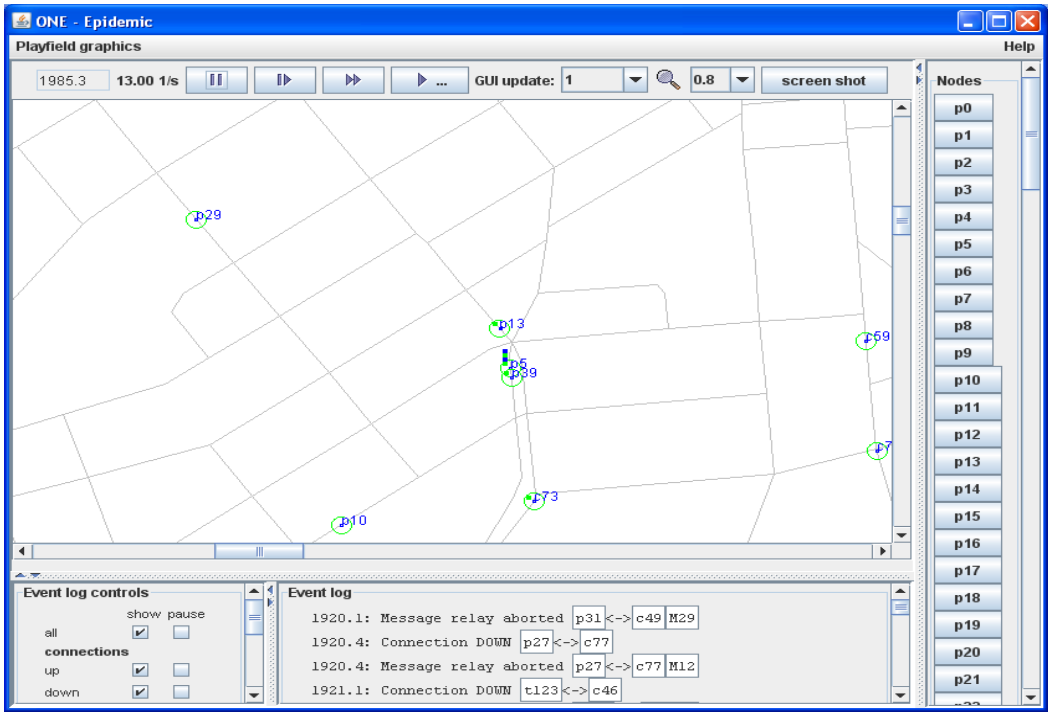

Figure 2.14: Overview of the ONE Simulator GUI (Keranen & Ott, 2009)

Several built-in routing modules are available in ONE (that is, direct forwarding, first contact, spray and wait, prophet, maxprop, epidemic), also external simulators such as NS2 can be used. For external programs, a separate parser is needed to convert message events to a form the ONE understands. ONE is able to visualize results of the simulation in two ways (Keranen & Ott, 2009):

1) Showing the simulation in real time within the graphic user interface with node locations, current paths, connections among nodes, number of messages carried by a node.
2) Using a map-based movement model, showing all map paths in the GUI.

The whole simulation can be configured to pause when an event occurs and a single node can be selected for further information about the message a node is carrying and about routing modules state (Keranen & Ott, 2009).

2.2.14 Performance metrics

The performance metrics used in running simulations include (Aneja & Garg 2015):

1) **Delivery probability**: Refers to total probability of the messages delivery. It is gotten by taken the ratio of the packets created to the packets successfully delivered to their final destination. A high delivery probability implies that more packets are delivered to the destination and vice versa. A routing protocol with a higher delivery probability performs better than a routing protocol with a lower delivery probability.

2) **Delay:** This is called the average latency. It is the average value for time for each successfully delivered packets. A good routing protocol is expected to incur a low delay.

3) **Overhead Ratio:** This is the amount of system resources (CPU, memory, bandwidth) needed to deliver a packet to its final destination. A good routing protocol is expected to incur a low overhead ratio, implying that it can deliver a packet using less system resources. The ONE simulator has been pre-configured to generate this value by default.

2.2.15 Helsinki simulation area

The Helsinki simulation area is a region in Finland. Researches in oppnets use the Helsinki region to validate results due to the fact that the oppnet environment simulator was originally developed on the region, an example is a research carried out by Verma and Srivastava (2012), Keranen and Ott (2007), Keranen and Ott (2009), etc.

A research carried out by Moreira, *et al*. (2011) showed that comparing the performance of routing protocol using different conditions, parameters, metrics, and scenarios could result to an assessment that is biased (Moreira, *et al.*, 2011). This is because the performance metrics of a given scenario may favour a given model. As such, most work uses the Helsinki simulation region for their simulation to create basis for comparing their results with that of others. A part of the Helsinki downtown area (8300 x 7300m) was used in the ONE simulator. The region is shown in Figure 2.15 while Figure 2.16 shows the extracted line map of the region. Nodes follow a predefined path and communicate in a transitive manner. Humans (pedestrians), car and trams are the mobile nodes moving in the terrain.

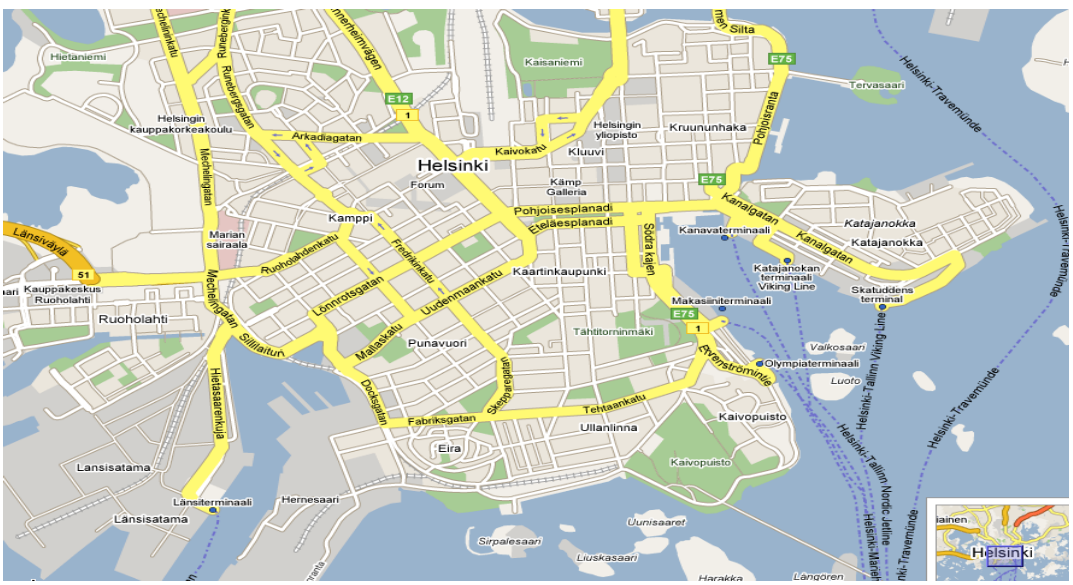

Figure 2.15: Helsinki Simulation Area (Keranen and Ott, 2007)

Figure 2.16: Line Map of the Helsinki Simulation Region (Keranen and Ott, 2009)

1029 and 544 mobile nodes are referred to as large and small scenarios respectively. Interpersonal communication between mobile users in the city using modern mobile phones or similar devices is assumed while 10m radio range bluetooth at 2Mbit/s net data rate is used. It is also assumed that the mobile devices have up to 100MB of free buffer space for storing and forwarding messages since flash memories are mostly occupied by music, photos and videos (Keranen and Ott, 2009).

2.2.16: Ahmadu Bello University (Main Campus) Zaria

ABU main campus is a region in Samaru district of Kaduna state in Nigeria. This region as depicted in Figure 2.17 will be modeled in the ONE simulator. Road paths are of interest to this research since nodes (mobile phone, PDA, etc.) are carried by people who move along this path either in cars, tri-cycles, motorcycles, bicycles or as pedestrians. Pedestrians do follow other paths apart from the roads. The number of nodes were obtained from a detailed survey of movement patterns in the region, information from the security unit as well as information from the Management Information System (MIS) of the university. The area has a population of about 64,000 people consisting of staff and students. An average of 4000 cars pass through the area between 6am and 6pm. The total number of tricycles are 11, while the total number of shuttle buses are 4.

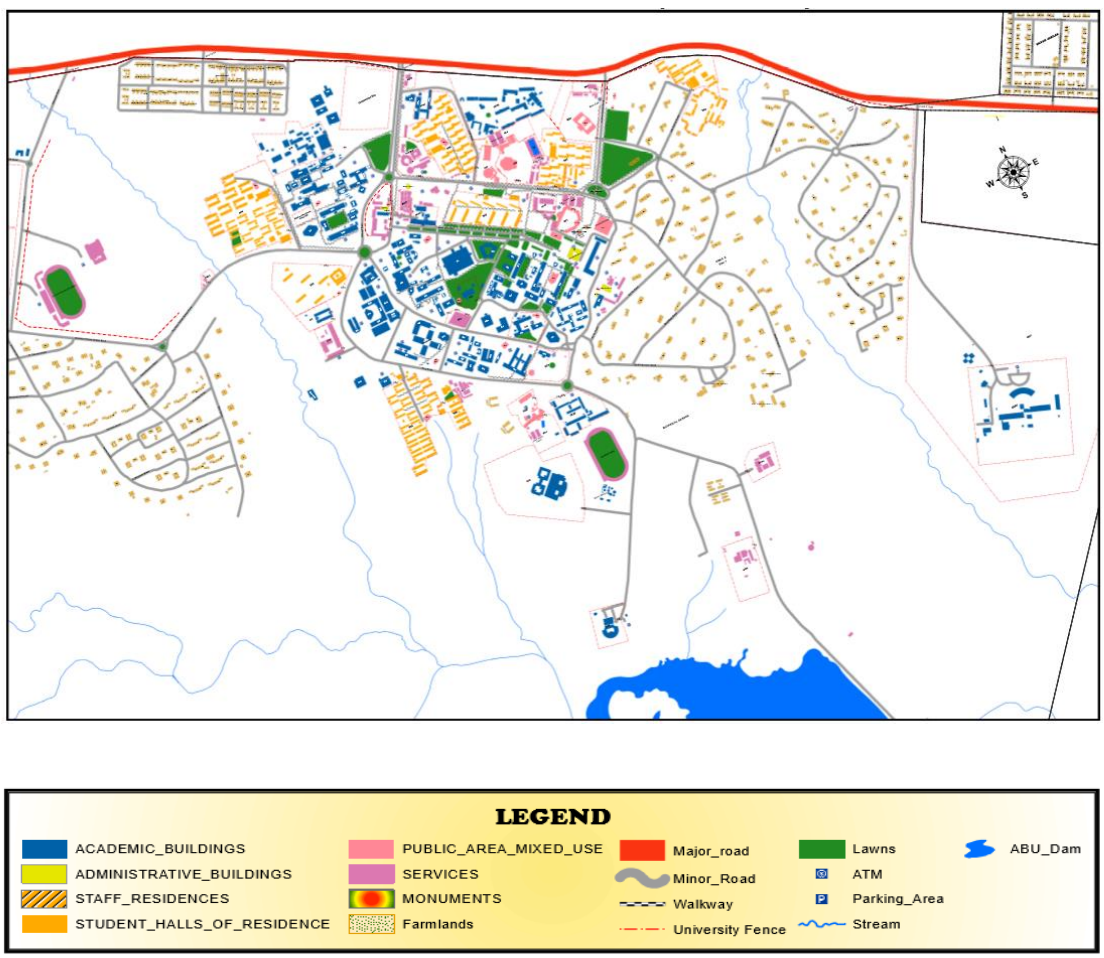

Figure 2.17: Map of ABU Main Campus Simulation Area

2.3 Review of Similar Works

Several research works have been carried out on security issues and trust management schemes with respect to oppnet. A proper review of these works will clearly show the state of art of research in this area as well as the research gap which should be closed. Some of the pertinent literatures are reviewed as follows:

Lilien *et al.* (2006) were the first to introduce a new paradigm and technology called "Opportunistic Network". They were the first to define and investigate opportunistic networks where they pointed out its possible challenges (identity spoofing, Denial of Service (DoS) attacks by malicious devices etc.). They also suggested intrusion detection as security measure since malicious devices or networks are likely to join opportunistic networks because of lack of initial authentication mechanism. The researchers did not show the architecture of the oppnet nor did they propose a routing protocol for the network which would have aided the understanding of their newly introduced oppnet better.

Kaur and Kaur (2009) surveyed the different routing protocol in the oppnets as well as the security issues in oppnets. They pointed out that the main source of privacy and security threat in oppnet was due to the fact that authentication could not be established when devices join the oppnet. The survey also pointed out that it was difficult to deliver secret keys securely to all non-malicious devices in an ad hoc environment. As such, relying on cryptography-based authentication would not help in the oppnet. They also classified a taxonomy for the routing protocols of the oppnet, where they divided them into infrastructure-based and infrastructure-less. The infrastructure-based was further classified as fixed and mobile infrastructure while the later was split into context-based and dissemination-based. They proposed an intrusion detector as the solution to security problem in oppnets. However, the intrusion detector is not a good security check in the oppnet due to its heterogeneous nature and nature of security threats, which affected the efficacy of their model.

Trifunovic and Legendre (2009) argued that security in a distributed identity management system (such as oppnets) could not be achieved by cryptographic means and strongly recommended the use of trust models. They discussed security issues, complexity, and trust propagation characteristics. They established that reputation is trust based on direct interaction and presented three ways of establishing trust (social, environmental and similarity-based trust). They presented algorithms for the trust models and recommended that trust values should be combined in order to eliminate their disadvantages and to yield a better-composite trust method. However, their trust algorithm was mainly against Sybil attack. Also, they considered only environmental and social forms of trust in selecting their threshold trust value which affected the robustness of the threshold selected.

Trifunovic *et al*. (2010) presented two complementary approaches for the establishment of social trust (implicit social trust and the explicit social trust) and how they could be combined in order to obtain a consolidated social trust value. They used mobility traces, synthetic mobility model and real world complex graphs to evaluate the complexity, trust propagation and security issues of their social trust. It was shown that their model limited the number of Sybil attackers irrespective of the network size, and their model was more robust against random attacks as compared to the other approaches (i.e. certificate chains and distribution communication detection algorithms). However, they did not include the consolidated social trust in the performance evaluation in order to check its performance. They, also, did not consider trust threshold in both the implicit and explicit trust models, which affected the general performance of their social trust model.

Shikfa *et al.* (2010) developed a novel security scheme by combining the identity-based cryptography and policy-based encryption to improve privacy in opportunistic routing process. Security issues were surveyed in oppnet where the problems of cooperation enforcement and secure routing were analyzed. They also suggested means by which privacy could be enhanced in both context-based and content-based routing. The scheme included a trusted third party (TTP) that provided the key generation server for all the nodes in order to protect them from the Sybil attack (a form of attack that malicious nodes create a number of fake IDs). However, the TTP was difficult to realize in the oppnets. This was only applicable to the context-based oppnet, where the destination of the TTP that was in charge of providing each intermediate nodes with trapdoors corresponding to their context is replaced. The TTP need only to be contacted once before nodes join the network and is offline during the network operation, which is suitable for opportunistic networks. Furthermore, this solution allows intermediate nodes to compute the matching ratio between their context and the destination's context and to forward the message to nodes that show increasing match, thus enabling privacy preserving context-based forwarding. In content-based communication there is a complete decoupling between sender and receiver: intermediate nodes build their routing tables based on the interests advertised by receivers. These interests are private information and it is therefore important to guarantee the confidentiality of advertisements while still enabling intermediate nodes to build their routing tables. This issue is very different from the previous one because contrary to context, interests are not intrinsically linked to a node and they change frequently. The problem is therefore to enable intermediate nodes to build routing tables with encrypted interests and to perform secure look-up of encrypted content in the routing tables.

Chen *et al.* (2011) analyzed and compared the performance of trust-based routing protocols of a stochastic Petri-net based DTN model consisting of heterogeneous (different social and network characteristics) mobile nodes with connectivity-based, epidemic and PRoPHET routing protocols in terms of message delivery ratio, message delay and message overhead. Connectivity, which was used to measure the QoS trust level of nodes was combined with honesty and unselfishness, which were used to measure the social trust level of nodes into a composite trust for decision making in dynamic routing in DTN. Each node was associated with a trust value that was calculated and arranged using different weights. The results showed that the developed trust-based routing protocols outperformed PRoPHET for all the metrics and had similar performance as the epidemic routing in delivery ratio and message delay but with lower message overhead. The integrated social and QoS trust-based protocols outperformed the connectivity-based routing protocols with respect to message delivery ratio and message overhead but at the expense of message delay.

Chang *et al*. (2011) introduced a trust-threshold-based routing in DTN called (TTBR). Trust-threshold-based routing leverages two trust thresholds (recommender threshold and message forwarding threshold) for accepting recommendations and choosing the next forwarder of the message. TTBR uses two form of trust relationships (social trust and QoS trust). Healthiness and cooperativeness were the measures considered for social trust, while connectivity and energy were considered for QoS trust. Three types of nodes were defined (good, uncooperative and malicious) for simulation using the NS3 simulator. A cooperative analysis carried out with epidermic, social-trust-based and QoS trust-based routing using delivery probability, message delay and number of copies propagated per message showed that TTBR effectively traded message delay and message overhead for higher delivery probability. However, they did not benchmark the TTBR with prominent trust algorithms in oppnets in order to evaluate its effectiveness.

Poonguzharselvi and Vetriselvi (2012) developed a trust framework for data forwarding in opportunistic networks using mobile traces. Next hop selection was based on the trust value as well as the direction of movement of nodes towards the destination. The trace file of the movement of the nodes in the network was maintained by the nodes and was used to determine the direction of the destination. Messages were encrypted to secure both the message and the path information. The trust framework was analyzed using different number of malicious nodes in the network and simulated using the ONE simulator. Results showed that the routing performance was improved (higher delivery probability and lower overhead) when compared with those obtained using the routing algorithm without any trust framework. However, the application of trust values presented a unique challenge in opportunistic routing in terms of selection of an appropriate trust threshold as it affected the delivery probability and overhead ratio.

Goyal and Chaudhary (2013) devised a means of ensuring privacy in the opportunistic network. They introduced the concept of opportunistic network, then differentiated it from MANETs. They presented the attacks in opportunistic network as worms and viruses attacks. They presented a table that summarized works done on routing protocols and security in the opportunistic network. They proposed a novel algorithm to maintain privacy in opportunistic network using cluster estimation. The algorithm relied on a single node in each cluster (called the trusted node) to get a virtual ID before communication can take place. Also, the trusted nodes authenticate other nodes using cryptography. The question is what if the battery of trusted nodes runs down. NS-2 simulator was used to run simulations and result showed that the model provided privacy to user and reduced packet loss carried by selfish nodes.

However, the use of cryptographic means is not considered appropriate for oppnets as they require computational intensive operations which most oppnet nodes cannot handle.

Chen *et al.* (2013) designed a dynamic trust management for secured routing in DTN. The forms of attacks considered were self-promoting attacks, bad routing attack, ballot stuffing as well as random attacks. They obtained a composite trust metric by combining QoS and social trust. They were the first to introduce notion of subjective versus objective trust. They presented a novel method (using stochastic Petri Net (SPN) technique) of analyzing their trust model and they used NS-3 simulator to validate the trust model using simulation. They compared their trust model with other routing protocols (Bayesian trust-based routing, non-trust-based PRoPHET and epidemic protocols). Their trust model consistently outperformed the PRoPHET and Bayesian trust model (with a higher delivery probability and number of copies propagated per message and a lower delay). However, their trust based model had lower delivery probability, lower number of copies propagated and higher delay than the epidemic routing protocol.

Gupta *et al.* (2013) developed a trust-based security protocol (TSP) to reduce blackhole attacks in oppnets. The TSP relies on the calculation of social group value (SGV) and trust distribution technique. They implemented the TSP on the PRoPHET routing protocol and stimulated it using the ONE stimulator. On comparison between TSP and the PRoPHET under blackhole attack (PBH), significant improvement was archived in terms of lower dropped messages, and a higher delivery ratio. However, they did not consider a dynamic function in calculating the SGV of randomized malicious behavior of nodes. Also, they did not optimally select a trust threshold value for their protocol which has been shown to affect the effectiveness of any trust-based scheme.

Singh and Chawla (2014) presented a trust management technique for DTN using a dynamic threshold updating scheme for changing network conditions. They evaluated the model using different types of selfishness (barter-based, credit-based and reputation-based) as well as different number of malicious nodes via simulations using the ONE simulator. The dynamic threshold update scheme reduced the false detection probability of the malicious nodes and the overheads and increased the efficiency of the routing network. When compared with previous routing protocols, it was shown that the technique handled effectively the selfish behavior by providing significant gain on the delivery ratio but this was at the expense of message overhead and delay.

Wu *et al*. (2015) assessed the challenges faced by oppnets as a result of their uncontrollable nature due to the possibility of opportunistic communication, they identified and described seven prominent security threats as unauthorized access, message tampering, injection attack, resource depletion attack, denial of Service (DoS), confidentiality disclosure and privacy invasion. As a result of these, it was suggested that protocol design and network deployment for oppnets should critically consider authentication, authorization and access control, data confidentiality and integrity, and privacy protection, trust management and cooperation. As such, a five-module based generalized security architecture consisting of authentication, secure routing, access control, trust management and cooperation, and application/user-specific privacy protection was developed with the caveat that only the required modules may be implemented depending on the specific user or network requirements and/or security threats. It was concluded that the focus of further research should be on trust management and cooperation. However, it is required that more testbeds be developed for validation of security and trust protocols in oppnets.

Alajeely *et al.* (2015) classified the security threats or attacks that oppnets are susceptible to as either internal or external and determined that the success of security mechanisms (authentication, integrity, confidentiality, access control, availability, and non-repudiation) required to overcome these threats is highly dependent on trust between people (since oppnets strongly depend on human interaction). Internal attacks are generated from within the oppnets and are generally more impactful as some of the malicious nodes are privy to some critical network information or resources. External attacks are generated from outside the network and are generally less impactful than the internal attacks as the malicious nodes may have little or no access to network information or resources. The attacks assessed in the work included wormhole attacks, black hole attacks, selfish attacks, Sybil attacks and selective dropping attacks and these attacks are aided by the ease at which node locations are detected (as this makes it easier to track the path and movement of nodes in the oppnets). The anti-localization anonymous routing (ALAR) technique was suggested as a means of enhancing the sender's location privacy while maximizing message delivery. However, the ALAR technique was limited by its routing performance, which can be degraded in terms of delivery ratio and delivery latency depending on the number of segments and number of neighbours or receivers selected. Another suggested approach for increasing security in oppnets is the intrusion detection system (IDS) in which nodes observe traffic sent by one-hop neighbours and then compare, using some metrics like packet dropping ratio, with some observed historical values in order to detect abnormal behaviour. However, the nodes in oppnets using the IDS mechanism process all the observed packets since they are in a promiscuous node and this is not energy efficient. The IDS is also affected by the available number of neighbours, especially if they are not enough to adequately observe packets at all the nodes. The final strategy suggested was the trust strategy, which required that a node's identity be authenticated in order to avoid malicious nodes and this was classified as social, reputation and history (or a combination of all three). All these classifications were further divided into direct and indirect. Some of the mechanisms described for trust measurement included rating experiences, reputation models, recommendations, past experiences, word of mouth, weighting, probability, encounter-based trust, voting methods, cluster-based methods, honey bee mating, beacon nodes, public key cryptography, digital signatures, Bayesian networks, chain optimization, social networks, iterative algorithms, acknowledgements, watchdog monitoring nodes, neural networks, game theory, fuzzy logic, confident and core, swarm intelligence, and directed and undirected graphs. However, the performance of all these mechanisms is dependent on the trust value threshold and as such its selection is very critical for the trust based approach

Xi *et al*. (2015) developed a novel trust management scheme based on behavioral feedback to support forwarding protocols in order to address the limitations of identity authentications in extremely sparse oppnets. The trust management scheme considered the assumptions of each node knowing the social context information of others, each node having a limited buffer for messages and certificate graph, and each node being able to communicate with each other if only they are within a certain transmission range. Verified feedback packet (VFP) feedback system and hierarchical trust were utilized in order to guarantee the security and efficiency for the forwarding protocols and in order to overcome the possibility of certificate collision, the reliability value for each node in the oppnet is calculated using the feedback information. The forwarding protocol is critical in ensuring improved network performance. The developed scheme was implemented into PRoPICMAN (social context-based) routing protocol and emulated in the ONE simulator. It is suggested that the developed scheme is the first ever attempt at providing trust management scheme for forwarding protocols in oppnets. However, the threshold value th_i, which is a vital parameter in the success selection algorithm used in the developed scheme, is not optimally determined and its value affected the success probability of building behaviour trust.

