Trust-based Energy Aware Geographical Routing for Smart Grid Communications Networks

Ming Xiang

A thesis submitted to
Auckland University of Technology
in partial fulfilment of the requirements for the degree
of
Mater of Computer and Information Sciences (MCIS)

2013

School of Computing and Mathematical Sciences

Table of Contents

List of Figures

List of Tables

Attestation of Authorship

"I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning."

Signature of Candidate: ……

Acknowledgements

I would like to express my deep and sincere gratitude to my supervisor, Dr William Liu, of the Department of Computing and Mathematical Sciences, Auckland University of Technology. His wide knowledge and logical way of thinking have been of great value for me. His understanding, encouragement and personal guidance have provided a good basis for the present thesis. I am also deeply grateful to my co-supervisor, Dr Quan Bai, for his great support whenever it was needed.

I owe my loving thanks to my parents. Without their encouragement and understanding it would have been impossible for me to finish this work.

A sincere thank is given to Network and Security Research Group (NSRG) for their valuable suggestions and feedback to my research and conference presentations. The financial support of the contestable fund provided by School of Computing and Mathematical Sciences Research Committee is gratefully acknowledged. This generous support enable me for the overseas trips to attend two IEEE/ACM distinguish conferences to present my accepted papers. The first trip to Tainan, Taiwan where I have attended the 3rd IEEE International conferences on Smart Grid Communication, so as to present my paper entitled: Trust-based Geographical Routing for Smart Grid Communication Networks in the workshop on Wireless Infrastructure for Smart Grid (WISG). The second trip was to Macau to attend the 2012 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, to present my paper entitled: Self-adjustable Trust-based Energy Efficient Routing for Smart Grid Systems in the workshop on Green Computing and Sustainable Society.

Publications, presentations and talks

Publications

Published

M. Xiang, W. Liu, Q. Bai, "Trust-based Geographical Routing for Smart Grid Communication Networks," IEEE Smart Grid Communications 2012 Workshop on Wireless Infrastructure for Smart Grid (WISG), ISBN: 978-1-4673-0909-7, pp. 704, Nov 2012, Taiwan.

M. Xiang, Q. Bai, W. Liu, "Self-adjustable Trust-based Energy Efficient Routing for Smart Grid Systems," The 2012 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology Workshop on Green Computing and Sustainable Society, pp. 379-382, Dec 2012, Macau.

Submitted and under review

M. Xiang, Q. Bai, W. Liu, "A Dynamic Fuzzy Based Energy Aware Trusted Routing Scheme for Smart Grid Network," International Journal of Communication Systems, submitted on 10th of Feb 2013 and under review.

Presentations

1. The 3rd IEEE International conferences on Smart Grid Communication, on Wireless Infrastructure for Smart Grid (WISG) workshop in Tainan, Taiwan. Topic: "Self-adjustable Trust-based Energy Efficient Routing for Smart Grid Systems". December 2012.
2. The 2012 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, on Green Computing and Sustainable Society (GCSS) workshop in Macau. Topic: "Self-adjustable Trust-based Energy Efficient Routing for Smart Grid Systems". November 2012.

3. The 2012 IEEE NZ Wireless workshop at AUT in Auckland CBD campus. Topic: “Energy-aware trust-based geo routing in smart grid wireless mesh network”. August 2012.

Talks

1. AUT post graduate mix and mingle presentation. Topic: “Trust your neighbours? - A trust-based geographical routing protocol in Smart Grid Communication network”. May 2012.
2. Network and Security Research Group (NSRG) weekly meeting, research progress report on 3 months literature reviews. May 2012.
3. NSRG weekly meeting, Dynamic Trust Elective Geo Routing (DTEGR). August 2012.
4. NSRG weekly meeting, Trust-based Intelligent Geo Elective Routing (TIGER). October 2012.

Abstract

Smart Grid is the trend of next generation power distribution and management network that enable interactive communication and operation between consumers and suppliers, so as to achieve intelligent resource allocation management and optimization. The wireless mesh network technology is a promising infrastructure solution to underpin and support these smart functionalities flexibly and scalably, as well as it can provide redundant routes for the smart grid communication network to guarantee the network availability. However, the wireless mesh network infrastructure is vulnerable to some cyber-attacks which need to be addressed. As the Smart Grid is heavily relying on the communication network, it makes these security concerns more critical. There are three major security concerns in the Smart Grid communication network such as network availability, data integrity, and information privacy. The previous security mechanisms such as cryptography, public key infrastructure (PKI), and authentication are the traditional ways to address these security concerns, but there is an emerging research area to discover the alternative solutions on trust and reputation mechanisms for wireless mesh network in Smart Grid environment.

In this thesis, we have proposed and implemented three trust-based geographical routing algorithms to tackle the cyber-attacks in Smart Grid, which are inspired from the existing Ambient Trust Sensor Routing (ATSR) algorithm. The first proposed algorithm, called as Trust-based Intelligent Geo Elective Routing (TIGER), is to resolve the time-insensitive problem in current ATSR by introducing the timestamp. Then the Dynamic Trust Elective Geo Routing (DTEGR) is proposed to tackle the inflexible weight factors selection problem of ATSR using a two-step selection strategy among the forwarding list of neighbours. At last, to advance the flexibility and scalability of TIGER and DTEGR algorithms, the Fuzzy-based Energy Aware Trust Geo Routing (FEATGR) is proposed by using the fuzzy logic approach. It can synthesize the trust, energy, and distance metrics to calculate one final score as the routing metric for determining the best route between source and destination nodes. The extensive simulation studies have confirmed that our new TIGER algorithm is more time-sensitive to detect then avoid the recent malicious attacks than the existing ATSR algorithm, and the DTEGR algorithm is able to achieve better routing performance in different network

scenarios by solving the weight factor issue. The FEATGR algorithm is flexible and scalable to maintain good network performance always upon dynamical network scenarios such as various attack patterns.