Barai and Bhaumik (2016) undertook a survey of some security techniques being used by oppnets that are broadly classified as cryptography based, non-cryptography based or trust based. The cryptography based approaches required computationally intensive operations to be carried out using powerful hardware in order to attain the required level of protection. This, however, makes the cryptographic based protocols unsuitable for oppnets containing mobile devices (since they are meant to be portable and energy efficient). The mobile nodes in oppnets interact in a distributed way with each other and this is an issue as there is no trusted third party (TTP) to generate keys (private/public) and certificates required by most of the cryptography-based schemes. In order to protect the data forwarding process in oppnets, the non-cryptography based protocols utilized the reputation system. However, this is difficult to implement in oppnets since it was assumed that pre-existing key management is inbuilt in the reputation system in order to facilitate the ease at which each of the nodes obtained their private/public key pairs. Trust based protocols depend on the determination of a trust value or computational trust which is a measure of the trustworthiness of a node in the network and the basic framework for calculating it. The trust value is determined for the segment of the oppnet that is relevant to a node and as such it is possible to implement trust based path for delivering messages. However, the need for a robust trust metric is vital in an oppnet since it is possible for a single user to generate multiple identities, as there is no central authority.

Padhi *et al*. (2016) used cyber foraging based on trusted cloudlets (A cloudlet is a mobility-enhanced small-scalable cloud data center that is located at the edge of the internet. It arises from the convergence of Internet of Things (IoS) and cloud computing in order to support resource-intensive and interactive mobile applications by providing powerful computing resources to mobile devices with low latency) to improve security of mobile nodes in opportunistic mobile nodes. They used cloudlets as trusted access point and data exchange point for oppnet nodes since it acted as a black box which do not compromise the security model of the network. They also designed a key management module, data storage module for the cloudlets and a cryptosystem which hybridized both symmetric and asymmetric key encryption system. They analyzed several attacks for their proposed model. However, the attacks presented did not portray the nature of attacks in oppnets. They are more prominent in conventional networks. Also, their method of dealing with security (cryptography) is not effective in oppnets. This is because serious cryptosystem cannot be realizable in oppnet due to its requirement for complex and computationally intensive operations which most oppnet nodes cannot handle.

Kungwani and Dudhe (2016), proposed a trust based privacy preserving friend recommendation as a security scheme in social networks. They used K nearest neighbor (KNN) algorithm as the running convention in determining the scrambled social direction coordinating results. The security and possibility of the plan was demonstrated through security examination and test assessment. However, they did not demonstrate the practicability of this model to any particular network.

Yao *et al.* (2016) developed a trust management scheme for optimized secure routing by integrating social trust into the routing decision process called the trust routing method based on social similarity (TRSS) for oppnets. TRSS used both routing behaviors and social similarity to obtain a complete trust metric. Routing performance in oppnets is expected to be improved by nodes with social relationship (direct and indirect) fully cooperating with each other but then this creates the possibility of them being susceptible to attacks (especially if the routing protocols are not secured). This made it necessary to develop a strategy for detecting and isolating selfish and malicious nodes from the well-behaving ones. TRSS has been found to be quite effective in detecting and purging malicious nodes from the trusted list while providing the requisite incentives for selfish nodes to behave properly by ensuring that only packets from trusted nodes are forwarded. The performance of the TRSS was compared with that of three other oppnet related protocols (non-trust based protocol (PRoPHET) and social-aware routing algorithm (dLife)) for their ability to protect the oppnets from such attacks as promise–then–drop, trust–boosting and defamation) using ONE and Infocom 2006 trace simulators. The comparison was carried out using metrics such as delivery ratio, number of duplicated packets and delivery latency. Results showed that the TRSS outperformed the others in not only accurately detecting the selfish or malicious nodes but also improving the delivery performance even in the presence of the malicious nodes. However, the TRSS is premised on the fact that users were willing to share their social features, which is not always the case in real life due to privacy related issues and as such the scheme will not perform optimally where there is unwillingness for information sharing.

Khot and Mogal (2017) designed a trust preservation algorithm as a trust based mechanism for misbehavior detection of nodes in oppnets using Eclipse and Java for implementation in

ONE. They considered individual selfishness, social selfishness and malicious nodes in their design and utilized two of the available routing protocols (Epidemic and PRoPHET) in ONE. The routing modules determined how messages generated are forwarded over existing contacts. Encounter history, delegation history and forward history were the three metrics used to assess the trust model. The results showed that the developed approach was effective in misbehavior detection with low transmission and signature verification cost. However, the messages generated were only unicast, having single source and single destination.

Trifunovic *et al.* (2017) carried out a survey on research challenges in oppnets. The study showed that after a decade of research in the field of the oppnet, most research questions have been answered but privacy and security still remain a problem. The survey invited researchers to work on security in opportunistic network, especially in emerging application like mobile sensing and opportunistic computing.

Based on the presented reviews and their several shortcomings, it is clear that the security aspect of the oppnet has not been well explored and implemented (Barai & Bhaumik, 2016). Security in oppnets is critical since they are prone to malicious attacks due to their nature (absence of initial authentication and lack of end-to-end connectivity, unstable structure). These malicious devices are capable of posing serious challenges that can affect the privacy, confidentiality and integrity of data, and also, the performance of the network in the long run. There is no concise literature that dealt with all security issues in a formal manner (Barai & Bhaumik, 2016). Different attempts to address security issues in oppnets have shown that trust management is the preferred method (Xi *et al*., 2015; Trifunovic & Legendre, 2009; Barai & Bhaumik, 2016; Wu *et al*., 2015). However, most trust models are developed using trust threshold value selection approaches that are not optimal (Xi *et al*., 2015;

Poonguzharselvi and Vetriselvi, 2012). It has been shown that the trust threshold value is critical to oppnets since it holds a direct relationship with the behavioural trust of the forwarding process (Barai & Bhaumik, 2016; Xi *et al*., 2015) and as such this work focuses on developing an eTMS using an optimally determined trust threshold value for the successor selection algorithm of the conventional TMS. This is implemented on an improved PRoPICMAN model of Xi, *et al*., (2015).

CHAPTER THREE

MATERIALS AND METHODS

3.1 Introduction

This chapter describes the detailed procedure carried out in setting and modelling the ONE simulation environment, replicating the PRoPICMAN and developing the iPRoPICMAN into the ONE simulator, developing the TMS and the eTMS and their simulations. The process of modeling and simulation of the new ABU simulation environment had also been presented here.

3.2 Materials

The materials used in the research includes; ONE simulator**,** integrated development environment, the java development kit, the java runtime environment, OpenJUMP software, the ABU map and movement model information and MATLAB.

3.3 Setting of the Simulation Environment

The materials needed in order to have the ONE simulator up and working includes java development kit (jdk), java runtime environment (jre), ONE simulator setup and an integrated development environment (IDE). While the methodology involved includes:

1) Downloading and installing the java development kit (jdk) and java runtime environment (jre)

   Both jdk and jre are needed in order to write and compile (transfer codes from human understandable form to codes computer can understand) java codes since the ONE simulator is java-based. The executable files of Java SE are downloaded from *http://www.oracle.com/java/javase/downloads/index.html* and installed accordingly.

2) Setting the java path

Even after successfully installing the jdk, the computer cannot access the compiler. This can be tested using the *javac* command on the command prompt. As such, the computer (through its advanced setting) is told where to get the compiler as shown in Figure 3.1. The compiler's location is gotten from,

This PC → Windows (C) → Program files → java → jdk1.8.0_111→ bin.

The address there is then copied and taken to the location:

System property → Advanced System Settings → Environment Variables → System Variables → New.

The variable name is set as "path"

The bin location is set as the variable value.

By so doing, the Java path has been set and the computer should be able to recognize java codes, and to be more precise, the ONE simulator. This can be verified using the *javac* command on the command prompt again.

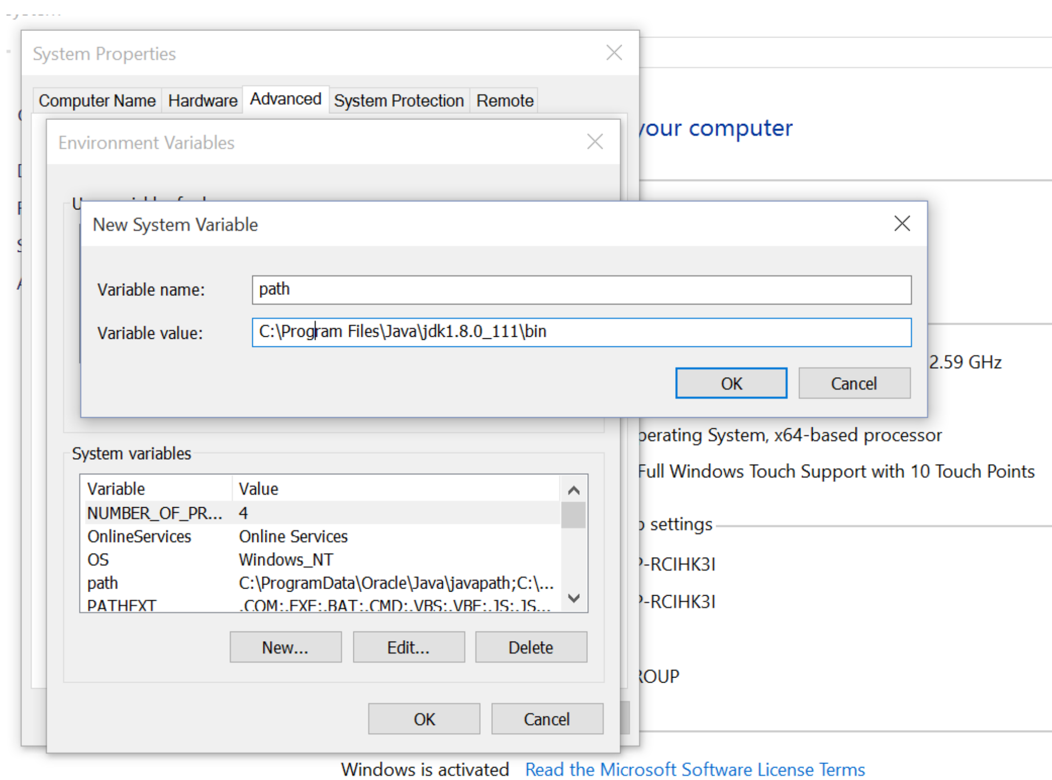

Figure 3.1: Setting Java Path

3) Downloading and installing the ONE simulator

A compressed source code of ONE (version "ONE_1.5.1.tar.gz") was downloaded from:

http://www.netlab.tkk.fi/tutkimus/dtn/theone/down/one_1.6.0.tar.gz.

The source code was then uncompressed by running the compile in the extracted ONE and the folder \one 1.6.0 was extracted and saved in a desired location. This is compiled into the IDE.

4) Downloading and installing an integrated development environment (IDE).

The IDE used to debug and run the ONE simulator is downloaded from http://www.eclipse.org/downloads/. Another option is the 'Netbeans' which can be downloaded from https://netbeans.org/community/releases/73/index.html. The reason for using Eclipse is that it is more user-friendly than Netbeans. Netbeans also utilizes more computer resources and its running speed is slower than the Eclipse. The downloaded Eclipse IDE was uncompressed and packaged to a folder in order to run the application.

5) Interfacing the ONE simulator with the IDE

   The ONE simulator source codes were imported into the eclipse by first creating a workspace in the eclipse. This work space has to be ONE's computer name. After selecting a new Java project in the file menu of Eclipse, "default location" is unselected and then browsed to the ONE folder and opened.

Two important jars were imported for the ONE to compile, that is; *DTNConsoleConnected.jar* and *ECLA.jar*. '*DTNConsoleConnector*' jar offers a way for DTN simulator to use commands to communicate with DTN simulated scenarios. This enables information to be managed and obtained from the console instead of only from the output files. As for *ECLA.jar* file, it is used to simulate network interface hardware and it tries to finish the work in the link layer. *junit.jar* is used to correct any configuration error in the simulation process. This jars are imported by browsing to the ONE folder from the eclipse as shown in Figure 3.2 using this steps;

From the eclipse, click file → new → java project → unselect default location → browse to the ONE folder and open it → next → libraries → Add external jars → browse to the ONE folder →

lib → open two folders and import jars → click finish.

At this point, the ONE simulator can be ran using Eclipse.

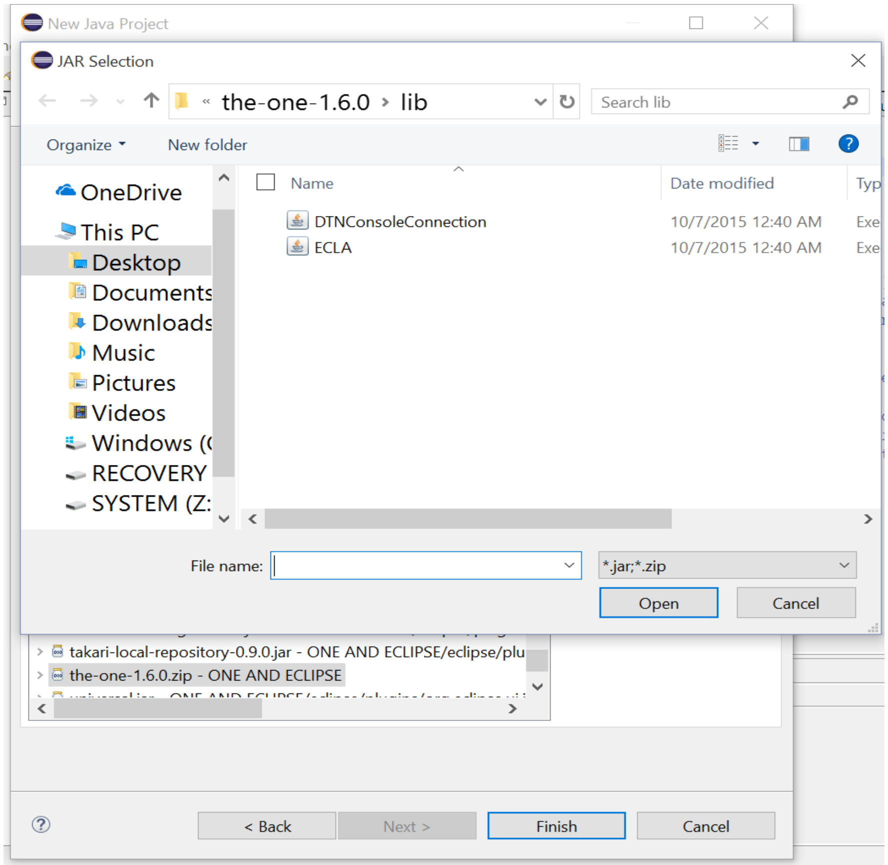

Figure 3.2: Interfacing the ONE simulator with the IDE

3.4 Modeling the Simulation Environment

A research carried out by Moreira, *et al*., (2011) showed that comparing the performance of routing protocols using different conditions, parameters, metrics, and scenarios could result to an assessment that is biased (Moreira, *et al*., 2011). This is because the performance metrics of a given scenario may favour a given model. As a result, the same simulation scenario as that of the TMS is used in this research in order to create basis for comparison with the eTMS. The TMS and eTMS are implemented in the iPRoPICMAN routing protocol using the ONE simulator. A part of the Helsinki city was used in the simulation covering $3800{\times}3000m^2$. This is depicted in Figure 2.16. The following are the steps:

1) **Defining the types of nodes**

A hundred (100) nodes were used in order to emulate an extremely sparse environment. Twenty-five nodes needed permission by the trusted initial nodes in order to access to the network (to get the valid public key). Communication is assumed to be between modern mobile phones and similar devices capable of serving as oppnet nodes. Users holding these devices are defined as pedestrians and people in cars/buses.

2) **Defining the characteristic of nodes (speed, transmission range, event generation interval, buffer space)**

The speed of nodes was determined by the category of users holding it. 0.8–1.6m/s was used for walking speed of nodes (i.e. pedestrians) and 8 -12m/s was used for cars and buses. Bluetooth data rate of 250kB was used. The normal Bluetooth transmission range of about 10m was assumed. The buffer size of nodes was considered to be 30MB and the message size was between 500kB-2MB. The event generation range was assumed to be 25-35s. The simulation time was taken to be 43200s which is the default time for simulation of the Helsinki region on the ONE simulator.

These settings are implemented at the default settings of the ONE simulator to come up with a scenario-specific setting. The program code is as shown in Figure 3.3.

```
## Scenario settings

Scenario.name = PRoPICMAN_scenario

Scenario.simulateConnections = true

Scenario.updateInterval = 0.1

# 43200s == 12h
```

```
Scenario.endTime = 43200

## 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

# Define 2 different node groups

Scenario.nrofHostGroups = 2

# Common settings for all groups

Group.router = PropicmanRouter

Group.bufferSize = 30MB

Group.waitTime = 0, 120

# All nodes have the bluetooth interface

Group.nrofInterfaces = 1

Group.interface1 = btInterface

# Message TTL of 300 minutes (5 hours)

Group.msgTtl = 300

Group.nrofHosts = 100

# group1 (pedestrians) specific settings

Group1.groupID = p

Group2.speed = 0.8, 1.6

# group2 specific settings
```

```
Group2.groupID = c
# cars can drive only on roads
Group2.okMaps = 1
Group2.speed = 8.0, 12.0
#
## 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 - 2MB)
Events1.size = 500k,2M
# range of message source/destination addresses
Events1.hosts = 0,125
# Message ID prefix
Events1.prefix = M
```

Figure 3.3: Simulation Scenario Settings

The ONE simulator generates statistical results from the simulation performed when configured to do so This scenario has also been configured to output the statistical results of the simulations performed in the Message Stats report as shown in the code snippet of Figure 3.4

```
# how many reports to load
Report.nrofReports = 2
# default directory of reports (can be overridden per Report with
output setting)
Report.reportDir = reports
# Report classes to load
#Report.report1 = ContactTimesReport
Report.report1 = ConnectivityONEReport
Report.report2 = MessageStatsReport
```

Figure 3.4: Message Stats Report Configuration

The Message stats report was configured to output the number of messages created, aborted, relayed, and delivered. By extending and implementing the Message Listener `(public class MessageStatsReport extends Report implements MessageListener {})`, the stats report also provided the message buffer time, the round-trip time and latencies. It also generated the delivery probability, delay, and the overhead ratio. The complete code for the Message stats report is presented in the Appendix 1A.

Also, the format of displaying the GUI was also configured at this stage as shown in the codes of Figure 3.5

```
## GUI settings
# GUI underlay image settings
GUI.UnderlayImage.fileName = data/helsinki_underlay.png
# Image offset in pixels (x, y)
GUI.UnderlayImage.offset = 64, 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]$
```

Figure 3.5: GUI Configuration

3.5 Replication of the of PRoPICMAN

The PRoPICMAN was replicated using the following steps:

1) **Building nodes' profile using set of evidence pair**

   It was assumed that a sender knows some information about the destination it is intending to send a message to. The information about a node includes; name, residence address, workplace, nationality and hobby. Hobby in this context refers to an activity a node does outside the home and not at his workplace (for example, the mosque or church attended, watching football at the cinema, football, basketball, spending the evening with some group of friends at a particular location, etc.). This information constitutes a nodes' profile. A nodes profile is then represented by a set of evidence pairs. The process of mapping a node's profile to the evidence is presented as in Table 3.1.

Table 3.1: Node Profile to Evidence Mapping

Nodes Profile		Evidence name (E)
Name	--	E_1
Residence Address	------------------------------------	E_2
Work Place	---	E_3
Nationality	--	E_4
Hobby	---	E_5
		$\ldots.E_n$

A value for exists for each evidence as in Figure 3.7

Name: Abdul

Residence Address: Suleiman Hall

Workplace (department): Electrical Engineering

Nationality: Nigerian

Hobby: Basketball

Figure 3.7: Evidence Values

Each evidence is assigned a value. That is, E_1- V_1, $E_2 - V_2$, $E_3 - V_3$, E_4-V_4, $E_5 - V_5$, ……… E_n −,V_n.

2) Hashing the evidences

After assigning values to the evidences, the evidence/value pair of the nodes is then hashed by a hash function H(E_i, V_i) as shown in Table 3.2.

Table 3.2: Hashing of Evidence/Value Pairs

Evidence Names	Value	Hashed Values
Name	Abdul	H(Name, Abdul)
Residence Address	Suleiman Hall	H(Residence Address, Suleiman Hall)
Nationality:	Nigerian	H(Nationality, Nigerian)
Workplace	Electrical Engineering	H(Workplace, Electrical Engineering)
Hobby	Basketball	H(Hobby, Basketball)

3) Development of header for each node

Hashing assigns some particular bits to each evidence/value pair. The process of joining all the hashed values of a particular node is called concatenation.

4) Building the two-hop forwarding strategy

A source node intending to send a message to a destination node first sends the message header to its neighbors (those nodes within the source communication range), each neighbor then receives the header and compares it with its own header in order to obtain the match in evidence/value pairs. The neighbor then computes its delivery probability to the destination based on the matched attributes obtained using the equation (2.4a).

The neighbor further sends the header to its next neighbor (making the second hop). This second hop compares the header with its header to obtain the match in evidence/value pair and then uses it to compute its delivery probability to the destination node. The first hop neighbor picks the second hop with the highest delivery probability and sends it to the source. The source then selects the highest delivery probability of the two-hop routes and forwards the message through that path. The two-hop path is shown in Figure 3.8. This is implemented in the ONE simulator.

The ONE simulator in section 3.3 was used in running the simulations on the Helsinki simulation area. The simulation model in section 3.4 was used in running the simulations. The PRoPICMAN was called from the scenario setting of the ONE simulator using the following code:

Scenario.name = PRoPicmanRouter

The codes for the PRoPICMAN routing is presented in Appendix 2A

Figures 3.9 and 3.10 show the simulation of the PRoPICMAN routing in the ONE simulator at the beginning of the simulation and at time = 2000s respectively.

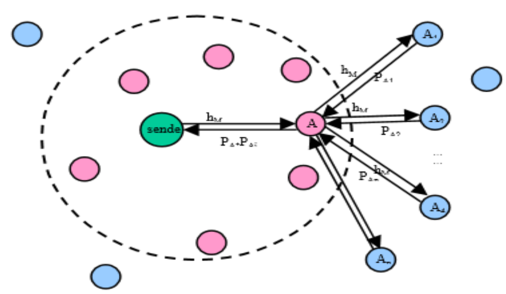

Figure 3.8: Two-Hop Route Selection Process.

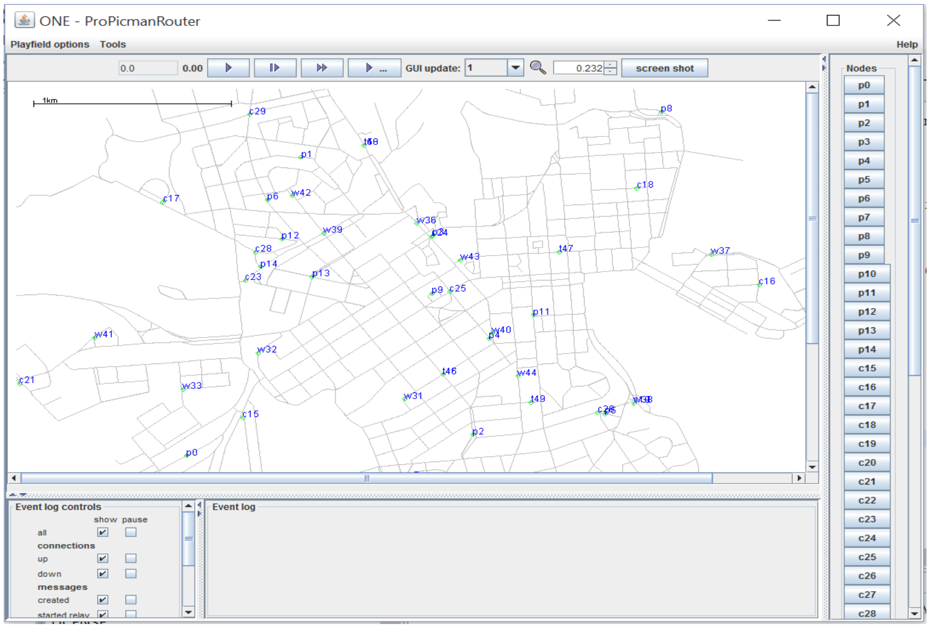

Figure 3.9: Nodes position at the Initial Stage of simulating the PRoPICMAN in the ONE Simulator.

Figure 3.9 was taken at time 0s, nodes are yet to start moving, as such an empty event log can be seen in the Figure. All performance metrics at this time were equal to zero as no exchange of information has taken place.

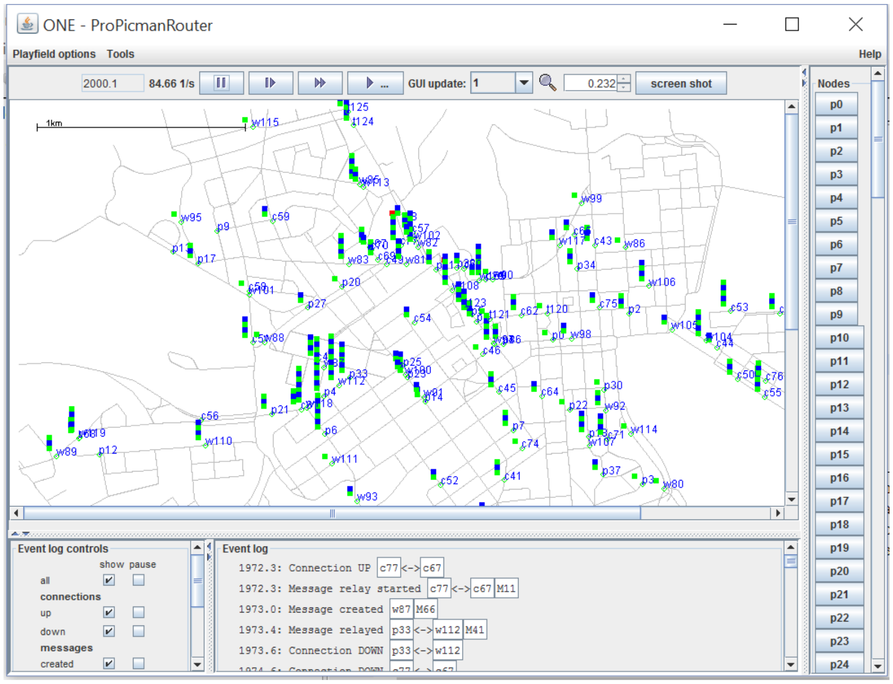

Figure 3.10: Simulation of the PRoPICMAN Routing in the ONE Simulator

Nodes can be seen following through the predefined Helsinki map path in Figure 3.10. Nodes used the two-hop forwarding strategy in forwarding message in this scenario. Nodes were already building their buffers as depicted in the Figure 3.10 and the event log is continuously updating the events taking place.

3.6 Development of the Improved PRoPICMAN (iPRoPICMAN)

The traditional PRoPICMAN was developed for MANET species of DTN. It needed to be modified to make it fit into the oppnet rather than directly applying it as the case of Xi *et al.*, (2015). The modification involved extending the two-hop scheme to a multi-hop scheme. This allows for more than one hobby to be selected in order to improve the chances of getting more matches in attributes that will eventually lead to an increase in the delivery probability computed. The iPRoPICMAN was developed using the following steps:

1) **Building nodes' profile using set of evidence/value pair**

The same information as that of the traditional PRoPICMAN were used in building the nodes profile. (name, residence address, workplace, nationality and hobby). However, this approach assigns more than one hobby to a node since in reality an individual can have as many hobbies as possible. This is done in order to increase the match in the evidence/value pairs.

This allows a node to have numerous values for the evidence hobby as shown in Table 3.3

Table 3.3: Multiple Hobby Values.

Evidence Names	**Value**	**Hashed Values**
Hobby	Basketball, Central Mosque, Social center	H(Hobby, (Basketball Central Mosque, Social center))

2) **Hashing the evidences**

The evidence/value pair of the nodes is then hashed by a hash function H(E_i, V_i) as in the case of the PRoPICMAN. However, the hashed hobby evidence/value is first checked when a header is sent to node in this case.

3) **Development of header for each node**

The hashed values of the evidence/value pairs were then concatenated to form the header of a node.

4) **Building the multi-hop forwarding strategy**

In the multi-hop forwarding strategy, the source node uses as many as possible intermediate nodes available in order fit the oppnet scenario. This is because either the message might never be delivered to the destination if a two-hop scheme is employed, or, the delay will be high. A source node S intending to send a message to destination node D first sends D's header to its first hop neighbor n_1. n_1 checks the header and calculates the match in attribute, calculates its delivery probability to D, and then further send the header to its neighbors n_2 (second hop). n_2 does the same as n_1 and then forwards it to its neighbor n_3. n_3 to n_4., and eventually to n_n. Each intermediate node after calculating its delivery probability sends it to the sending node. The source uses this information to decide the path with the highest delivery probability and then send the message through that path and send the message using equation (3.1). Figure 3.11 shows the multi-hop forwarding scheme. Figures 3.12 and 3.13 show the processes of simulating the iPRoPICMAN in the ONE simulator.

$$P_A = \frac{\sum Matched(X_i)}{\sum X_{Mi}} \tag{3.1}$$

Where X_i is the weight of evidences of the iPRoPICMAN which considers multiple hobbies.

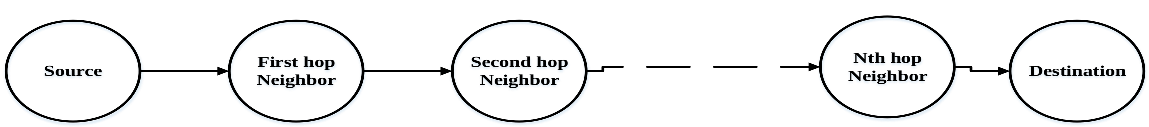

Figure 3.11: Multi-Hop Forwarding Strategy

The iPRoPICMAN is called from the scenario setting of the ONE simulator using the following Code: ***Scenario.name = PRoPICMANImprovedRouter***

The program class for the iPRoPICMAN is presented in Appendix 2B.

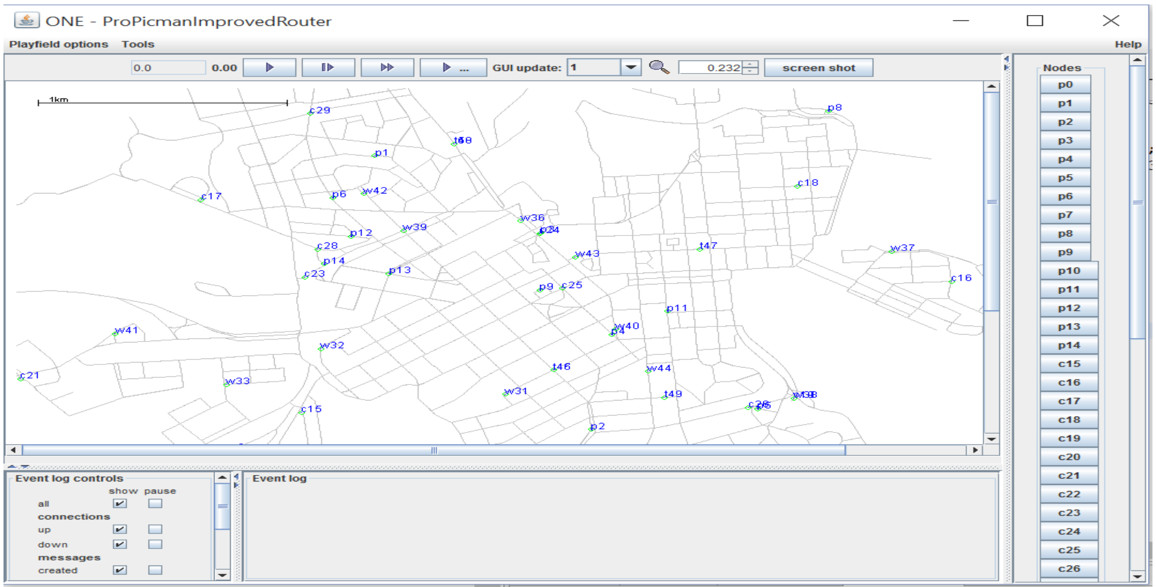

Figure 3.12: Image of the iPRoPICMAN Simulation at the Initial Stage

Figure 3.12 shows the initial stage of simulating the iPRoPICMAN. At this initial stage (time = 0s) the event log is empty as nodes are yet to start moving.

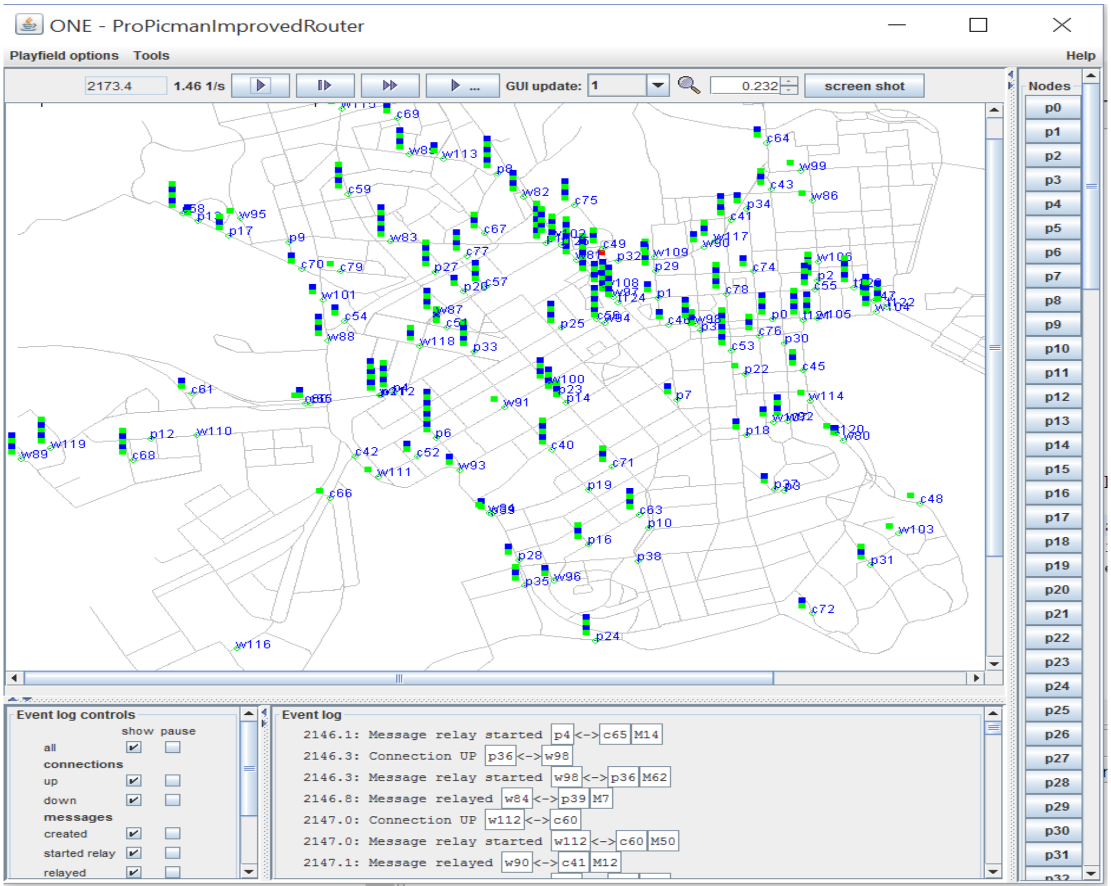

Figure 3.13: Image of Simulation of the iPRoPICMAN in the ONE Simulator.

In Figure 3.13, nodes are following the predefined Helsinki map path and exchanging information with each other when they come in communication range using the multi-hop forwarding scheme. At the simulation time, nodes are already building their buffer. The event log is recording all information which would be used by the simulator in calculating the performance metrics (for example, messages created and messages delivered would be used to calculate the delivery probability in the message stat report).

3.7 Replication of the TMS

The TMS was replicated in order to compare it with the eTMS. The following steps were adopted in order to replicate the TMS:

1) Obtaining nodes' profile information for development of certificate

Every node has its public/private key pairs. When two nodes meet (say node u comes in contact with another node v), they exchange their profile information. Based on this information, node u will check if it owns a valid certificate part to node v, If not, node u will query the attribute list of node v and calculate the match of social attributes Match(u,v) between itself and node v. If match(u,v) is less than a given threshold, u refuses to make connection with node v. Else if match(u,v) is greater than the given threshold, node u will use VFP to inspect the integrity and security of node v in order to ensure that node v's behaviour and security can be trusted. If the identity ID_v and security of node v can be trusted, node u will issue a new certificate $Cert_{u\_v}$ to node v, which is signed by node u. The certificate's reliability $Relia_{u\_v}$ and period of validity T_{u_v} are functions of match(u,v) as shown in the equations (2.5) and (2.6) (Xi *et al*., 2015). T_{max} is taken as the duration of the simulation which is 12 hours in this case. The maximum reliability is left as 0.9 (90%) since it cannot be 1 (100%). The issued certificate $Cert_{u\_v}$ is given by equation (2.7) (Xi *et al*., 2015). TMS only assumes a threshold value which lies between 0 and 1. For this research, 0.6 was adopted as the threshold value.

2) Development of the successor selection algorithm

Successor selection algorithm is used to determine the node amongst the trusted nodes to be given a message. It is the node issuing the message that makes this decision. Identity trust is first tested, whenever identity trust cannot be achieved, behavior trust is used to decide if the successor is qualified for secure forwarding. The successor selection algorithm is presented as in Figure 3.14 (Xi *et al*., 2015).

$TC_{u,v}$ *is the trust value u confirms for v*
If *u has a record of v in its certificate records*
Then
U forwards message to v
Else if $TC_{u,v}$ *>= 0.6*
Then
U forwards the message to v
Else *u stores the message in its buffer and keep moving.*
End if

Figure 3.14: Successor Selection Algorithm

3) Development of the message selection algorithm

Message Selection Algorithm is used by the receiving node to check if it is safe for it to receive a message from the sender (i.e. the sender should be a trusted node too so that it does not accept a malicious packet). Also in message selection algorithm, the identity trust is tested first in order to decide whether a node should accept a message from other nodes for relaying. Message selection algorithm is presented as in Figure 3.15 (Xi *et al.*, 2015).

If *v has a record of u in its certificate records or* $TC_{u,v}$ *>= 0.6*
Then
U accepts message from v
Else if *u has the record of the source certificate in its record*
Then
U accepts message from v
Else *u rejects the message from v.*
End if

Figure 3.15: Message Selection Algorithm

4) Development of the VFP propagation algorithm

VFP Propagation Algorithm is used to ensure that VFP has not expired so as to reflect the recent behavior of nodes. When a node receives a VFP, it will check the validity by examining its expiration date. If it is still valid, then it helps in propagating the VFP. VFP propagation algorithm is presented as in Figure 3.16 (Xi *et al*., 2015):

If *v accepts message from u*
Then
V generates $VFP_{v,u}$ *which has the time record on it and propagates it to encountered nodes*
End if
While *any node* n_i *receives a* $VFP_{v,u}$
Do
If n_i *has a record of v in its certificate records and Time < 10000s*
Then
n_i *testifies the validity of* $VFP_{v,u}$
If $VFP_{v,u}$ *is valid*
Then TC_u *is increased and* n_i *forwards it to encountered node*
End if
Else n_i *drops* $VFP_{v,u}$
End if
End while

Figure 3.16: VFP Propagation Algorithm

The snippet of certificate exchange during the warm up period is shown in Figure 3.17. This process occurs only with the security mechanism in place. Other simulations without security do not have this process.

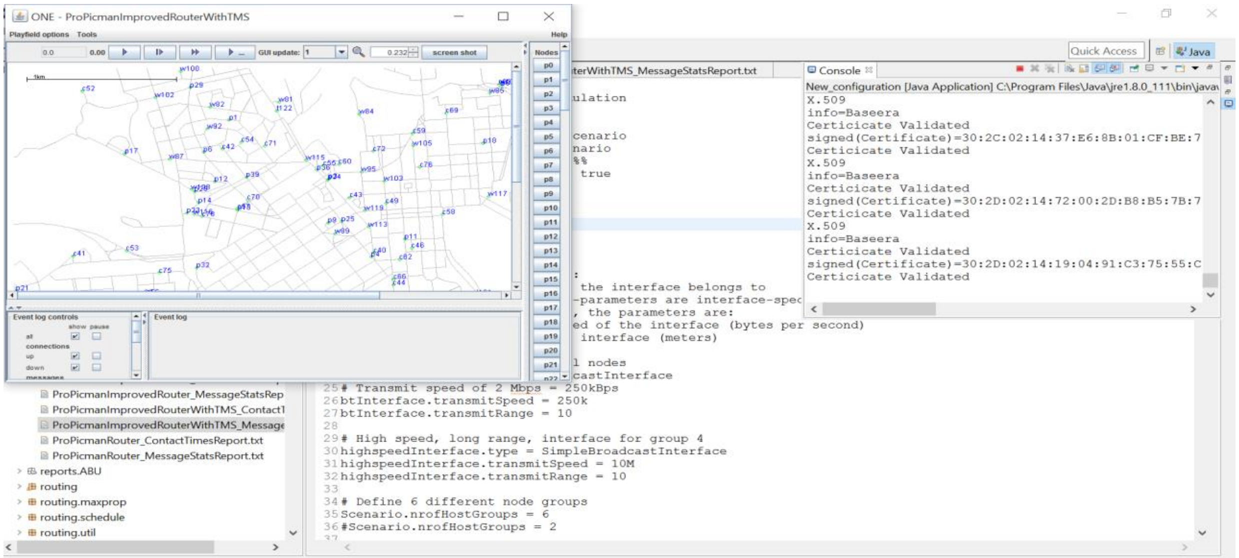

Figure 3.17: Warm Period in TMS

Figure 3.17 shows the process of certificate exchange in simulation of the TMS before the routing starts. This can be seen in the console tab.

3.8 Development of the eTMS

In the development of eTMS, the processes of obtaining nodes' profile information for development of certificate, developing the successor selection algorithm, developing the message selection algorithm and developing the VFP propagation algorithm is the same as that of the TMS, as such it would not be presented again for brevity. The main objective of the eTMS is to make use of an optimum trust threshold value.

3.8.1 Selection of the optimum trust threshold value

The trust threshold value lies between 0 and 1, with 0 indicating no trust while 1 indicating full trust. The determination of an optimum value is critical because selecting a value that is too low (say 0.2) will imply that most of the nodes will have access to the network to do whatever kind of activity they intend to do (including malicious activities that can eventually decrease the performance of the network). This is because most of the malicious nodes can attain the low value. On the other hand, selecting a value that is too high (say 0.9) will imply that nodes (including non-malicious nodes) will be restricted from accessing the network, thus reducing the number of intermediate forwarders. This will result in a higher time for a message to be delivered (increased delay) and in some cases, decreased delivery probability of the network. This is because many well-behaved nodes may not be able to attain the high value.

An experimental (simulation) setup was used to determine the optimum value that the threshold value should be kept so that malicious nodes are filtered out and non-malicious nodes are not prevented from joining the network. It should be noted that some simulations can run for between 12 – 48 hours depending on the type of scenario. The processes involved in the simulation of the eTMS include:

1) Setting up the simulation environment
2) Modeling the scenario in the default setting of ONE simulator
3) Uploading the iPRoPICMAN in the ONE simulator on the Helsinki simulation region
4) Setting the Message stat report in ONE simulator
5) Running the TMS in the iPRoPICMAN
6) Varying the trust threshold value between 0 and 1 at intervals of 0.1 in the TMS.
7) The range where the best results (from 6) are obtained is further varied at intervals of 0.01.

8) The optimum value is selected where the best result obtained is 0.73. This value is now used to run the TMS in order to get the eTMS.

3.9 Running Simulations of TMS and eTMS.

The ONE simulator set in section 3.2 is used in running the simulations on the Helsinki simulation area as described in section 3.3. The iPRoPICMAN is used for the TMS and eTMS since it is seen to perform better for oppnets in the simulations. The simulations are presented as:

1) **TMS on the Helsinki simulation area**

The TMS was called from the scenario setting of the ONE using the following code:

Scenario.name = ProPicmanImprovedRouterWithTMS

This code prompts the graphical user interface of the ONE simulator as shown in Figure 3.18.

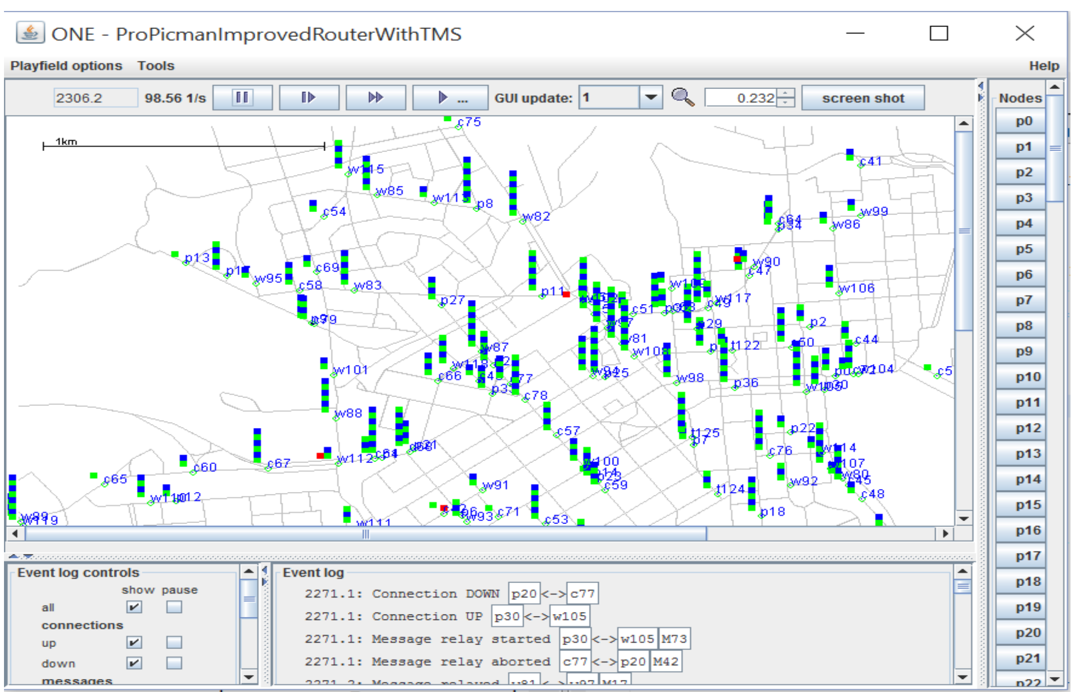

Figure 3.18: Image of Simulation of the TMS in the ONE Simulator

Figure 3.18 shows the image of simulation of the TMS on the iPRoPICMAN in the ONE simulator. The images on the screen are nodes moving in their predefined paths.

2) **eTMS on Helsinki simulation area**

The same scenario setting of the TMS in the ONE simulator is used to run the eTMS on the iPRoPICMAN. The main difference is that the best experimental threshold value was used in running the eTMS.

3) Comparison between the TMS and the eTMS

The performance of the TMS and eTMS are compared using the following performance metrics: delivery probability, delay and overhead ratio.

The contributions of this research led to the modified flow charts of the TMS as presented in Figures 3.19 and 3.20. Processes *1 and decision *2 are of interest to this research. They are the points at which the modifications would be applied.

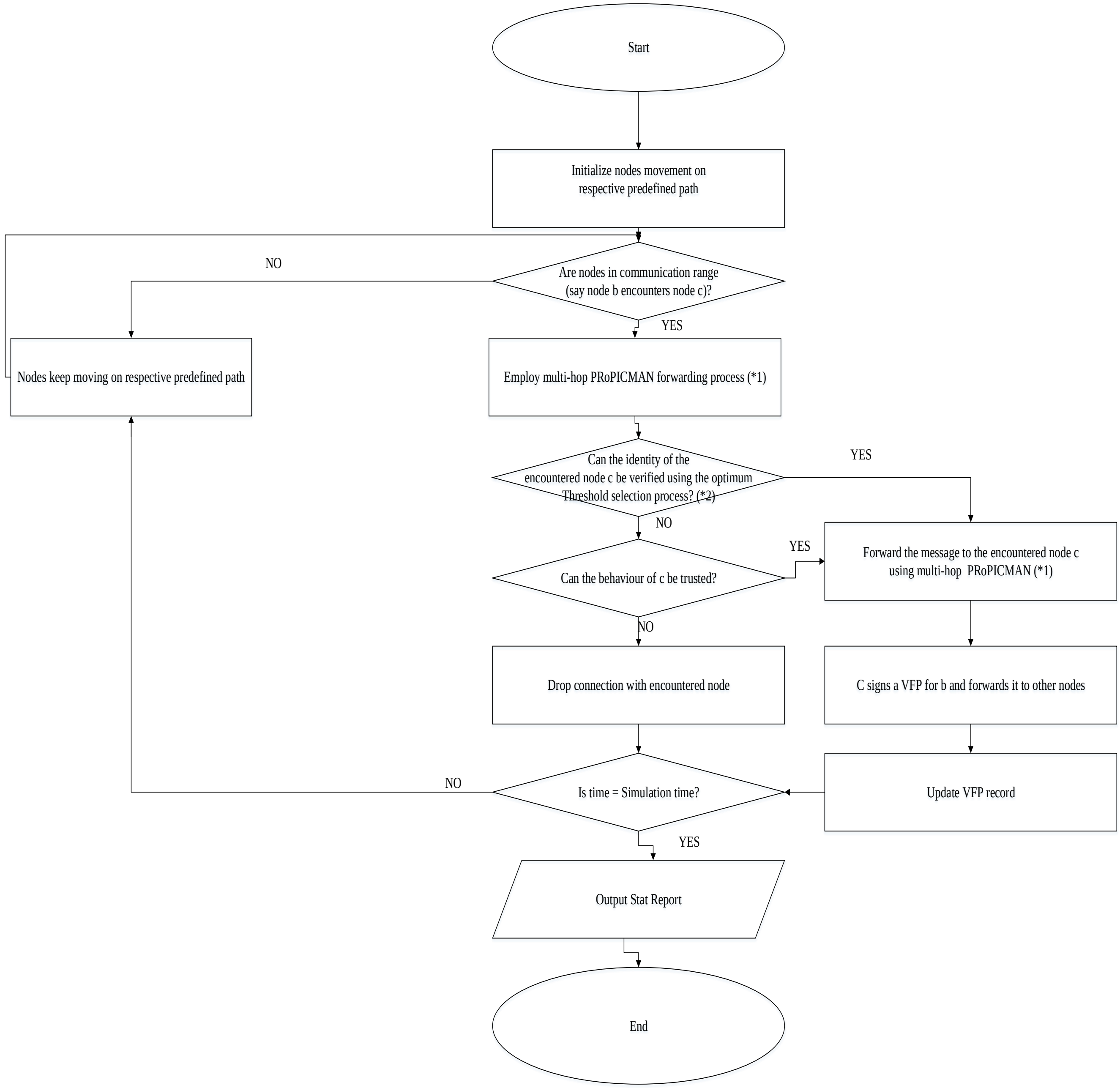

Figure 3.19: Proposed eTMS Routing Process

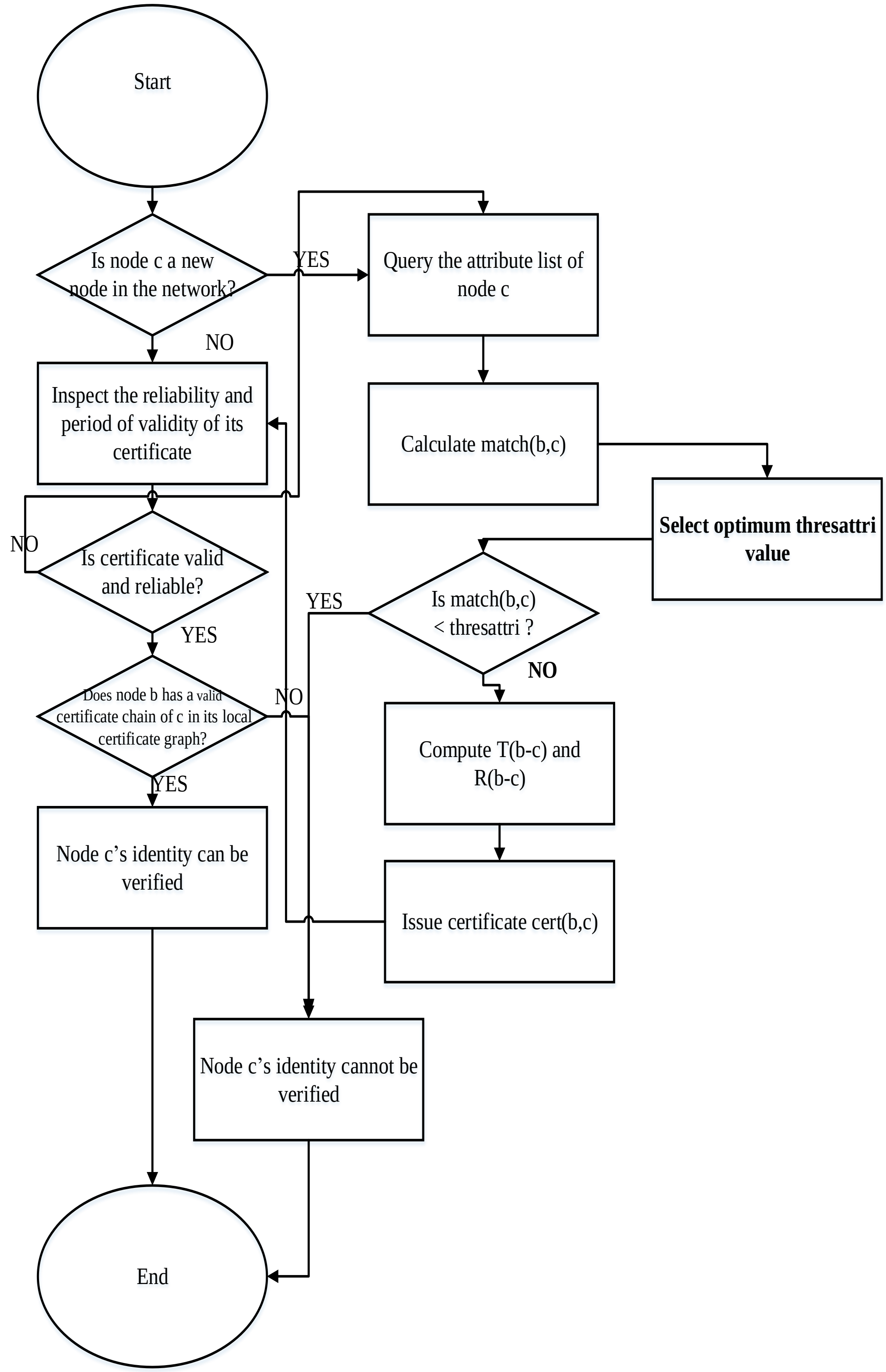

Figure 3.20: Modified Process of Nodes' Identity Verification

3.10 Modeling the ABU Network Simulation Area.

The following steps were adopted in order to implement ABU scenario on the ONE simulator.

1) **Defining the types of nodes**

   A hundred (100) nodes were used like that of the conventional TMS model. Twenty-five nodes needed permission by the trusted initial nodes in order to access to the network. Communication is assumed to be between smart phones and similar devices capable of serving as oppnet nodes. Users holding these devices are defined as pedestrians and people in cars, tricycles and motorcycle constitute other nodes.

2) **Defining the characteristic of nodes**

   The speed of a node is determined by the category of user holding it and as such, 0.8–1.6m/s is used for walking speed of nodes (i.e. pedestrians) and 6m/s for cars and other nodes. The Bluetooth data rate of 250kB/s was used with a transmission range of about 10m. The buffer size of nodes was assumed to be 30MB with a message size between 500kB-2MB. The event generation range is assumed to be 25-35s while the simulation time is also taken as 43200s.

3) **Defining the simulation area**

   The core academic area of ABU is used in the simulation. This region is depicted in Figure 2.17. The region spans from latitude 11.113345ºN to 12.2768518 ºN and longitude 6.6132515 ºE to 7.7266562 ºE with a total of 578 connected streets.

   Extracting line map from the map.

   The map was obtained in .osm file extension format from www.openstreetmap.com. The obtained map was stored as ABuniversity.osm. Information and data from the

ABU map in the .osm file extension format were encrypted. In order to get access to this information, the .osm file was converted into a well-known text (.wkt) file format using the command window. This was done by first changing the current directory of the command window to the directory where the .osm file is stored in the computer using the following command:

C:\Users\basee\Desktop\ABU MAP\osm2wkt\release\v1.1.0

The .osm file was then converted into .wkt format by adding the following command to the previous command

>java -jar osm2wkt.jar ABuniversity.osm

The snippets of the extraction processes on the command window are given in Figures 3.21 and 3.22.


```
Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\HP USER>cd\

C:\>Users\HP USER\Desktop\ABU MAP\osm2wkt\release\v1.1.0
'Users\HP' is not recognized as an internal or external command,
operable program or batch file.

C:\>cd Users\HP USER\Desktop\ABU MAP\osm2wkt\release\v1.1.0

C:\Users\HP USER\Desktop\ABU MAP\osm2wkt\release\v1.1.0>java -jar osm2wkt.jar ma
p.osm
osm2wkt v1.2.0- convert openstreetmap to wkt - Christoph P. Mayer - mayer@kit.ed
u
converting file map.osm ...
reading in openstreetmap xml ...
osm file map.osm does not exist

C:\Users\HP USER\Desktop\ABU MAP\osm2wkt\release\v1.1.0>java -jar osm2wkt.jar AB
university.osm
osm2wkt v1.2.0- convert openstreetmap to wkt - Christoph P. Mayer - mayer@kit.ed
u
converting file ABuniversity.osm ...
reading in openstreetmap xml ...
parsing osm found 587 streets and 4445 landmarks
transforming geographic landmarks ...
found geographic bounds: latitude from 11.113345 to 12.2768516 longitude from 6.
6132515 to 7.7266562
geographic area dimensions are: height 129376.031m, width 121483.267m
checking landmark completeness for all streets ...
all required landmarks available for all streets
do you want to fix missing landmarks? this will take very long but can heavily r
educe map partitioning
type 'y' or 'n': y
checking for missing landmarks for crossing street parts. this may take a while
...
inserted 108 missing landmarks. currently have 4550 landmarks
simplicatiation run 1 of max 10
simplyfing model, removing unconnected parts ...
graph is not connected, analyzing partitions ...
found 508 partitions: [1] [4] [5] [6] [10] [767] [1001] [6] [28] [7] [12] [4] [1
5] [6] [4] [4] [4] [11] [4] [4] [8] [12] [4] [8] [4] [4] [4] [1] [4] [1] [4] [8]
 [4] [4] [13] [4] [4] [4] [6] [1] [4] [4] [5] [4] [4] [4] [13] [4] [18] [6] [4]
[4] [4] [6] [4] [4] [6] [4] [5] [4] [4] [17] [17] [8] [6] [8] [8] [4] [7] [7] [2
6] [10] [6] [4] [4] [4] [10] [6] [4] [4] [4] [4] [10] [4] [4] [4] [4] [4] [4] [4
] [4] [4] [4] [4] [8] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [6] [4] [4] [1
1] [12] [6] [7] [1] [4] [4] [18] [6] [6] [6] [6] [24] [4] [4] [4] [14] [4] [4] [
8] [4] [4] [4] [4] [6] [6] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [
4] [4] [4] [12] [8] [14] [4] [4] [4] [8] [15] [13] [5] [12] [4] [4] [4] [4] [4]
[6] [5] [4] [4] [4] [4] [4] [4] [1] [4] [4] [8] [4] [6] [12] [8] [4] [4] [8] [4]
 [4] [7] [4] [11] [12] [4] [6] [4] [4] [4] [4] [10] [6] [4] [4] [4] [12] [4] [10
] [4] [8] [6] [4] [10] [4] [4] [9] [8] [4] [4] [4] [4] [8] [4] [4] [10] [4] [10]
 [10] [4] [10] [4] [4] [4] [4] [10] [4] [10] [4] [10] [11] [10] [4] [4] [10] [10
] [4] [4] [5] [4] [4] [4] [4] [4] [5] [4] [4] [4] [4] [4] [4] [4] [11] [6] [4] [
4] [4] [4] [6] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [4] [
4] [4] [4] [4] [4] [4] [4] [4] [8] [4] [5] [4] [4] [4] [4] [4] [4] [4] [10] [4]
```

Figure 3.21: Command Prompt with Extracted Information

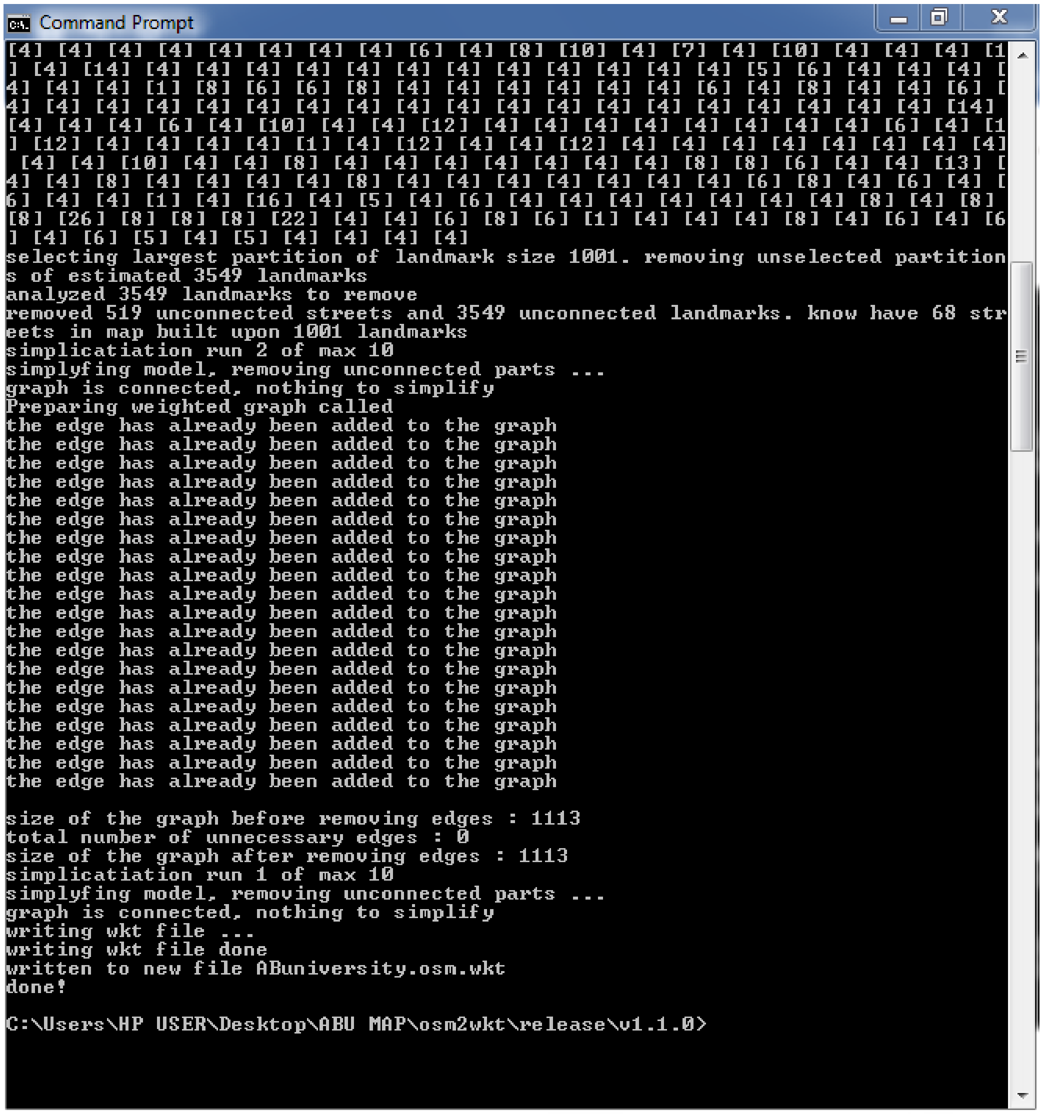

Figure 3.22: Extraction of ABU Map on the Command Window

Figures 3.21 and 3.22 show the extraction process of the ABU map to be imported into the ONE simulator. Some vital information (which include width, height, connect street, landmarks, latitude, longitude, map size) can be viewed from the Figures 3.21 and 3.22.

4) Importing the line map into the ONE simulator and defining path for different types of nodes

After the map has been converted from the .osm format to the .wkt file format, the data and the map itself needed are saved for importing into ONE simulator. This is done using OpenJUMP software. In the OpenJUMP software, the map is saved along with other information such as roads.wkt (which specifies path followed by all nodes in the network, including pedestrians), pedestrians.wkt (these are paths on the map that are used by only pedestrians), shops.wkt, building.wkt, etc. These are information used alongside the name of the map in ONE simulator.

3.11 Running Simulations on the ABU Simulation Area.

The simulation of the following were carried out:

1) **TMS using the iPRoPICMAN on ABU simulation area**

   The TMS was run on the described ABU modeled simulation area using the iPRoPICMAN just like case with the Helsinki simulation region. The same assumed threshold value of 0.6 was used. This simulation was called from the scenario setting of the ONE simulator using the following code:

   Scenario.name = ProPicmanImprovedRouterWithTMS

   This code prompts the graphical user interface of the ONE simulator as shown in Figure 3.23.

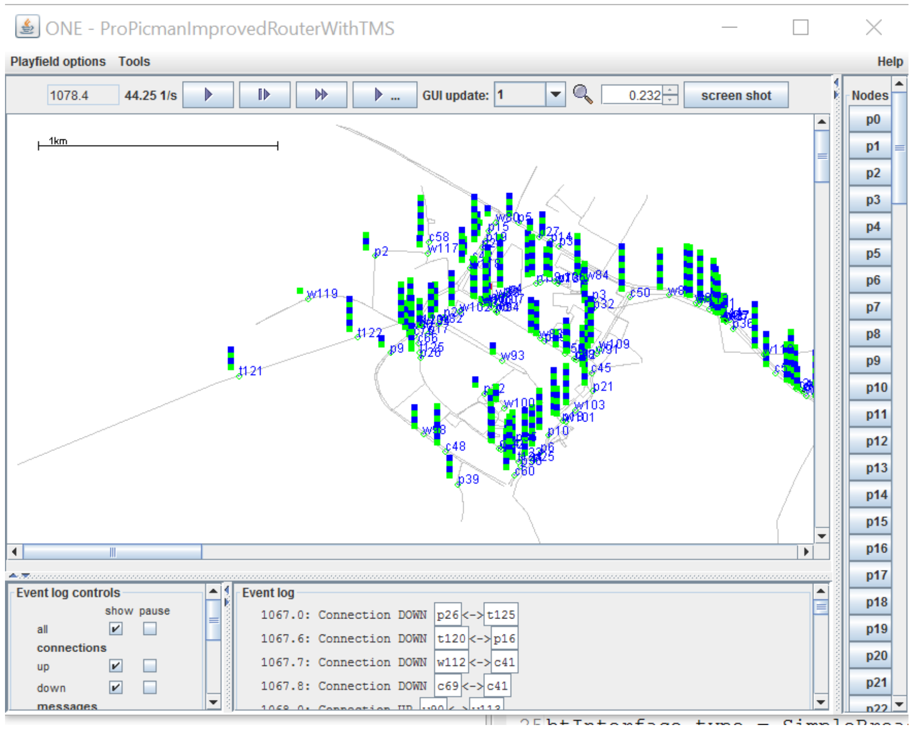

Figure 3.23: Image of Simulation of the TMS on ABU in the ONE Simulator

Figure 3.23 shows the simulation process of the TMS on the iPRoPICMAN in the ONE simulator on ABU simulation area. As seen from Figure, the ABU has less connected streets than the Helsinki simulation area. There are less connected paths as compared to developed countries.

2) eTMS using the iPRoPICMAN on ABU simulation area

Similarly, the same scenario setting of the TMS in the ONE simulator is used to run the eTMS on the iPRoPICMAN on the ABU simulation region. The main difference is that the best experimental threshold value was used in running the eTMS. The best threshold value was also obtained by running simulations like that of the Helsinki simulation region.

3) **Comparison between the TMS and the eTMS**

The performance of the TMS and eTMS are compared using delivery probability, delay and overhead ratio as performance metrics.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Introduction

This section presents the results obtained from the simulations using the ONE simulator and their discussion. The performance of the PRoPICMAN and the iPRoPICMAN are presented in this section. Also, the performance of the TMS and the eTMS on the Helsinki simulation region and the ABU simulation area are evaluated here. The order of results presentation has been made in accordance to the sequence of the methodology which was laid based on the sequence of the stated objectives. In order to calculate the percentage improvement, Equation 4.1 is used;

$$\frac{\text{Initial value} - \text{Final value}}{\text{Initial value}} * 100 \qquad (4.1)$$

4.2 Result of the PRoPICMAN

The results obtained from simulation of the PRoPICMAN in the ONE simulator on the Helsinki simulation area are as shown in Table 4.1.

Table 4.1: Message Statistical Report for the PRoPICMAN Simulation

Sim_time (s)	Delivery Probability	Delay (s)	Overhead ratio
0	0	0	0
1000	0.0303	728	40
2000	0.0299	883	43
4000	0.2164	1841	45
6000	0.2376	2160	56
8000	0.2788	2652	53
10000	0.2861	2972	55
12000	0.2696	3175	61
16000	0.2583	3468	66
20000	0.2667	3762	67
25000	0.2719	3921	67
30000	0.2729	4116	68
35000	0.2758	4240	68
40000	0.2846	4456	67
43200	0.2806	4613	68

Table 4.1 shows that over the simulation period of 0 to 43200s, the delivery probability varied between 0 and 0.2806 while the delay increased from 0 to 4613s. The overhead ratio also increased from 0 to 68. The delivery probability gradually increased between time=0s to about 30000s, then became partially stable till the end of the simulation time.

4.3 Results of the iPRoPICMAN

The results obtained from simulation of the iPRoPICMAN in the ONE simulator on the Helsinki simulation area are as shown in Table 4.2.

Table 4.2: Message Statistical Report for the iPRoPICMAN Simulation.

Sim_time (s)	Delivery Probability	Delay	Overhead ratio
0	0	0	0
1000	0.0303	728	35
2000	0.0299	735	38
4000	0.2239	1847	40
6000	0.2574	2191	47
8000	0.3234	2736	43
10000	0.3569	3244	43
12000	0.3505	3452	47
16000	0.3432	3761	50
20000	0.3407	4044	52
25000	0.3700	4729	50
30000	0.4049	5213	46
35000	0.4113	5384	46
40000	0.4169	5548	46
43200	0.4157	5717	46

Table 4.2 shows that over the simulation period of 0 to 43200s, the delivery probability varied between 0 and 0.4157 while the delay increased from 0 to 5717s. The overhead ratio also

increased from 0 to 46. This implies that less overhead ratio is incurred at the beginning of the simulation (when less activity is taking place).

4.4 Comparison of the PRoPICMAN with the iPRoPICMAN

The performances (delivery probability, delay and overhead ratio) of the PRoPICMAN and iPRoPICMAN routing on the Helsinki simulation area are compared as shown in Figures 4.1 to 4.3. The plots are generated using MATLAB scripts of Appendices 3A to 3C.

From Figure 4.1, it can be seen that over the simulation period, the iPRoPICMAN outperformed the PRoPICMAN by 48% (considering the final average values of 0.4157 and 0.2806) with respect to the delivery probability. This is as a result of the fact that the iPRoPICMAN is more flexible than the PRoPICMAN, thereby, increasing the opportunities for message delivery.

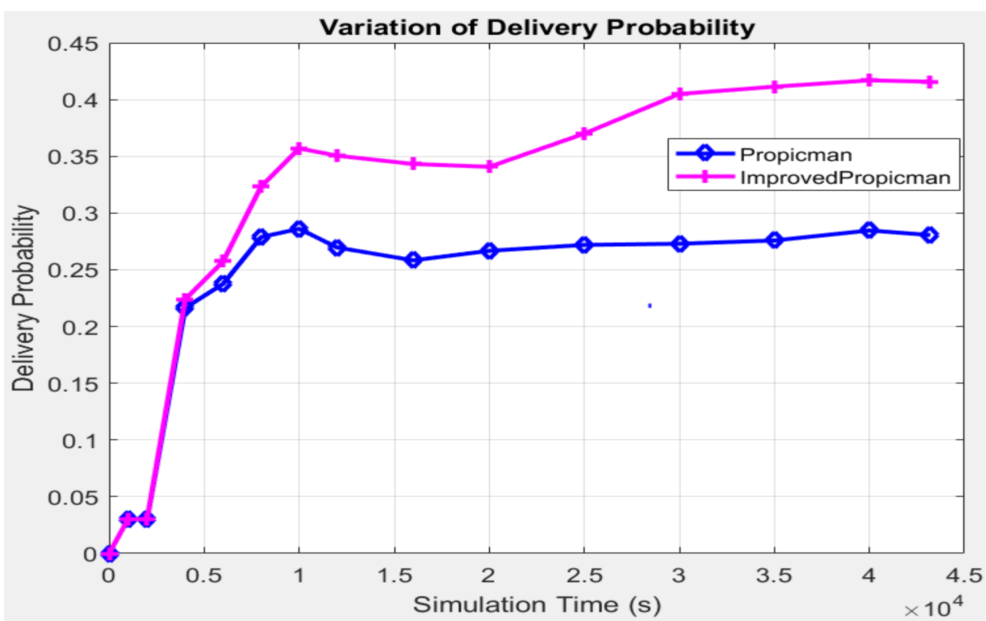

Figure 4.1: Variation of Delivery Probability with Time

From Figure 4.2, it can be seen that over the simulation period, the iPRoPICMAN incurred a higher delay than the PRoPICMAN. This is due to the additional time required by the iPRoPICMAN in selecting multiple hobbies and using multiple hops. As such, it had to trade-off lower delay for higher delivery probability. For DTNs, this is tolerable since more messages will eventually be delivered to their destination. The PRoPICMAN outperformed the iPRoPICMAN by 23% (considering the final average values of 4613s and 5717s) with respect to delay.

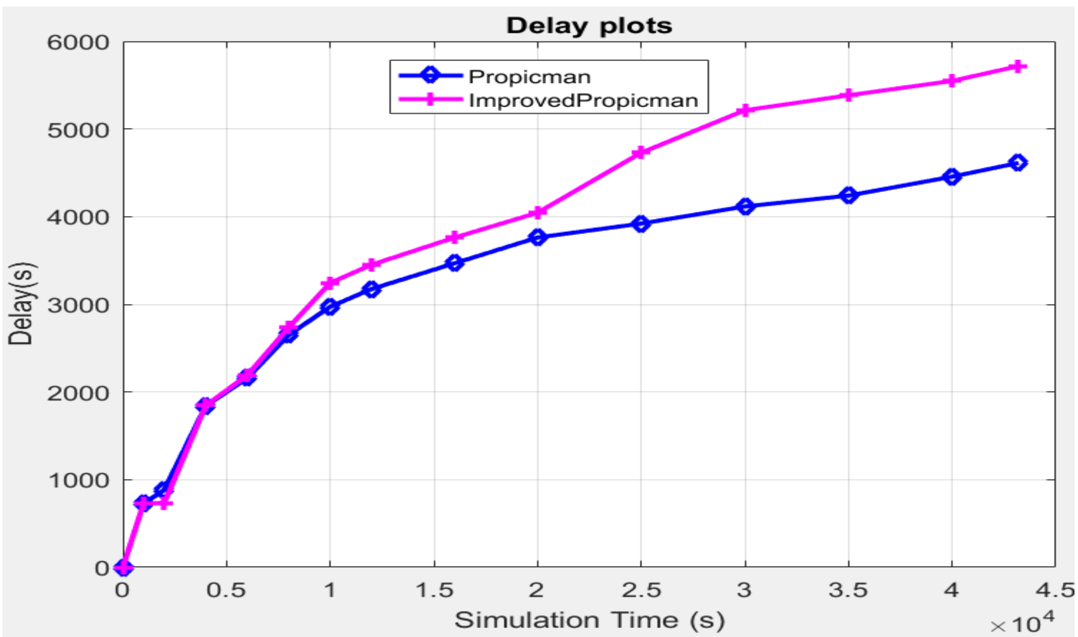

Figure 4.2: Variation of Delay with Time

From Figure 4.3, it can be seen that over the simulation period, the iPRoPICMAN obtained a lower network overhead than the PRoPICMAN. A lower network overhead is a characteristic of a good routing protocol because it is an indication that the routing protocol used lower system resources (CPU, memory, bandwidth, etc.) in delivering a packet. The iPRoPICMAN outperformed the PRoPICMAN by 32% (considering the final average values of 46 and 68) with respect to the overhead ratio.

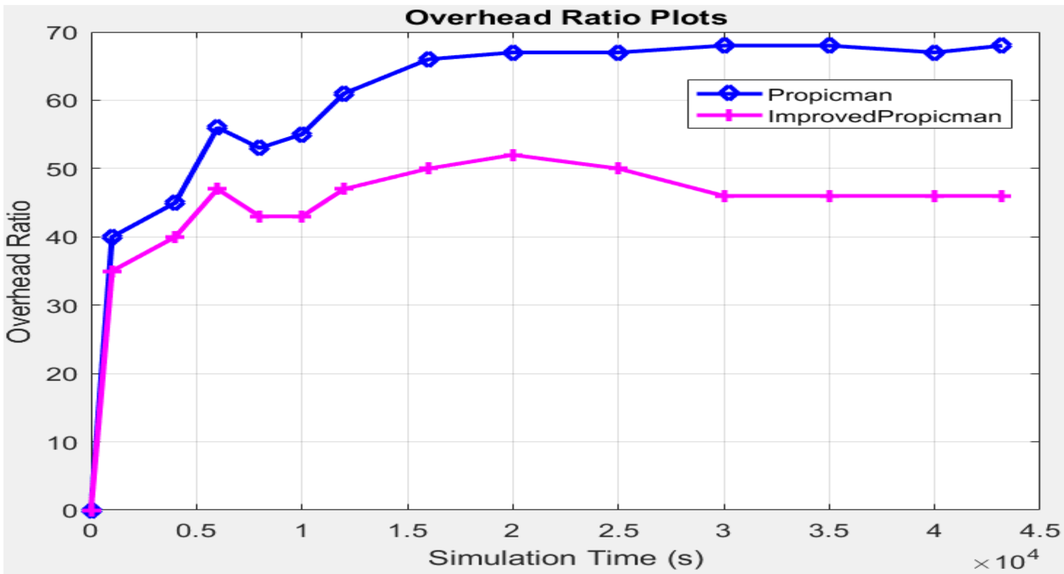

Figure 4.3: Variation of Overhead Ratio with Time

4.5 Results of the iPRoPICMAN with TMS

The results obtained from simulation of the iPRoPICMAN with the TMS in the ONE simulator on the Helsinki simulation area are as shown in Table 4.3.

Table 4.3: Message Statistical Report for the iPRoPICMAN with TMS Simulation

Sim_time (s)	Delivery Probability	Delay	Overhead ratio
0	0	0	0
1000	0.0303	728	43
2000	0.0448	953	45
4000	0.2015	1714	47
6000	0.2723	2285	45
8000	0.3494	2872	39
10000	0.3835	3401	39
12000	0.3750	3649	43
16000	0.3801	4173	45
20000	0.3926	4589	44
25000	0.4279	5314	43
30000	0.4690	5806	39
35000	0.4890	6294	38
40000	0.5026	6529	37
43200	0.4976	6713	38

Table 4.3 shows that over the simulation period of 0 to 43200s, the delivery probability varied between 0 and 0.4976 while the delay increased from 0 to 6713s. The overhead ratio also increased from 0 to 38.

The performances (delivery probability, delay and overhead ratio) of the iPRoPICMAN and iPRoPICMAN with TMS routing on the Helsinki simulation area are compared as shown in Figures 4.4 to 4.6. The plots are generated using MATLAB scripts of Appendices 3D to 3F.

From Figure 4.4, it can be seen that over the simulation period, the iPRoPICMAN with TMS outperformed the PRoPICMAN by 19% (considering the final average values of 0.4976 and 0.4157) with respect to the delivery probability. This can be attributed to the fact that packet dropping activities of malicious nodes are reduced by the TMS.

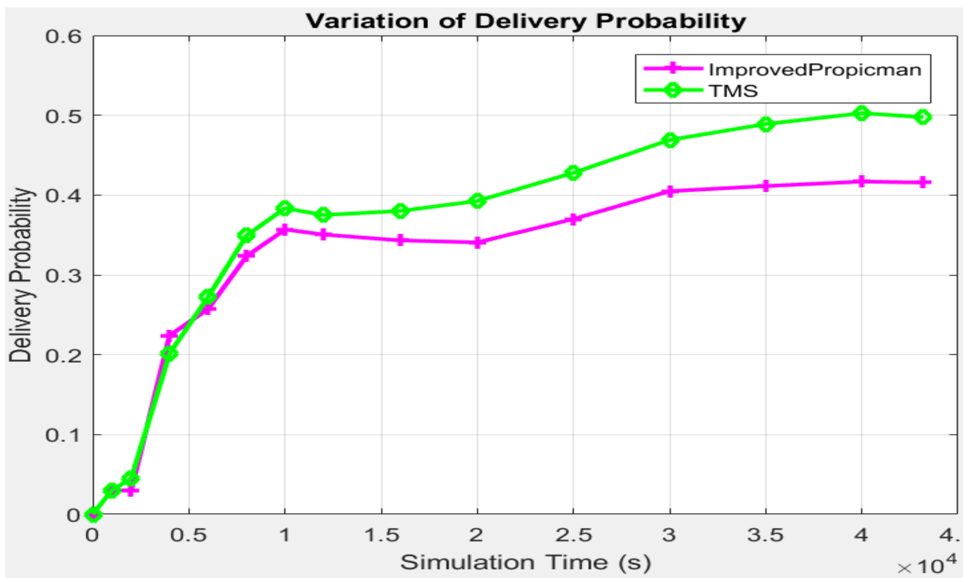

Figure 4.4: Variation of Delivery Probability with Time for the iPRoPICMAN and the iPRoPICMAN with TMS

From Figure 4.5, it can be seen that over the simulation period, the iPRoPICMAN with TMS incurred a 17% increment in delay (considering the final average values of 5717 and 6713) compared with the iPRoPICMAN. This increased delay is due to the additional processes required to secure the messages during routing. Since the network is delay tolerant, it is worth trading delay for a higher delivery probability and better network security.

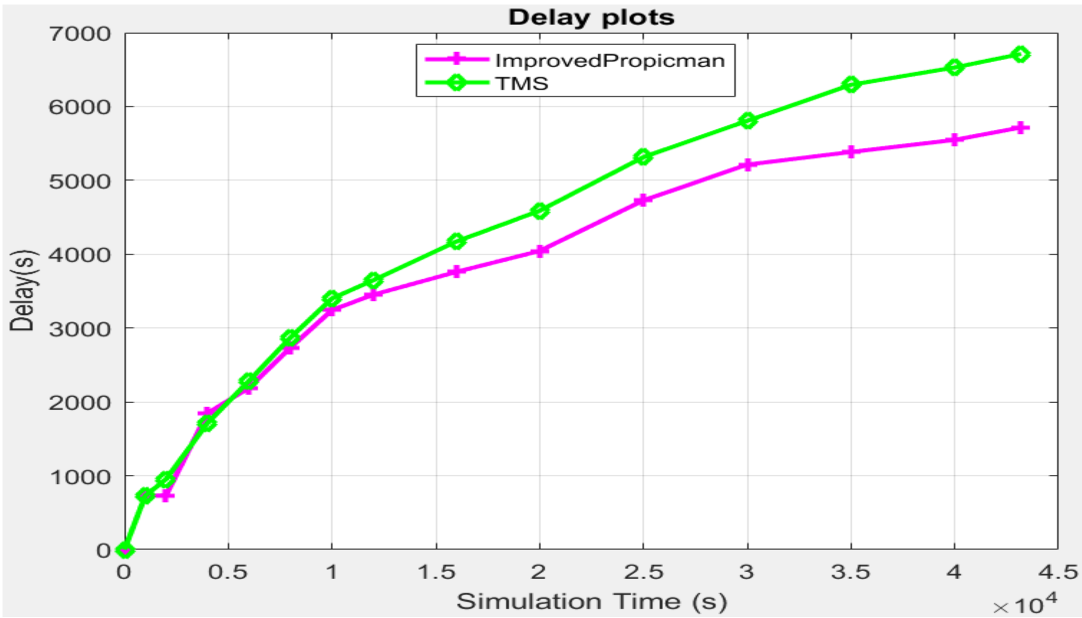

Figure 4.5: Variation of Delay with Time for the iPRoPICMAN and the iPRoPICMAN with TMS.

From Figure 4.6, it can be seen that over the simulation period, the iPRoPICMAN with TMS obtained a comparatively lower network overhead than the iPRoPICMAN. The iPRoPICMAN with TMS outperformed the iPRoPICMAN by 17% (considering the average final values of 38 and 46) with respect to the overhead ratio.

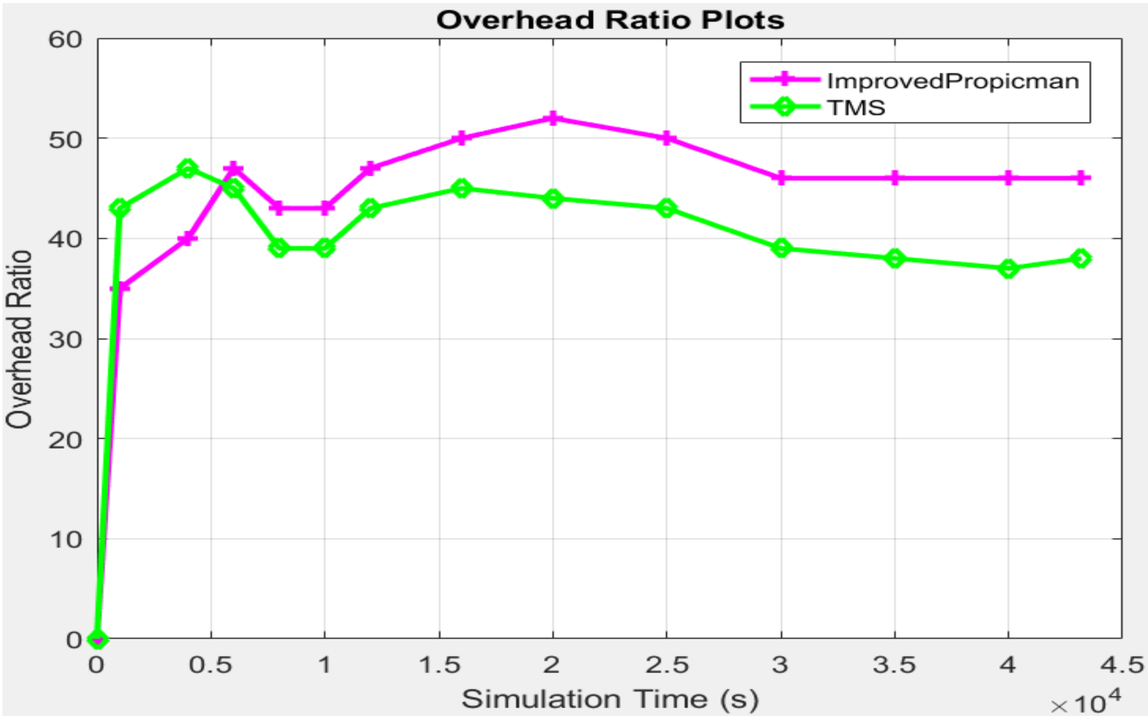

Figure 4.6: Variation of Overhead Ratio with Time for the iPRoPICMAN and the iPRoPICMAN with TMS

4.6 Results of the eTMS

The results of the several simulations carried out on the TMS in order to obtain the best threshold value are as presented in Table 4.4. The threshold values were varied from 0 to 1 in intervals of 0.1 and each simulation runtime was set at 43200s.

Table 4.4: Result of the Various Threshold Values using the Helsinki Simulation Area

Threshold	Delivery Probability	Delay	Overhead ratio
0.1	-	-	-
0.2	-	-	-
0.3	-	-	-
0.4	-	-	-
0.5	0.3215	6824	38
0.6	0.4976	6713	38
0.7	0.5044	6605	38
0.8	0.4909	6624	38
0.9	0.4901	6531	39
1.0	-	-	-

It can be observed from Table 4.4 that when the threshold value was set at 0.1 to 0.4, the simulations timed out before 43200s and as such, no results could be obtained. At 0.1, it timed out after 25000s and at 39000s when the set at 0.2. At 0.3, it timed out after 17000s and after 28000s when set at 0.4. It was also observed that the highest delivery probability was obtained when the threshold was set between 0.6 and 0.8, while the overhead ratio was constant. However, there were slight variations in the delay time.

The interval, 0.6 to 0.8, is further varied in steps of 0.01 (0.61 to 0.80) in order to obtain the best possible threshold value, it was observed to occur at about 0.73 as seen in Table 4.5.

Table 4.5: Results for Threshold Values between 0.61 and 0.80

Threshold	Delivery Probability	Delay	Overhead ratio
0.61	0.4972	6713	38
0.62	0.4949	6610	38
0.63	0.4901	6529	39
0.64	0.4935	6652	38
0.65	0.4919	6502	38
0.66	0.4928	6506	39
0.67	0.4935	6557	38
0.68	0.4936	6542	39
0.69	0.4962	6598	38
0.70	0.5044	6606	38
0.71	0.4923	6529	38
0.72	0.4901	6509	39
0.73	0.5065	6607	37
0.74	0.4996	6829	38
0.75	0.4993	6731	39
0.76	0.4990	6696	38
0.77	0.4828	6682	38
0.78.	0.4909	6673	38
0.79	0.4911	6632	38
0.80	0.4909	6624	38

The obtained optimum threshold value of 0.73 is then used in the TMS to obtain the eTMS. The results obtained from simulation of the iPRoPICMAN with the eTMS in the ONE simulator on the Helsinki simulation area are as shown in Table 4.6.

Table 4.6: Message Statistical Report for the iPRoPICMAN with eTMS Simulation

Sim_time (s)	Delivery Probability	Delay	Overhead ratio
0	0	0	0
1000	0	0	0
2000	0.0299	1117	39
4000	0.2313	1928	40
6000	0.2822	2295	43
8000	0.3457	2794	40
10000	0.3717	3295	41
12000	0.3848	3580	42
16000	0.3893	4195	44
20000	0.4044	4562	43
25000	0.4313	5258	43
30000	0.4788	5911	38
35000	0.4959	6228	37
40000	0.5107	6571	37
43200	0.5065	6607	37

Table 4.6 shows that over the simulation period of 0 to 43200s, the delivery probability varied between 0 and 0.5065 while the delay increased from 0 to 6707s. The overhead ratio also varied between 0 and 44.

4.7 Comparison between the iPRoPICMAN with TMS and iPRoPICMAN with eTMS

The performances (delivery probability, delay and overhead ratio) of the iPRoPICMAN with TMS and iPRoPICMAN with eTMS routing on the Helsinki simulation area are compared as shown in Figures 4.7 to 4.9. The plots are generated using MATLAB scripts of Appendices 3G to 3I.

From Figure 4.7, it can be seen that over the simulation period, the iPRoPICMAN with eTMS outperformed the iPRoPICMAN with TMS by 2% (considering the final values of 0.5065 and 0.4976) with respect to the delivery probability. It is worth recalling that a threshold value of 0.6 was used for the TMS (as compared to the optimum value of 0.73) and that Table 4.4 shows how critical the threshold value can be (since it is even possible to obtain zero network performance).

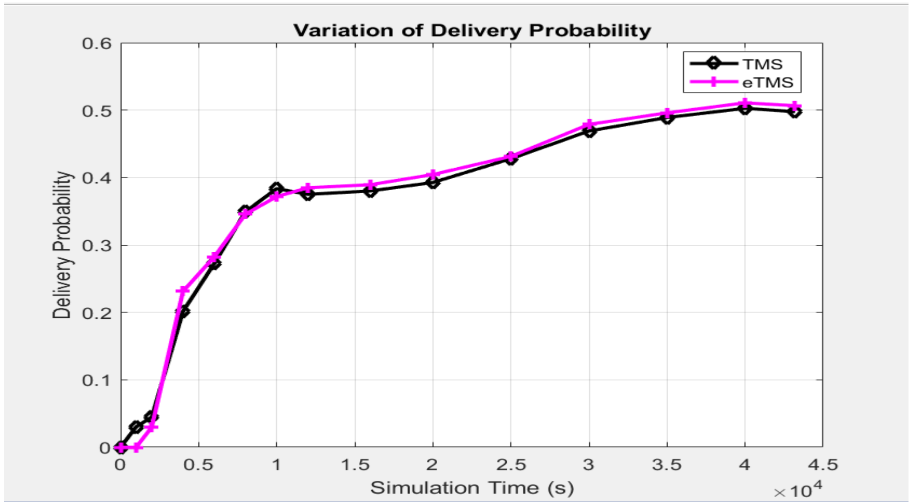

Figure 4.7: Variation of Average Delivery Probability with Time for the iPRoPICMAN with TMS and eTMS.

From Figure 4.8, it can be seen that over the simulation period, the iPRoPICMAN with eTMS had a reduced (2%) delay (considering the final values of 6607s and 6713s) compared with the iPRoPICMAN with TMS. This is consistent with Table 4.4, where the. threshold value is shown to have a slight effect on the delay.

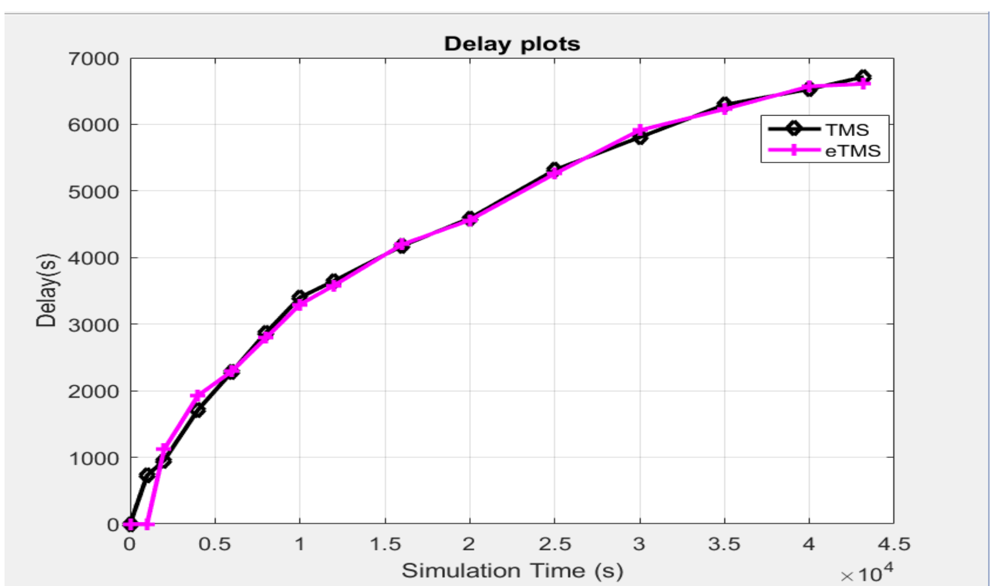

Figure 4.8: Variation of Delay with Time for the iPRoPICMAN with TMS and eTMS

From Figure 4.9, it can be seen that over the simulation period, the iPRoPICMAN with eTMS comparatively obtained similar network overhead with the iPRoPICMAN with TMS. This is consistent with Table 4.4 where it is seen that the threshold value did not necessarily affect the overhead ratio.

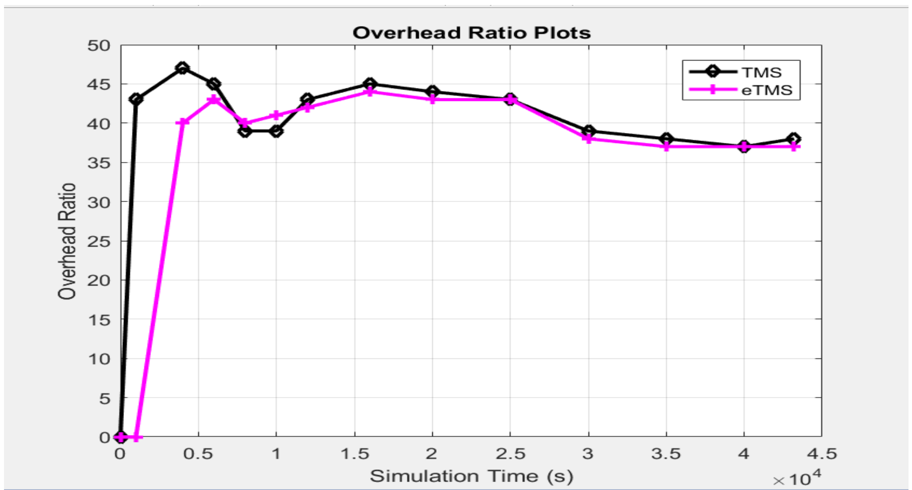

Figure 4.9: Variation of Overhead Ratio with Time the iPRoPICMAN with TMS and eTMS.

In order to further illustrate the effect of the trust threshold value, an assumed threshold of 0.85 was used on the TMS. The delivery probability, delay and overhead ratio obtained were 0.4901, 6527 and 39 respectively. The eTMS outperformed the TMS in this case by 3% with respect to delivery probability.

With an assumed threshold of 0.5, the delivery probability, delay and overhead ratio values obtained are 0.3215, 6824 and 38 respectively. The eTMS outperformed this TMS in this case by 57% with respect to delivery probability.

4.8 Results of the iPRoPICMAN with TMS on the ABU Simulation Area

The results obtained from simulation of the iPRoPICMAN with the TMS in the ONE simulator on the ABU simulation area are as shown in Table 4.7.

Table 4.7 shows that over the simulation period of 0 to 43200s, the delivery probability varied between 0 and 0.5103 while the delay increased from 0 to 2736s. The overhead ratio also varied between 0 and 135.

Table 4.7: Message Statistical Report for the iPRoPICMAN with TMS Simulation on ABU Simulation Area

Sim_time (s)	Delivery Probability	Delay	Overhead ratio
0	0	0	0
1000	0.2424	331	68
2000	0.3881	732	73
4000	0.3507	963	85
6000	0.4505	1336	93
8000	0.4721	1575	103
10000	0.4543	1713	112
12000	0.4510	1847	115
16000	0.4114	2022	129
20000	0.4104	2257	130
25000	0.4314	2231	123
30000	0.4315	2267	126
35000	0.4650	2355	130
40000	0.5061	2400	132
43200	0.5103	2736	135

4.9 Results of the iPRoPICMAN with eTMS on the ABU Simulation Area

Similar to the case of the Helsinki simulation region. Several simulations were carried out on the TMS in order to obtain the best threshold value. The results are as presented in Table

4.8. The threshold values were varied from 0.1 to 1 in intervals of 0.1 and each simulation runtime was set at 43200s.

Table 4.8: Results of the Various Threshold Values using ABU Simulation Area

Threshold	Delivery Probability	Delay	Overhead ratio
0.1	-	-	-
0.2	-	-	-
0.3	0.4021	2623	134
0.4	0.4756	2637	137
0.5	0.5103	2609	134
0.6	0.5103	2736	135
0.7	0.5162	2584	133
0.8	0.5102	2567	134
0.9	0.4728	2683	134
1.0	-	-	-

It can be observed from Table 4.8 that when the threshold value was set at 0.1 to 0.2, the simulations timed out before 43200s and as such no results could be obtained. It was also observed that the highest delivery probability is obtained when the threshold is between 0.6 and 0.8. This interval coincides with that of the Helsinki simulation region. The overhead ratio varied slightly between 133 and 137 while the delay time also varied slightly between 2567 and 2736. The interval, 0.6 to 0.8, is further varied in steps of 0.01 (0.61 to 0.80) in order to obtain the best possible threshold value and it is observed to occur at about 0.75 as seen in Table 4.9.

Table 4.9: Result of Threshold Values between 0.61 and 0.80 for the ABU Simulation Area

Threshold	Delivery Probability	Delay	Overhead ratio
0.61	0.5103	2736	135
0.62	0.5074	2587	133
0.63	0.5162	2523	137
0.64	0.4926	2513	136
0.65	0.4985	2514	137
0.66	0.4989	2512	135
0.67	0.5044	2541	135
0.68	0.5044	2564	136
0.69	0.5103	2571	134
0.70	0.5162	2584	133
0.71	0.5133	2485	134
0.72	0.5074	2486	135
0.73	0.4956	2543	130
0.74	0.5032	2522	129
0.75	0.5221	2464	126
0.76	0.5015	2482	131
0.77	0.5103	2476	129
0.78.	0.5015	2510	132
0.79	0.5221	2633	128
0.80	0.5102	2567	134

It was observed from Table 4.9 that the highest delivery probability (0.5221) occurred at 0.75 and 0.79. Considering other metrics, 0.75 is chosen since it had a lower delay (2464s) and lower overhead ratio (126) compared with 2633s and 128 respectively for 0.79.

The obtained optimum threshold value of 0.75 is then used in the TMS to obtain the eTMS. The results obtained from simulation of the iPRoPICMAN with the eTMS in the ONE simulator on the ABU simulation area are as shown in Table 4.10.

Table 4.10: Message Statistical Report for the iPRoPICMAN with eTMS Simulation on ABU Simulation Area

	Delivery Probability	Delay	Overhead ratio
0	0	0	0
1000	0.2424	331	68
2000	0.3881	730	67
4000	0.4627	1072	83
6000	0.4799	1471	85
8000	0.4867	1636	91
10000	0.4921	1791	95
12000	0.5049	1998	101
16000	0.4923	2015	111
20000	0.4989	2129	116
25000	0.5001	2100	123
30000	0.5019	2169	122
35000	0.5123	2198	126
40000	0.5211	2359	125
43200	0.5221	2464	126

Table 4.10 shows that over the simulation period of 0 to 43200s, the delivery probability varied between 0 and 0.5221 while the delay increased from 0 to 2426s. The overhead ratio also varied between 0 and 126.

4.10 Comparison between the TMS and the eTMS on ABU Simulation Area

The performances (delivery probability, delay and overhead ratio) of the iPRoPICMAN with TMS and iPRoPICMAN with eTMS routing on the ABU simulation area are compared as shown in Figures 4.10 to 4.12. The plots are generated using MATLAB scripts of Appendices 4A to 4C.

From Figure 4.10, it can be seen that over the simulation period, the iPRoPICMAN with eTMS outperformed the iPRoPICMAN with TMS by 2% (considering the final values of 0.5103 and 0.5221) with respect to the delivery probability. The delivery probability of the ABU simulation region was more stable than that of the Helsinki simulation region due to the nature of the ABU map obtained. It concentrates the nodes to a particular part, thereby increasing the contact probability and making it resist changes in the event that occurred in the routing process.

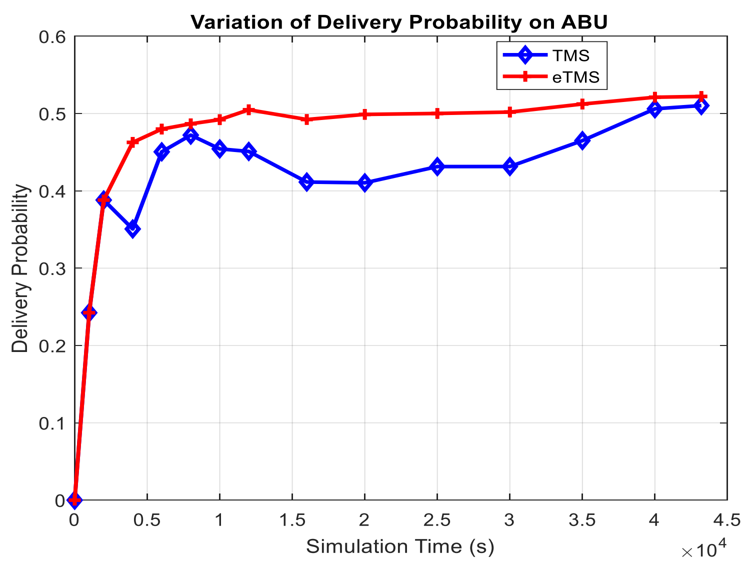

Figure 4.10: Variation of Delivery Probability with Time for the iPRoPICMAN with TMS and Etms on ABU

From Figure 4.11, it can be seen that over the simulation period, the iPRoPICMAN with eTMS had a reduced (10%) delay (considering the final values of 2736s and 2464s) compared with the iPRoPICMAN with TMS. This is consistent with Table 4.8, where the threshold value is shown to have some effect on the delay.

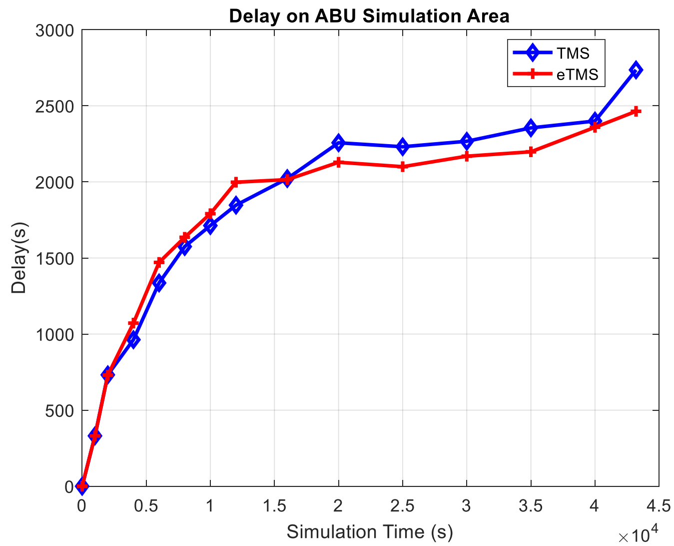

Figure 4.11: Variation of Delay with Time for the iPRoPICMAN with TMS and eTMS

From Figure 4.12, it can be seen that over the simulation period, the iPRoPICMAN with eTMS reduced the network overhead by 7% compared with the iPRoPICMAN with TMS. This is consistent with Table 4.8 where it is seen that the threshold value did not significantly affect the overhead ratio

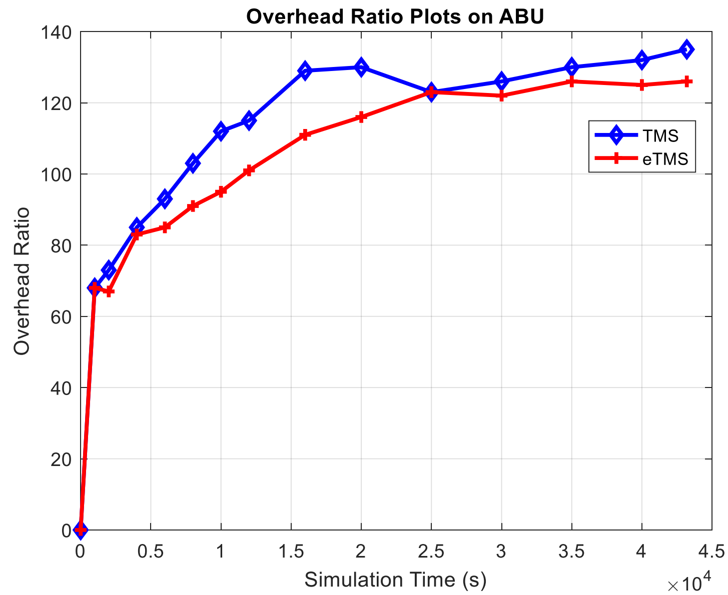

Figure 4.12: Variation of Overhead Ratio with Time the iPRoPICMAN with TMS and eTMS on ABU.

It has also been shown, using the ABU simulation area, that the threshold value is critical since it is even possible to obtain zero network performance.

CHAPTER FIVE

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

5.1 Summary

This research work presents the development of an eTMS for secured routing in oppnets based on the optimal selection of the trust threshold value (via simulation). The research showed that the trust threshold value is critical as it affects the performance of the network (since some values are shown to result in zero network performance). The iPRoPICMAN is a new variant of the PRoPICMAN that is developed in order to emulate a multi-hop scenario typical of oppnets and is implemented on the eTMS and its performances compared with those of the TMS. The iPRoPICMAN with eTMS is shown to outperformed the iPRoPICMAN with TMS using delivery probability, delay and overhead ratio using the Helsinki simulation area.

The iPRoPICMAN with eTMS is shown to outperformed the iPRoPICMAN with TMS using delivery probability, delay and overhead ratio using the ABU simulation area. The ABU simulation area was developed as a novel oppnet simulation area using the ONE simulator and OpenStreet

5.2 Conclusions

The iPRoPICMAN, which is a new variant of the PRoPICMAN (that considers a multi-hop scenario), was developed and simulated using the ONE simulator on the Helsinki simulation area.

The eTMS has been developed using an optimal t rust threshold value determined through simulations. The trust threshold value determines the success probability of building behaviour trust in the TMS model of Xi *et al*. (2015) and it has shown in this work to be critical (from results obtained in tables 4.4 and 4.8). This is because, over the simulation period, some of its values have been shown to produce zero network performance. For the Helsinki simulation area, an optimal threshold value of 0.73 was obtained.

The delivery probability, delay and overhead ratio for the PRoPICMAN are 0.2806, 4613s and 68 respectively, while those for the iPRoPICMAN are 0.4156, 5717s and 46 respectively, at the end of the simulation time of 43200s. The iPRoPICMAN increased the delivery probability by 48%, increased the delay by 23% and reduced overhead ratio by 32% when compared with the conventional PRoPICMAN. These results showed that the iPRoPICMAN eventually delivered more messages using less network resources even though it took longer time to do so.

The TMS and eTMS were also developed and simulated. The iPRoPICMAN with TMS outperformed the iPRoPICMAN by 19% with respect to delivery probability and 17% with respect to overhead ratio but had a 17% increment in delay. This showed that securing a network improved the performance of that network but with a trade-off in delay. The iPRoPICMAN with eTMS was shown to improve on the delivery probability (by 2%) while slightly reducing the delay (by 2%) when compared with the iPRoPICMAN with TMS. It is worth noting that the TMS threshold value was set at 0.6 while the experimentally determined optimum value was 0.73.

The ABU main campus was modeled into the ONE simulator as novel oppnet simulation area called the ABU simulation area and also used to run simulations of the TMS and the eTMS on the iPROPICMAN. The iPRoPICMAN with eTMS (with optimal trust threshold value of 0.75) outperformed the iPRoPICMAN with TMS using delivery probability, delay and overhead ratio as metrics by 2%, 10% and 7% respectively.

It is also worth noting that by their nature, oppnets have low delivery probabilities due to the associated long delays between transmissions. These delays are mostly due to the absence of connectivity between nodes (which make oppnets employ a store and forward communication mechanism).

5.3 Contribution to Knowledge

The following are the significant contributions to knowledge of this work:

1) The development of a new variant of the PRoPICMAN called the iPRoPICMAN, which is multi-hop based, that is ideal for the oppnet scenario and implemented on the developed eTMS (which uses the optimal trust threshold value determined via simulations) was achieved.
2) The performance evaluation of the iPRoPICMAN with eTMS on the Helsinki simulation area obtained an improvement in the delivery probability by 2%, reduction in delay by 2% with minimal change in the overhead ratio (using trust threshold value of 0.6 and the optimal value of 0.73).
3) The performance evaluation of the iPRoPICMAN with eTMS on the ABU simulation area (developed as a novel oppnet simulation area) obtained an improvement in the delivery probability over the TMS by 2%, reduction in delay by 10% and improvement

in the overhead ratio by 7% (using trust threshold value of 0.6 and the optimal value of 0.75).

5.5 Recommendations for Future Work

The following are recommended for consideration for further work:

1) Other methods of determining the trust threshold value can be explored (for example, using computational intelligence techniques) in order to compare it with the simulated threshold value.
2) Blacklist can also be developed in the TMS alongside the VFP in order to remove the detected malicious nodes.
3) Security threats (DoS attack, black hole attack, ID spoofing, etc.) can be introduced into the network inorder to see how the TMS behaves in the presence of these threats.

REFERENCES

Alajeely, M., Doss, R., & Ahmad, A. (2015). Security and Trust Opportunistic Network- A Survey. Journal presented to IETE Technical Review. 33(3), 256-268.

Aneja, M., & Garg, V. (2015). Simulation of Epidermic, Spray and Wait and First Contact Routing Protocols in Delay Tolerant Network. IOSR Journal of Electronics and Communication Engineering (IOSR-JECE). 30-34. ISSN: 2278-2834. http://www.iosrjournals.org/iosr-jece/papers/AETM'15_ECE/06-ECE-173.pdf

Asgari, C., Zareie, A., & Torkashvand, R. R. (2013). Intelligent Routing for Opportunistic Networks Based on Distributed Learning Automata. *Journal of Basic and Applied Science Research,* 3(7), 117-126.

Balaganesh, M., & Nalini, N. (2014). A Survey of Trust Based Automatic Routing in Delay Tolerant Networks. *International Journal of Advanced Information Science and Technology (IJAIST). 3 (12), 51-54*. DOI:10.15693/ijaist/2014.v3i12.51-54.

Barai, S. & Bhaumik, P. (2016). A Taxonomy of Recent Security Concerns in Opportunistic Networks. *International Journal for Scientific Research & Development.* 1(3), 34-46. ISSN: 2321-0613

Benamar, N., Singh, K.-D., Benamar, M., Ouadghiri, & D., Bonnin, J. (2014). Routing Protocols in Vehicular Delay Tolerant Networks: A comprehensive survey, *Computer Communications, Elsevier.* 48, 141–158.

Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K. & Weiss, H. (2007) Delay-Tolerant Networking Architecture. *Network Working Group Request for Comment: 4838, Category: Information. Published by Internet Research Task Force (IRTF).*

Chang, M., Chen, I., Bao, F., & Cho, J. (2011). Trust Threshold Based Routing in Delay Tolerant Networks. *5th International Conference on Trust Management (TM), Denmark, Springer, IFIP Advances in Information and Communication Technology*, AICT-358, 265-276.

Chen I. R., Bao F., Chang M., & Cho J. (2011). Integrated Social and QoS Trust-Based Routing in Delay Tolerant Networks. *In wireless personal communications, springer. 66,* 443-459.

Chen, I., Bao, F., Chang, M., & Cho, J. (2013). Supplemental Material for Dynamic Trust Management for Delay Tolerant Networks and its Application to Secure Routing. *IEEE Transaction on Parallel Distribution Systems. DOI:10.1109/tpds.2013.116,* 116-126.

Ciobanu, R. I., Marin R. C., Dobre, C., & Cristea V. (2016). Trust and Reputation management for Opportunistic Dissemination. *Pervasive and Mobile Computing.* 1-13. http://dx.doi.org/10.1016/j.pmcj.2016.09.016

Dinakar, S., R. M. Bhavadharini, & Karthik, S. (2013). Study of Opportunistic Network and MANET. *International Journal of Software & Hardware Research in Engineering, 1*(4).

D'Souza, R.J., & Jose, J. (2010). Routing Approaches in Delay Tolerant Network: A Survey. *International Journal of Computer Application.* 1(17), 8-14.

Fall, K. & Farrell, S. (2008). DTN: An Architectural Retrospective. *IEEE Journal on Selected Areas in Communication.* 26(5), 828-836.

Gao *et al*, *(*2013). Multidimensional Routing Protocol in Human-Associated Delay-Tolerant Networks. *IEEE Transactions on Mobile Computing*. 12(11), 2132-2144.

Goyal, M & Chaudhary M. (2013). Ensuring Privacy in Opportunistic Network. *IOSR Journal Computer Engineering*. 13(2), 74-82.

Gupta, S., Woungang, I., Dhurandher, S. K., Obaidat, M. S., & Kumar, A. (2013). A Trust-Based Security Protocol Against Blackhole Attack in Opportunistic Networks. *9th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. 724-729.

Hu, W., Xie, J., & Zhang, Z. (2013). Practical Opportunistic Routing in High-Speed Multi-Rate Wireless Mesh Networks. Proceedings of the 14th ACM International Symposium on Mobile Ad-hoc Networking and Computing (MobiHoc'13). 127-136. Bangalore, India — July 29 - August 01, 2013. doi>10.1145/2491288.2491310

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

http://www.netlab.tkk.fi/tutkimus/dtn/theone/down/one_1.6.0.tar.gz. March 29, 2015.

http://www.eclipse.org/downloads/ March 29, 2015.

https://netbeans.org/community/releases/73/index.html. March 29, 2015.

http://www.openstreetmap.com. October 05, 2017

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.*

Islam, M. A., & Waldvogel, M. (2011). Questioning flooding as a routing benchmark in Opportunistic Networks. *Paper presented at the Internet Communications (BCFIC Riga), 2011 Baltic Congress on Future*. 128-133.

Jain, S., Fall, K., & Patra, R. (2004). Routing in Delay Tolerant Network. *In Proceedings of Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.* 145-158.

Jingwei, M., Omar, H., Sonia, B. M., Lionel, B., Kangbin. (2013). An investigation on the unwilingness of nodes to participate in mobile delay tolerant network routing. *International Journal of Information Management*. 33(2). 252-262.

Jones, E. P., & Ward, P. A. (2006). Routing strategies for delay-tolerant networks. *Submitted to ACM Computer Communication Review (CCR)*.

Kattimani, S. L., & Indikar, J. N. (2015). Dynamic Trust and Security Management Protocol for Delay Tolerant Networks using Information Centric-Networks Architecture. *International Journal of Engineering and Computer Science*. 4(6). 12698-12704.

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

Kaur, M. (2015). Attacks in Opportunistic Networks. *International Journal of Computer Science and Mobile Computing.* 4(3). Pp. 21-26.

Keranen, A., & Ott, J. (2007). Increasing Reality for DTN Simulation. *Technical report, Helsinki University of Technology, Networking Laboratory.*

Keranen, A., & Ott, J. (2009). The ONE Simulator for DTN Protocol Evaluation. *Special report, Helsinki University of Technology, Networking Laboratory.*

Khot, A., & Mogal V. (2017). Trust Preservation in Opportunistic Networks. *International Journal of Innovative Research in Computer and Communication Engineering*. DOI: 10.15680/IJIRCCE.2017.0502121. 5(2). 2489-2494.

Kungwani, B. & Dudhe, P. (2016). Trust Based Privacy Preserving Friend Recommendation in Social Network Web. *International Journal of Research in Science & Engineering.* 2(5). 38-43.

Le, Z., Vakde, G., & Wright, M. (2009). PEON: Privacy-Enhanced Opportunistic Network with Applications in Assistive Environments. *Proceedings of the 2nd International Conference on Pervasive Technology related to Assistive environments.* 27. 1-7.

Lilien, L., Kamal, Z. H., Bhuse, V., & Gupta, A. (2006) a. Opportunistic networks: the concept and research challenges in privacy and security. *Proc. of the WSPWN*. 134-147. https://pdfs.semanticscholar.org/f040/f7688a05d118d72f8fe876979551560b2df8.pdf.

Lilien, L., Kamal, Z. H., & Gupta, A. (2006) b. Opportunistic networks: challenges in specializing the P2P Paradigm. *17th International Workshop on Database and Expert System Applications, an IEEE Conference Publication*. 72. 2-726.

Lin, Y., Li, B., & Liang, B. (2008). *CodeOR:* Opportunistic routing in wireless mesh networks with segmented network coding. *Paper presented at the Network Protocols,ICNP 2008. IEEE International Conference on.*

Lindgren, A., Doria, A., & Schelen, O. (2003). Probabilistic Routing in intermittently connected Network, *ACM Mobile Computing and Communications Review*, 7. 19-20.

Lindgren, A., Doria, A., & Schelen, O. (2004). Probabilistic Routing in intermittently connected Networks, *In Proceedings of SAPIR Fortalrnza, Brazil*.

Liu, L. (2014). A Survey on Reputation-based Incentive Mechanism in Opportunistic Networks. *Applied Mechanics and Materials Trans Tech Publications, Switzerland.* 543, 4288-4290.

Moreira, W., Mendes, P., & Sargento, S. (2011). Assessment model for opportunistic routing. *Paper presented at the Communications (LATINCOM), 2011 IEEE Latin-American 1Conference on,* 1-6.

Nelson, S., Bakht, M., & Kravets R., (2009). Encouter-based Routing in DTNs. *In INFOCOM IEEE Transact.* 846-854.

Nguyen, A. H., Giordano, S., & Puiatti, A. (2007). Probabilistic Routing Protocol for Intermittently Connected Mobile Ad hoc Network (PRoPICMAN). *In Proceedings of the IEEE International Symposium on World of Wireless Mobile and Multimedia Networks. Piscataway NJ: IEEE Press,* 1-6

Padhi, S., Tiwary, M., Priyadarshini, R., Panigrani, C. R., & Rachita, M. (2016). SecOMN: improved Security Approach for Opportunistic Mobile Networks Using Cyber Foraging. *3rd Internationall Conference on Recent Advances in Information Technology. IEEE*. 1-7.

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

Poonguzharselvi B. & Vetriselvi V. (2012). Trust framework for data forwarding in opportunistic networks using mobile traces. *International journal of wireless & mobile networks,* 4(6). 115-126.

Ristanovic, N. (2012). Modeling and Measuring Performance of Data Dissemination in Opportunistic Networks, PhD *Thèse École Polytechnique Fédérale De Lausanne.* 5448. http://infoscience.epfl.ch/record/180634/files/EPFL_TH5448.pdf

Shikfa, A., Onen M., & Molva R. (2010). Security Issues in Opportunistic Networks. *MobiOpp '10 Proceeding of the Second International Workshop on Mobile Opportunistic Networking.* 215-216.

Singh, D., Indora, S., Rani, A., & Sharma, A. (2015). Routing Policies and Strategies in Delay Tolerant Network. *International Journal of Engineering Research and Applications, National Conference on Advances in Engineering and Technology*. ISSN; 2248-9622.

Singh, R., & Chawla, M. (2014). An Efficient Trust Management Technique for Delay Tolerant Network. *International Journal of Computer Applications.* 98(21). 8-12.

Spyropoulos, T., Rais R. N., Turletti, T., Obraczka, K., & Vasilakos, A. (2010). Routing for Disruption Tolerant Networks: Taxonomy and Design. *Wireless Network Journal.* 16(8) 2349-2370.

Trifunovic, S. & Legendre, F. (2009). Trust in Opportunistic Networks. *Computer Engineering and Networks Laboratory*. 1-12.

Trifunovic, D. S., Legendre, F., & Anastasaides. C. (2010). Social Trust in Opportunistic Networks. *Infocom IEEE Conference on Communication Workshop.* DOI: 10.1109/INFCOMW.2010.5466696.

Trifunovic, S., Kouyoumdjieva, S. T., Distl, B., Pajevic, L., & Karlsson, G. (2017). A decade of Research in Opportunistic Network – Challenges, Relevance, and Future Directios. *IEEE Communication Magazine.* 55(1). 168-173.

Vahdat A. & Becker D. (2000). Epidemic Routing for Partially Connected Ad Hoc Networks. *A Technical Report VS-200006, Duke University*

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

Wang, D., Muller, T., Liu, Y., & Zhang, J. (2014). Towards Robust and Effective Trust Management for Security: A Survey. *13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), China. DOI: 10.1109/TrustCom*. 65. 65-73.

Wu, Y., Zhao, Y., Riguidel, M., Wang, G., & Yi, P. (2015). Security and Trust Management in Opportunistic Network: A Survey. *Security & Communication Networks in Online Library.* 8(9). 1812-1827.

Xi, C., Liang, S., Jianfeng, M.A., & Zhuo, M.A. (2015). A Trust Management Scheme Based on Behavior Feedback for Opportunistic Networks. *China Journal of Network Technology and Application*. 117-129.

Yao, L., Man, Y., Huang, Deng, J., & Wang, X. (2016). Secure Routing Based on Social Similarity in Opportunistic Networks. *IEEE Transactions on wireless communications.* 15(1). 594-605.

Yogi, M. K., & 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.

Zhang, Z. (2006). Routing in Intermittently Connected Mobile Ad hoc Networks and Delay Tolerant Networks: Overview and Challenges. *Communication Survey Tutorials.* 8(1). 24-37.

Zhao, Z., Mosler, B. R., & Braun, T. (2012). Performance evaluation of opportunistic routing protocols: a framework-based approach using OMNeT++. *Paper presented at the Proceedings of the 7th Latin American Networking Conference*, pp. 28-35.

APPENDICES

Appendix 1A

Message Stat Report

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import core.DTNHost;
import core.Message;
import core.MessageListener;
/**
 * Report for generating different kind of total statistics about message
 * relaying performance. Messages that were created during the warm up period
 * are ignored.
 * <P><strong>Note:</strong> if some statistics could not be created (e.g.
 * overhead ratio if no messages were delivered) "NaN" is reported for
 * double values and zero for integer median(s).
 */
public class MessageStatsReport extends Report implements MessageListener {
	private Map<String, Double> creationTimes;
	private List<Double> latencies;
	private List<Integer> hopCounts;
	private List<Double> msgBufferTime;
	private List<Double> rtt; // round trip times

	private int nrofDropped;
	private int nrofRemoved;
	private int nrofStarted;
	private int nrofAborted;
	private int nrofRelayed;
	private int nrofCreated;
	private int nrofResponseReqCreated;
	private int nrofResponseDelivered;
	private int nrofDelivered;
	/**
	 * Constructor.
	 */
	public MessageStatsReport() {
		init();
	}
	@Override
	protected void init() {
		super.init();
		this.creationTimes = new HashMap<String, Double>();
		this.latencies = new ArrayList<Double>();
		this.msgBufferTime = new ArrayList<Double>();
		this.hopCounts = new ArrayList<Integer>();
		this.rtt = new ArrayList<Double>();

		this.nrofDropped = 0;
		this.nrofRemoved = 0;
```

```
		this.nrofStarted = 0;
		this.nrofAborted = 0;
		this.nrofRelayed = 0;
		this.nrofCreated = 0;
		this.nrofResponseReqCreated = 0;
		this.nrofResponseDelivered = 0;
		this.nrofDelivered = 0;
	}
	public void messageDeleted(Message m, DTNHost where, boolean dropped) {
		if (isWarmupID(m.getId())) {
			return;
		}
		if (dropped) {
			this.nrofDropped++;
		}
		else {
			this.nrofRemoved++;
		}
		this.msgBufferTime.add(getSimTime() - m.getReceiveTime());
	}
	public void messageTransferAborted(Message m, DTNHost from, DTNHost to) {
		if (isWarmupID(m.getId())) {
			return;
		}
		this.nrofAborted++;
	}
	public void messageTransferred(Message m, DTNHost from, DTNHost to,
			boolean finalTarget) {
		if (isWarmupID(m.getId())) {
			return;
		}
		this.nrofRelayed++;
		if (finalTarget) {
			this.latencies.add(getSimTime() -
				this.creationTimes.get(m.getId()) );
			this.nrofDelivered++;
			this.hopCounts.add(m.getHops().size() - 1);

			if (m.isResponse()) {
				this.rtt.add(getSimTime() -	m.getRequest().getCreationTime());
				this.nrofResponseDelivered++;
			}
		}
	}
	public void newMessage(Message m) {
		if (isWarmup()) {
			addWarmupID(m.getId());
			return;
		}
		this.creationTimes.put(m.getId(), getSimTime());
		this.nrofCreated++;
		if (m.getResponseSize() > 0) {
			this.nrofResponseReqCreated++;
		}
	}
	public void messageTransferStarted(Message m, DTNHost from, DTNHost to) {
```

```
		if (isWarmupID(m.getId())) {
			return;
		}
		this.nrofStarted++;
	}
	@Override
	public void done() {
		write("Message stats for scenario " + getScenarioName() +
				"\nsim_time: " + format(getSimTime()));
		double deliveryProb = 0; // delivery probability
		double responseProb = 0; // request-response success probability
		double overHead = Double.NaN;	// overhead ratio
		if (this.nrofCreated > 0) {
			deliveryProb = (1.0 * this.nrofDelivered) / this.nrofCreated;
		}
		if (this.nrofDelivered > 0) {
			overHead = (1.0 * (this.nrofRelayed - this.nrofDelivered)) /
				this.nrofDelivered;
		}
		if (this.nrofResponseReqCreated > 0) {
			responseProb = (1.0* this.nrofResponseDelivered) /
				this.nrofResponseReqCreated;
		}
		String statsText = "created: " + this.nrofCreated +
			"\nstarted: " + this.nrofStarted +
			"\nrelayed: " + this.nrofRelayed +
			"\naborted: " + this.nrofAborted +
			"\ndropped: " + this.nrofDropped +
			"\nremoved: " + this.nrofRemoved +
			"\ndelivered: " + this.nrofDelivered +
			"\ndelivery_prob: " + format(deliveryProb) +
			"\nresponse_prob: " + format(responseProb) +
			"\noverhead_ratio: " + format(overHead) +
			"\nlatency_avg: " + getAverage(this.latencies) +
			"\nlatency_med: " + getMedian(this.latencies) +
			"\nhopcount_avg: " + getIntAverage(this.hopCounts) +
			"\nhopcount_med: " + getIntMedian(this.hopCounts) +
			"\nbuffertime_avg: " + getAverage(this.msgBufferTime) +
			"\nbuffertime_med: " + getMedian(this.msgBufferTime) +
			"\nrtt_avg: " + getAverage(this.rtt) +
			"\nrtt_med: " + getMedian(this.rtt)
			;
		write(statsText);
		super.done();
	}
}
```

Appendix 2A

The Program Class for PRoPICMAN

```
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 java.util.Random;

import routing.util.RoutingInfo;
import util.Tuple;
import core.Connection;
import core.DTNHost;
import core.Message;
import core.Settings;
import core.SimClock;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.NoSuchAlgorithmException;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.util.Arrays;
import java.util.Base64;

 //
 // Implementation of PRoPICMAN router as described in
 // <I> Probabilistic Routing Protocol for Intermittently Connected Mobile Ad hoc Network </I> by // Nguyen,
A. H. et al.
 //
public class ProPicmanRouter extends ActiveRouter {
	private double cons = 0.75;
   private int maxSetSize;

	/** ProPICMAN router's setting namespace ({@value})*/
	public static final String PROPICMAN_NS = "ProPicmanRouter";
	/**
	 * 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";
	public static final String FAMILIAR_SETTING ="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;

	private static SecretKeySpec secretKey;
private static byte[] key;

int weights;
String evidenc, value, mac;
int sn;

	//public Map<DTNHost, String> neighbourSet;
	public Map<DTNHost, Double> neighbourSet;

	protected double lambda;
	protected double gamma;
	protected double familiarThreshold;

	private Random random = new Random();
	private List<Double> probabilities;
private double[] cdf;

	/**
	 * Constructor. Creates a new message router based on the settings in
	 * the given Settings object.
	 * @param s The settings object
	 */
	public ProPicmanRouter(Settings s) {
		super(s);
		Settings propicmanSettings = new Settings(PROPICMAN_NS);
		secondsInTimeUnit = propicmanSettings.getInt(SECONDS_IN_UNIT_S);

		this.familiarThreshold = propicmanSettings.getDouble(FAMILIAR_SETTING);

		if (propicmanSettings.contains(BETA_S)) {
			beta = propicmanSettings.getDouble(BETA_S);
		}
		else {
			beta = DEFAULT_BETA;
		}

		initPreds();
	}

	/**
	 * Copyconstructor.
	 * @param r The router prototype where setting values are copied from
```

```
 */
protected ProPicmanRouter(ProPicmanRouter r) {
        super(r);

        this.familiarThreshold = r.familiarThreshold;

        neighbourSet = new HashMap<DTNHost, Double>();

        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);
        }
}

/* searching to peer with the next neighbour hop in the connection
 * check and another node in other to compute the evident and
 * forward the message
 * */

public void checkNodePro(Connection con) {

        DTNHost peer = con.getOtherNode(getHost());
        ProPicmanRouter peerC = (ProPicmanRouter) con.getOtherNode(getHost()).getRouter();

        // 2) check that the connection has met the time threshold. If so:
        if(this.neighbourSet.get(peer) >= this.familiarThreshold) {

                return;
        }

private void updateDeliveryPredFor(DTNHost host) {



        Map.Entry<DTNHost, Double> probabilities = null;
double evidenceValue = Double.MAX_VALUE;

        /* Prepare the next hop level for the computing the probability
         * of the evidence weight value
         * */
```

```
		if (preds.size() == 0) {
			preds.put(host, 1.0);
			return;
		}
		/* the for loop check for each matched value
		 * and compute the delivery probability to forward the message
		 * and return the value to probabilities variable  */


		for (Map.Entry<DTNHost, Double> entry : preds.entrySet()) {
			entry.setValue(entry.getValue() / (1+ cons));
  if (entry.getValue() < evidenceValue) {
    probabilities = entry;
    evidenceValue = entry.getValue();
  }

		}

		double probsValue = getPredFor(host) + cons;			//cons;
		preds.put(host, probsValue);

		if (preds.size() >= maxSetSize) {
  core.Debug.p("Probsize: " + preds.size() + " dropping " +
        preds.remove(probabilities.getKey()));
}

	}
	public double getPredFor(DTNHost host) {
		//methodDeliveryProbs(); // make sure preds are updated before getting
		if (preds.containsKey(host)) {
			return preds.get(host);
		}
		else {
			return 0;
		}
	}
	private void updateTransitivePreds(DTNHost host) {
		MessageRouter otherRouter = host.getRouter();
		assert otherRouter instanceof ProPicmanRouter : "PRoPICMAN only works " +
			" with other routers of same type";

		double pForHost = getPredFor(host);
		Map<DTNHost, Double> othersPreds =
			((ProPicmanRouter)otherRouter).getDeliveryPreds();

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

			double pOld = getPredFor(e.getKey()); // P(a,c)_old
			double pNew = pOld + ( 1 - pOld) * pForHost * e.getValue() * 0.90;// * beta;
			preds.put(e.getKey(), pNew);
		}
	}
	private Map<DTNHost, Double> getDeliveryPreds() {
```

```
		//methodDeliveryProbs();								//
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();
		for(Connection c : getConnections()) {
			DTNHost peer = c.getOtherNode(getHost());
			ProPicmanRouter		peerC		=		(ProPicmanRouter)
c.getOtherNode(getHost()).getRouter();

			if(this.neighbourSet.containsKey(peer)) {
				this.neighbourSet.put(peer, this.neighbourSet.get(peer) + 1);
			}
			else
				this.neighbourSet.put(peer, 1.0);
		}

		double simTime = SimClock.getTime(); // (seconds since start)
		double timeInDay = simTime % 120; // ever 120 seconds check local community information
with connected nodes
		if(timeInDay == 0) {
			for(Connection c : getConnections())
			{
				checkNodePro(c);
			}
		}

		// try messages to destinations
		tryOtherMessages();
	}


	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());
			ProPicmanRouter othRouter = (ProPicmanRouter)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
					}

					//I changed the condition of the probability > to >=
					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) { //(messages.size() == 0)
				return null;
			}

			// sort the message-connection tuples
			Collections.sort(messages, new TupleComparator());
			return tryMessagesForConnected(messages);		// try to send messages
		}


		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 = ((ProPicmanRouter)tuple1.getValue().

		getOtherNode(getHost()).getRouter()).getPredFor(tuple1.getKey().getTo());
				// -"- tuple2...
				double p2 = ((ProPicmanRouter)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() {
		//methodDeliveryProbs();
	//ageDeliveryPreds
		RoutingInfo top = super.getRoutingInfo();
		RoutingInfo ri = new RoutingInfo(preds.hashCode() +
				" delivery prediction(s)");			//preds.size()
preds.hashCode()

		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() {
		ProPicmanRouter r = new ProPicmanRouter(this);
		return r;
	}

}
```

Appendix 2B

The Program Class for the iPRoPICMAN

```
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 java.util.Random;

import routing.util.RoutingInfo;
import util.Tuple;
import core.Connection;
import core.DTNHost;
import core.Message;
import core.Settings;
import core.SimClock;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.NoSuchAlgorithmException;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.util.Arrays;
import java.util.Base64;

 //
 // Implementation of Improved PRoPICMAN router as a modification of what was //described in
<I>Probabilistic routing in intermittently connected networks</I> by
 // Nguyen, A. H. et al.
 //
public class ProPicmanImprovedRouter extends ActiveRouter {

	protected double lambda;
	protected double gamma;
	protected double familiarThreshold;

	private Random random = new Random();
	private List<Double> probabilities;
   private double[] cdf;

	/**
	 * Constructor. Creates a new message router based on the settings in
	 * the given Settings object.
	 * @param s The settings object
	 *
```

```
public ProPicmanImprovedRouter(Settings s) {
		super(s);
		Settings improvedpropicmanSettings = new Settings(IMPROVED PROPICMAN_NS);
		secondsInTimeUnit = propicmanSettings.getInt(SECONDS_IN_UNIT_S);

		this.familiarThreshold = propicmanSettings.getDouble(FAMILIAR_SETTING);

		if (propicmanSettings.contains(BETA_S)) {
			beta = propicmanSettings.getDouble(BETA_S);
		}
		else {
			beta = DEFAULT_BETA;
		}

		initPreds();
}

/**
 * Copyconstructor.
 * @param r The router prototype where setting values are copied from
 */
protected ProPicmanImprovedRouter(ProPicmanImprovedRouter r) {
		super(r);

		this.familiarThreshold = r.familiarThreshold;

		neighbourSet = new HashMap<DTNHost, Double>();

		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);
		}
}

/* searching to peer with the next neighbour hop in the connection
 * check other node in order to compute the evident and
 * forward the message
 * */
public void checkNodePro(Connection con) {
```

```
			DTNHost peer = con.getOtherNode(getHost());
			ProPicmanImprovedRouter		peerC		=		(ProPicmanImprovedRouter)
con.getOtherNode(getHost()).getRouter();

			// 2) check that the connection has met the time threshold. If so:
			if(this.neighbourSet.get(peer) >= this.familiarThreshold) {

				return;
			}
		}
		private void updateDeliveryPredFor(DTNHost host) {

			Connection node = null;

			Map.Entry<DTNHost, Double> probabilities = null;
	double evidenceValue = Double.MAX_VALUE;

			/* Prepare the next hop level for the computing the probability
			 * of the evidence weight value
			 * */
			if (preds.size() == 0) {
				preds.put(host, 1.0);
				return;
			}

			/* the for loop check for each matched value
			 * and compute the delivery probability to forward the message
			 * and return the value to probabilities variable */

			for (Map.Entry<DTNHost, Double> entry : preds.entrySet()) {
				entry.setValue(entry.getValue() / (preds.hashCode() + 1));
		//entry.setValue(entry.getValue() / (1+ cons));
	   if (entry.getValue() < evidenceValue++) {
		probabilities = entry;
		evidenceValue = entry.getValue();

	   }
			}
			//DTNHost peer = node.getOtherNode(getHost());
			//ProPicmanImprovedRouter		peerNode		=		(ProPicmanImprovedRouter)
node.getOtherNode(getHost()).getRouter();

			// 2) check that the connection has met the time threshold. If so:
			//if(this.neighbourSet.get(peer) >= this.familiarThreshold) {
			//	for(peerNode = host; peerNode <= this.neighbourSet.get(peer); peerNode++)
			//	{

			//	}

			//	return;
			//}
			double probsValue = getPredFor(host) + cons;			//cons;
			preds.put(host, probsValue);
			if (preds.size() >= maxSetSize) {
	   core.Debug.p("Probsize: " + preds.size() + " dropping " +
```

```
                preds.remove(probabilities.getKey()));
    }

        }
        public double getPredFor(DTNHost host) {
                //methodDeliveryProbs(); // make sure preds are updated before getting
                if (preds.containsKey(host)) {
                        return preds.get(host);
                }
                else {
                        return 0;
                }
        }

        private void updateTransitivePreds(DTNHost host) {
                MessageRouter otherRouter = host.getRouter();
                assert otherRouter instanceof ProPicmanImprovedRouter : "PRoPICMAN only works " +
                        " with other routers of same type";

                double pForHost = getPredFor(host);
                Map<DTNHost, Double> othersPreds =
                        ((ProPicmanImprovedRouter)otherRouter).getDeliveryPreds();

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

                        double pOld = getPredFor(e.getKey()); // P(a,c)_old
                        double pNew = pOld + ( 1 - pOld) * pForHost * e.getValue() * 0.90;// * beta;
                        preds.put(e.getKey(), pNew);

                }
        }

        private Map<DTNHost, Double> getDeliveryPreds() {
                //methodDeliveryProbs();                                          //
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();

                for(Connection c : getConnections()) {
                        DTNHost peer = c.getOtherNode(getHost());
```

```
			ProPicmanImprovedRouter	peerC	=	(ProPicmanImprovedRouter)
c.getOtherNode(getHost()).getRouter();

			if(this.neighbourSet.containsKey(peer)) {
				this.neighbourSet.put(peer, this.neighbourSet.get(peer) + 1);
			}
			else
				this.neighbourSet.put(peer, 1.0);
		}
		double simTime = SimClock.getTime(); // (seconds since start)
		double timeInDay = simTime % 120; // ever 120 seconds check local community information
with connected nodes
		if(timeInDay == 0) {
			for(Connection c : getConnections())
			{
				checkNodePro(c);
			}
		}

		// try messages to destinations
		tryOtherMessages();
	}
	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());
			ProPicmanImprovedRouter			othRouter			=
(ProPicmanImprovedRouter)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
				}

				//I changed the condition of the probability > to >=
				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) { //(messages.size() == 0)
			return null;
		}
```

```
		// sort the message-connection tuples
		Collections.sort(messages, new TupleComparator());
		return tryMessagesForConnected(messages);		// try to send messages
	}
	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 = ((ProPicmanImprovedRouter)tuple1.getValue().

	getOtherNode(getHost()).getRouter()).getPredFor(tuple1.getKey().getTo());
			// -"- tuple2...
			double p2 = ((ProPicmanImprovedRouter)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() {
		//methodDeliveryProbs();
	//ageDeliveryPreds
		RoutingInfo top = super.getRoutingInfo();
		RoutingInfo ri = new RoutingInfo(preds.hashCode() +
				" delivery prediction(s)");			//preds.size()
preds.hashCode()

		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() {
		ProPicmanImprovedRouter r = new ProPicmanImprovedRouter(this);
		return r;
	}
```

Appendix 2C

The Program Class for the TMS

```
package routing;

import java.io.*;
import java.net.*;
import java.security.*;
import java.security.spec.*;

import javax.net.SocketFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.security.cert.Certificate;
import javax.security.cert.CertificateEncodingException;
import javax.security.cert.X509Certificate;

import sun.security.x509.X500Name;

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 java.util.Random;

import routing.util.RoutingInfo;
import util.Tuple;
import core.Connection;
import core.DTNHost;
import core.Message;
import core.Settings;
import core.SimClock;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.KeyManagementException;
import java.security.KeyStore;
import java.security.NoSuchAlgorithmException;

import java.security.SignatureException;
import java.io.UnsupportedEncodingException;
```

```
import java.security.MessageDigest;
import java.util.Arrays;
import java.util.Base64;


public class ProPicmanImprovedRouterWithTMS extends ActiveRouter {

	protected double lambda;
	protected double gamma;
	protected double familiarThreshold;

	private Random random = new Random();
	private List<Double> probabilities;
   private double[] cdf;
   private int maxSetSize;

	/** ProPICMAN Improved router's setting namespace ({@value})*/
	public static final String PROPICMAN_NS = "ProPicmanRouter";
	/**
	 * 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";
	public static final String FAMILIAR_SETTING ="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;
   int weights;
   String evidenc, value, mac;
   int sn;

	//public Map<DTNHost, String> neighbourSet;
	public Map<DTNHost, Double> neighbourSet;

	protected double lambda;
	protected double gamma;

	protected double familiarThreshold;

	private Random random = new Random();
	private List<Double> probabilities;
   private double[] cdf;
```

```
/**
 * Constructor. Creates a new message router based on the settings in
 * the given Settings object.
 * @param s The settings object
 */

public ProPicmanImprovedRouterWithTMS(Settings s) {
	super(s);

	Settings propicmanSettings = new Settings(PROPICMAN_NS);
	secondsInTimeUnit = propicmanSettings.getInt(SECONDS_IN_UNIT_S);

	this.familiarThreshold = propicmanSettings.getDouble(FAMILIAR_SETTING);

	if (propicmanSettings.contains(BETA_S)) {
		beta = propicmanSettings.getDouble(BETA_S);
	}
	else {
		beta = DEFAULT_BETA;
	}

	initPreds();
}

/**
 * Copyconstructor.

 * @param r The router prototype where setting values are copied from

 */

protected ProPicmanImprovedRouterWithTMS(ProPicmanImprovedRouterWithTMS r) {

	super(r);

	this.familiarThreshold = r.familiarThreshold;

	neighbourSet = new HashMap<DTNHost, Double>();

	if ((new java.io.File("myprikey.dat")).exists()==false) {		// check if digital signed cert.
file imported if not check alternative

		if (generateTrustManagementKeyCertificate()==false) {// use Generate trusted key cert.
if fail console write fail

			System.out.println("Key Generation fail");

			return;

		};

	}

	try{

		java.io.ObjectInputStream	in=new	java.io.ObjectInputStream(new
java.io.FileInputStream("mypubkey.dat")); // import the security certificate key file
```

```
                PublicKey pubkey=(PublicKey)in.readObject();                    // read the public key

                in.close();

                System.out.println(pubkey.getFormat());

                in=new java.io.ObjectInputStream(new java.io.FileInputStream("myinfo.dat"));

                String info=(String)in.readObject();

                byte[] signed=(byte[])in.readObject();

                in.close();

                java.security.Signature signetcheck=java.security.Signature.getInstance("DSA");

                signetcheck.initVerify(pubkey);
    // check and verifying incoming signature

                signetcheck.update(info.getBytes());                            // specified byte array to update the data to be signed or verified.

                if (signetcheck.verify(signed))                                 // confirm the verification and the certificate is valid

                {

                    System.out.println("info="+info);
                    System.out.println("Certicicate Validated");
                }
                else  System.out.println("Validation Fail");
            }
            catch(java.lang.Exception e){

                    e.printStackTrace();
                    System.out.println("Caught exception " + e.toString());
            };
            try{
                java.io.ObjectInputStream in=new java.io.ObjectInputStream(new java.io.FileInputStream("myprikey.dat"));
                PrivateKey myprikey=(PrivateKey)in.readObject();
                in.close();
                String myinfo="Baseera";
                java.security.Signature signet=java.security.Signature.getInstance("DSA");
                signet.initSign(myprikey);
                signet.update(myinfo.getBytes());
                byte[] signed=signet.sign();
                System.out.println("signed(Certificate)="+ byte2hex(signed));

                java.io.ObjectOutputStream out=new java.io.ObjectOutputStream(new java.io.FileOutputStream("myinfo.dat"));
                out.writeObject(myinfo);
                out.writeObject(signed);
                out.close();
                System.out.println("Certicicate Validated");
            }
```

```
		catch(Exception e){

		System.err.println("Caught exception " + e.toString());

		};

		this.secondsInTimeUnit = r.secondsInTimeUnit;
		this.beta = r.beta;
		initPreds();
	}

	/**
	 * Initializes predictability hash
	 */

	private void initPreds() {


		this.preds = new HashMap<DTNHost, Double>();

	}
	public boolean generateTrustManagementKeyCertificate()
	{
		try {

			java.security.KeyPairGenerator
keygen=java.security.KeyPairGenerator.getInstance("DSA");

			keygen.initialize(1024);		//512
			KeyPair keys=keygen.genKeyPair();

			PublicKey pubkey=keys.getPublic();
			PrivateKey prikey=keys.getPrivate();

			java.io.ObjectOutputStream	out=new	java.io.ObjectOutputStream(new
java.io.FileOutputStream("myprikey.dat"));
			out.writeObject(prikey);
			out.close();

			System.out.println("prikeys ok");
			out=new				java.io.ObjectOutputStream(new
java.io.FileOutputStream("mypubkey.dat"));
			out.writeObject(pubkey);
			out.close();

			System.out.println("pubkeys ok");
			System.out.println("Node Secured");

			return true;
		}
		catch (java.lang.Exception e) {

		  e.printStackTrace();
		  System.out.println("Problem with generating key");
		  return false;
	  }
```

```
	}

	public String byte2hex(byte[] b) {
		String hs="";
		String stmp="";
		for (int n=0;n<b.length;n++)
		{
			stmp=(java.lang.Integer.toHexString(b[n] & 0XFF));
			if (stmp.length()==1) hs=hs+"0"+stmp;
			else hs=hs+stmp;
			if (n<b.length-1)  hs=hs+":";
		}
		return hs.toUpperCase();
    }

	@Override
	public void changedConnection(Connection con) {
		super.changedConnection(con);
		PrivateKey priKey;
		if (con.isUp()) {

			DTNHost otherHost = con.getOtherNode(getHost());

			updateDeliveryPredFor(otherHost);
			updateTransitivePreds(otherHost);
		}
	}

	/* searching to peer with the next neighbour hop in the connection
	 * check and other node in other to compute the evident and
	 * forward the message
	 * */

	public void checkNodePro(Connection con) {

		DTNHost peer = con.getOtherNode(getHost());
		ProPicmanImprovedRouterWithTMS    peerC    =    (ProPicmanImprovedRouterWithTMS)
con.getOtherNode(getHost()).getRouter();

		// 2) check that the connection has met the threshold. If so:
		if(this.neighbourSet.get(peer) >= this.familiarThreshold) {


			return;
		}

	}

	private void updateDeliveryPredFor(DTNHost host) {

		Connection node = null;

		Map.Entry<DTNHost, Double> probabilities = null;
        double evidenceValue = Double.MAX_VALUE;

		/* Prepare the next hop level for the computing the probability
```

```
		 * of the evidence weight value
		 * */
		if (preds.size() == 0) {
			preds.put(host, 1.0);
			return;
		}

		/* the for loop check for the each matched value
		 * and compute the delivery probability to forward the message
		 * and return the value to probabilities variable  */

		for (Map.Entry<DTNHost, Double> entry : preds.entrySet()) {
			entry.setValue(entry.getValue() / (neighbourSet.size() + hops));
    //(preds.hashCode() + 1)); //entry.setValue(entry.getValue() / (1+ cons));
   if (entry.getValue() < evidenceValue++) {
    probabilities = entry;
    evidenceValue = entry.getValue();

   }

		}
		//DTNHost peer = node.getOtherNode(getHost());
		//ProPicmanImprovedRouter peerNode = (ProPicmanImprovedRouter)
node.getOtherNode(getHost()).getRouter();

		// 2) check that the connection has met the time threshold. If so:
		//if(this.neighbourSet.get(peer) >= this.familiarThreshold) {
		//	for(peerNode = host; peerNode <= this.neighbourSet.get(peer); peerNode++)
		//	{

		//	}

		//	return;
		//}


		double probsValue = getPredFor(host) + cons;			//cons;
		preds.put(host, probsValue);
		if (preds.size() >= maxSetSize) {
   core.Debug.p("Probsize: " + preds.size() + " dropping " +
      preds.remove(probabilities.getKey()));
  }

	}
	public double getPredFor(DTNHost host) {
		//methodDeliveryProbs(); // make sure preds are updated before getting
		if (preds.containsKey(host)) {
			return preds.get(host);
		}
		else {
			return 0;
		}
	}
	private void updateTransitivePreds(DTNHost host) {
		MessageRouter otherRouter = host.getRouter();
```

```
		assert otherRouter instanceof ProPicmanImprovedRouterWithTMS : "PRoPICMAN only works " +
			" with other routers of same type";

		double pForHost = getPredFor(host);
		Map<DTNHost, Double> othersPreds =
			((ProPicmanImprovedRouterWithTMS)otherRouter).getDeliveryPreds();

		for (Map.Entry<DTNHost, Double> e : othersPreds.entrySet()) {
			if (e.getKey() == getHost()) {
				continue;
			}
		}
	}
	private Map<DTNHost, Double> getDeliveryPreds() {
		//methodDeliveryProbs();					// 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();

		for(Connection c : getConnections()) {
			DTNHost peer = c.getOtherNode(getHost());
			//ProPicmanImprovedRouterWithTMS		peerC		= (ProPicmanImprovedRouterWithTMS) c.getOtherNode(getHost()).getRouter();

			if(this.neighbourSet.containsKey(peer)) {
				this.neighbourSet.put(peer, this.neighbourSet.get(peer) + 1);
			}
			else
				this.neighbourSet.put(peer, 1.0);
		}


		double simTime = SimClock.getTime(); // (seconds since start)
		double timeInDay = simTime % 120; // ever 120 seconds check local community information with connected nodes
		if(timeInDay == 0) {
			for(Connection c : getConnections())
			{
				checkNodePro(c);
			}
		}
```

```
		// try messages to destinations
		tryOtherMessages();

	}


	private Tuple<Message, Connection> tryOtherMessages() {
		Signature certificate = null;
		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());

			ProPicmanImprovedRouterWithTMS othRouter =
(ProPicmanImprovedRouterWithTMS)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
				}


				//I changed the condition of the probability > to >=
				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) { //(messages.size() == 0)
			return null;
		}


		// sort the message-connection tuples
		Collections.sort(messages, new TupleComparator());
		return tryMessagesForConnected(messages);	// try to send messages


	}


	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 = ((ProPicmanImprovedRouterWithTMS)tuple1.getValue().

		getOtherNode(getHost()).getRouter()).getPredFor(tuple1.getKey().getTo());
				// -"- tuple2...
				double p2 = ((ProPicmanImprovedRouterWithTMS)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() {
			//methodDeliveryProbs();
		//ageDeliveryPreds
			RoutingInfo top = super.getRoutingInfo();
			RoutingInfo ri = new RoutingInfo(preds.hashCode() +
					" delivery prediction(s)");				//preds.size()
preds.hashCode()

			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() {
			ProPicmanImprovedRouterWithTMS r = new ProPicmanImprovedRouterWithTMS(this);
			return r;
		}

}
```

Appendix 3A

Matlab Script for the Delivery Probabilty of the PRoPICMAN and the iPRoPICMAN

```
clc
%Sim_time: Is the time of simulation
%PD_Pro: is the probability of delivery for Propicman

%PD_Imp:is the probability of delivery for Improved Propicman

Sim_time= [0 1000 2000 4000 6000 8000 10000 12000 16000 20000 25000 30000 35000 40000 43200];
PD_Pro= [0 0.0303 0.0299 0.2164 0.2376 0.2788 0.2861 0.2696 0.2583 0.2667 0.2719 0.2729 0.2758 0.2846
0.2806];
PD_Imp= [0 0.0303 0.0299 0.2239 0.2574 0.3234 0.3569 0.3505 0.3432 0.3407 0.3700 0.4049 0.4113 0.4169
0.4157];

plot(Sim_time,PD_Pro,'bd-','LineWidth',2)
hold on
plot(Sim_time,PD_Imp,'m+-','LineWidth',2)

title('Variation of Delivery Probability')
xlabel('Simulation Time (s)')
ylabel('Delivery Probability')
legend('Propicman','ImprovedPropicman')
grid on
```

Appendix 3B

Matlab Script for the Delay of the PRoPICMAN and the iPRoPICMAN

```
clc
%Sim_time: Is the time of simulation
%Delay_Pro: is the Delay for Propicman

%Delay_Imp:is the Delay for Improved Propicman

Sim_time= [0 1000 2000 4000 6000 8000 10000 12000 16000 20000 25000 30000 35000 40000 43200];
Dlay_Pro= [0 728 883 1841 2160 2652 2972 3175 3468 3762 3921 4116 4240 4456 4613];
Dlay_Imp= [0 728 735 1847 2191 2736 3244 3452 3761 4044 4729 5213 5384 5548 5717];

plot(Sim_time,Dlay_Pro,'bd-','LineWidth',2)
hold on
plot(Sim_time,Dlay_Imp,'m+-','LineWidth',2)

title('Delay plots')
xlabel('Simulation Time (s)')
ylabel('Delay(s)')
legend('Propicman','ImprovedPropicman')
grid on
```

Appendix 3C

Matlab Script for the Ovehead Ratio of the PRoPICMAN and the iPRoPICMAN

```
clc
%Sim_time: Is the time of simulation
%Overhead_Pro: is the overhead ratio for Propicman

%Overhead_Imp:is the for overhead ratio Improved Propicman

Sim_time= [0 1000 2000 4000 6000 8000 10000 12000 16000 20000 25000 30000 35000 40000 43200];
Overhead_Pro=[0 40 45 56 53 55 61 66 67 67 68 68 67 68];
Overhead_Imp=[0 35 40 47 43 43 47 50 52 50 46 46 46 46];

plot(Sim_time,Overhead_Pro,'bd-','LineWidth',2)
hold on
plot(Sim_time,Overhead_Imp,'m+-','LineWidth',2)

title('Overhead Ratio Plots')
xlabel('Simulation Time (s)')
ylabel('Overhead Ratio')
legend('Propicman','ImprovedPropicman')
grid on
```

Appendix 3D

Matlab Script for the Delivery Probabilty of the iPRoPICMAN and the iPRoPICMAN with TMS

```
clc
%Sim_time: Is the time of simulation
%PD_TMS: is the probability of delivery for the TMS

%PD_Imp:is the probability of delivery for Improved Propicman

Sim_time= [0 1000 2000 4000 6000 8000 10000 12000 16000 20000 25000 30000 35000 40000 43200];
PD_Imp= [0 0.0303 0.0299 0.2239 0.2574 0.3234 0.3569 0.3505 0.3432 0.3407 0.3700 0.4049 0.4113 0.4169
0.4157];
PD_TMS= [0 0.0303 0.0448 0.2015 0.2723 0.3494 0.3835 0.3750 0.3801 0.3926 0.4279 0.4690 0.4890 0.5026
0.4976];

plot(Sim_time,PD_Imp,'m+-','LineWidth',2)
hold on
plot(Sim_time,PD_TMS,'gd-','LineWidth',2)

title('Variation of Delivery Probability')
xlabel('Simulation Time (s)')
ylabel('Delivery Probability')
legend('ImprovedPropicman','TMS')
grid on
```

Appendix 3E

Matlab Script for the Delay of the iPRoPICMAN and the iPRoPICMAN with TMS

```
clc
%Sim_time: Is the time of simulation
%Delay_Imp:is the Delay for Improved Propicman
%Delay_TMS: is the Delay for trust management scheme

Sim_time= [0 1000 2000 4000 6000 8000 10000 12000 16000 20000 25000 30000 35000 40000 43200];
Dlay_Imp= [0 728 735 1847 2191 2736 3244 3452 3761 4044 4729 5213 5384 5548 5717];
Dlay_TMS= [0 728 953 1714 2285 2872 3401 3649 4173 4589 5314 5806 6294 6529 6713];

plot(Sim_time,Dlay_Imp,'m+-','LineWidth',2)
hold on
plot(Sim_time,Dlay_TMS,'gd-','LineWidth',2)

title('Delay plots')
xlabel('Simulation Time (s)')
ylabel('Delay(s)')
legend('ImprovedPropicman','TMS')
grid on
```

Appendix 3F

Matlab Script for the Ovehead Ratio of the iPRoPICMAN and the iPRoPICMAN with TMS

```
clc
%Sim_time: Is the time of simulation
%Overhead_Imp:is the for overhead ratio Improved Propicman
%Overhead_TMS: is the overhead ratio for trust management scheme

Sim_time= [0 1000 4000 6000 8000 10000 12000 16000 20000 25000 30000 35000 40000 43200];
Overhead_Imp=[0 35 40 47 43 43 47 50 52 50 46 46 46 46];
Overhead_TMS=[0 43 47 45 39 39 43 45 44 43 39 38 37 38];

plot(Sim_time,Overhead_Imp,'m+-','LineWidth',2)
hold on
plot(Sim_time,Overhead_TMS,'gd-','LineWidth',2)

title('Overhead Ratio Plots')
xlabel('Simulation Time (s)')
ylabel('Overhead Ratio')
legend('ImprovedPropicman','TMS')
grid on
```

Appendix 3G

Matlab Script for the Delivery Probabilty of the TMS ant the eTMS

```
clc
%Sim_time: Is the time of simulation
%PD_TMS: is the probability of delivery for the TMS

%PD_eTMS:is the probability of delivery for eTMS

Sim_time= [0 1000 2000 4000 6000 8000 10000 12000 16000 20000 25000 30000 35000 40000 43200];
PD_TMS= [0 0.0303 0.0448 0.2015 0.2723 0.3494 0.3835 0.3750 0.3801 0.3926 0.4279 0.4690 0.4890 0.5026
0.4976];
PD_eTMS= [0 0 0.0299 0.2313 0.2822 0.3457 0.3717 0.3848 0.3893 0.4044 0.4313 0.4788 0.4959 0.5107
0.5065];

plot(Sim_time,PD_TMS,'kd-','LineWidth',2)
hold on
plot(Sim_time,PD_eTMS,'m+-','LineWidth',2)

title('Variation of Delivery Probability')
xlabel('Simulation Time (s)')
ylabel('Delivery Probability')
legend('TMS','eTMS')
grid on
```

Appendix 3H

Matlab Script for the Delay of the TMS and the eTMS

```
clc
%Sim_time: Is the time of simulation
%Delay_TMS: is the Delay for the trust management scheme
%Delay_eTMS:is the Delay for the TMS

Sim_time= [0 1000 2000 4000 6000 8000 10000 12000 16000 20000 25000 30000 35000 40000 43200];
Dlay_TMS= [0 728 953 1714 2285 2872 3401 3649 4173 4589 5314 5806 6294 6529 6713];
Dlay_eTMS= [0 0 1117 1928 2295 2794 3295 3580 4195 4562 5258 5911 6228 6571 6607];

plot(Sim_time,Dlay_TMS,'Kd-','LineWidth',2)
hold on
plot(Sim_time,Dlay_eTMS,'m+-','LineWidth',2)

title('Delay plots')
xlabel('Simulation Time (s)')
ylabel('Delay(s)')
legend('TMS','eTMS')
grid on
```

Appendix 3I

Matlab Script for the Ovehead Ratio of the TMS and the eTMS

```
clc
%Sim_time: Is the time of simulation
%Overhead_TMS: is the overhead ratio for trust management scheme
%Overhead_eTMS:is the for overhead ratio eTMS

Sim_time= [0 1000 4000 6000 8000 10000 12000 16000 20000 25000 30000 35000 40000 43200];
Overhead_TMS=[0 43 47 45 39 39 43 45 44 43 39 38 37 38];
Overhead_eTMS=[0 0 40 43 40 41 42 44 43 43 38 37 37 37];


plot(Sim_time,Overhead_TMS,'kd-','LineWidth',2)
hold on
plot(Sim_time,Overhead_eTMS,'m+-','LineWidth',2)



title('Overhead Ratio Plots')
xlabel('Simulation Time (s)')
ylabel('Overhead Ratio')
legend('TMS','eTMS')
grid on
```

Appendix 4A

Matlab Script for the Delivery Probabilty of the TMS and the eTMS on ABU

```
clc
%Sim_time: Is the time of simulation
%PD_TMS: is the probability of delivery for the TMS on ABU

%PD_eTMS:is the probability of delivery for eTMS on ABU

Sim_time= [0 1000 2000 4000 6000 8000 10000 12000 16000 20000 25000 30000 35000 40000 43200];
PD_TMS= [0 0.2424 0.3881 0.3507 0.4505 0.4721 0.4543 0.4510 0.4114 0.4104 0.4314 0.4315 0.4650
0.5061 0.5103];
PD_eTMS=[0 0.2424 0.3881 0.4627 0.4799 0.4867 0.4921 0.5049 0.4923 0.4989 0.5001 0.5019 0.5123
0.5211 0.5221];

plot(Sim_time,PD_TMS,'bd-','LineWidth',2)
hold on
plot(Sim_time,PD_eTMS,'r+-','LineWidth',2)

title('Variation of Delivery Probability on ABU')
xlabel('Simulation Time (s)')
ylabel('Delivery Probability')
legend('TMS','eTMS')
grid on
```

Appendix 4B

Matlab Script for the Delay of the TMS and the eTMS on ABU

```
clc
%Sim_time: Is the time of simulation
%Delay_TMS: is the Delay for the trust management scheme
%Delay_eTMS:is the Delay for the TMS

Sim_time= [0 1000 2000 4000 6000 8000 10000 12000 16000 20000 25000 30000 35000 40000 43200];
Dlay_TMS= [0 331 732 963 1336 1575 1713 1847 2022 2257 2231 2267 2355 2400 2736];
Dlay_eTMS= [0 331 730 1072 1471 1636 1791 1998 2015 2129 2100 2169 2198 2359 2464];

plot(Sim_time,Dlay_TMS,'bd-','LineWidth',2)
hold on
plot(Sim_time,Dlay_eTMS,'r+-','LineWidth',2)



title('Delay on ABU Simulation Area')
xlabel('Simulation Time (s)')
ylabel('Delay(s)')
legend('TMS','eTMS')
grid on
```

Appendix 4C

Matlab Script for the Ovehead Ratio of the TMS and the eTMS on ABU

```
clc
%Sim_time: Is the time of simulation
%Overhead_TMS: is the overhead ratio for trust management scheme on ABU
%Overhead_eTMS:is the for overhead ratio eTMS on ABU

Sim_time= [0 1000 2000 4000 6000 8000 10000 12000 16000 20000 25000 30000 35000 40000 43200];
Overhead_TMS=[0 68 73 85 93 103 112 115 129 130 123 126 130 132 135];
Overhead_eTMS=[0 68 67 83 85 91 95 101 111 116 123 122 126 125 126];

plot(Sim_time,Overhead_TMS,'bd-','LineWidth',2)
hold on
plot(Sim_time,Overhead_eTMS,'r+-','LineWidth',2)



title('Overhead Ratio Plots on ABU')
xlabel('Simulation Time (s)')
ylabel('Overhead Ratio')
legend('TMS','eTMS')
grid on
```