Chapter 1

Introduction

Intricate webs of interlinked critical infrastructure such as telecommunication and Internet, electricity and transportation networks are essential for the minimal functioning of contemporary societies and economies. Advances in information and communication technology (ICT) underpin the increasing interactions among these human-engineered networks. For example, the emerging higher expectations on the electrical power distribution and consumer systems are increasing every day, such as lower power price, more efficient and flexible usage, better power quality, smarter and interactive management, and also less carbon footprint impact on the environments, etc. [1 - 4]. The new Smart Grid paradigm is promising and regarded as the next generation electrical power network trend, which promises to meet the increasing requirements and satisfaction by our societies. It couples the power distribution network and communication networks as one smarter system, and they can perform two-way interactive information and control flows [2], so as to make intelligent decisions and optimization on electricity usages according to the state of the electrical power system and customer's dynamical needs [1].

There are a few examples to show how Smart Grid is intelligent, optimal resources and energy efficiency. The suppliers and consumers can observe the real time power usage and bill, while the legacy power system need to send personal workers to visit power meter physically and periodically, so as to read the meter collect usage manually. In such way, it will encourage consumers to better plan their power usage so as to save their power bill, also benefit the suppliers which they can better predict the power usage demand so as to optimize their resources and achieve energy efficiency. Smart Grid also can embed and manage the distributed renewable power sources such as solar, wind, hydro, etc. and non-renewable power like nuclear, gas, coal, etc. together into the Smart Grid. For example, the consumers can install the solar panel on the roof of their house to generate the clean power for their own use or even sell it back to their

power suppliers to reduce their power bills. In such way, it can achieve less carbon by using more renewable energy sources. The intelligent resource management example is like in a situation where a power outage occurs due to power cable cut accidently, the Smart Grid can automatically detect it and detour the power flow to avoid outage area so as to mitigate the outage damage. Moreover, Smart Grid will inform the technical department and urge the technician to be sent out on field as soon as possible.

As the Smart Grid is heavily relying on the underlie communication network to achieve these interactive functionalities and intelligent resource management, the concerns of reliability and security to the communication network are becoming more and more critical and essential. Generally, there are three major security concerns in the Smart Grid network, i.e., network availability, data integrity, and information privacy [1, 2, 5]. These security issues need to be addressed via security mechanisms, but unfortunately, most current existing Smart Grid frameworks lack of such focus at this beginning stage. Therefore, this thesis is emphasis on the secure communication and routing in Smart Grid communication network environment by using the trust-based geographical routing to detect and avoid the malicious attacks. Our novel contributions such as Trust-based Intelligent Geo Elective Routing (TIGER), Dynamic Trust Elective Geo Routing (DTEGR), and Fuzzy-based Energy Aware Trust Geo Routing (FEATGR) algorithms have been proposed, developed and verified by the extensive simulation results. These three routing solutions are valuable contributions to the research area of network availability in Smart Grid communication network.

1.1 Background

Smart Grid is the trend of next generation electrical power system that it enables the ability of inter-communication among electrical services, electrical units, and applications. The National Institute of Standards and Technology (NIST) defines Smart Grid standard in seven domains, which are market, customer, service provider, bulk generation, distribution, operation, and transmission in [6]. Smart Grid communication network underpins and connects these seven domains together with two-way information data transmission so as it enable the power grid to be smarter, intelligent, and also energy efficiency with optimal energy resource management.

Comparing with the traditional power grid, Smart Grid is the integration of information technology and power grid. As the communication network is becoming more and more crucial in Smart Grid, a high level on reliability and robustness on network connectivity is required to support these interactive electrical services and applications. The wireless mesh network (WMN) is a promising infrastructure solution for Smart Grid which it can provide the reliable connection as its nature on redundant routes provision, low cost scalability, and flexibility. The WMN can provide high-speed and reliable wireless transmission, as well as the mesh-like network topology enables the redundant communication path options to be selected. Therefore the WMN can conduct fault-tolerant and self-healing operations when the network failures e.g., node down or link cut occurs, and can adapt itself dynamically in an open error-prone wireless environment [7 - 10]. On the other hand, the WMN infrastructure has some security concerns due to its nature such as it broadcasts the data into the air which make it vulnerable to some cyber-attacks by adversaries.

1.2 Motivation

The objective of this thesis is to study the trust-based geographical routing in wireless mesh network for Smart Grid environment. It focuses on the network availability concerns in wireless mesh network infrastructure for Smart Grid environment by using the trust-aware routing mechanisms to avoid any malicious attacks. The main goal is to quantitatively define and model a trust-aware routing metric so as to make the malicious activities in WMN and smart grid environment can be measured, detected and avoided. At the same time, this trust-aware routing metric can also make the best use of geographical information to find the shortest path i.e., packet delay to the destination.

Smart Grid is the integration of existing power grid and the communication network, and it heavily relies on the communication network to enable the intelligence to the current existing power grid. As we introduced in previous, the WMN infrastructure is one of promising solutions for the Smart Grid communication network. However, it has some security issues to be addressed. In this peer to peer oriented wireless infrastructure, each electrical unit can be considered as an individual self-organized node, and thus the trust is an important factor for determining the relationships among the nodes. There are numerous studies on trust-based secure

routing for wireless network, and the Ambient Trust Sensor Routing (ATSR) [11] is one of the most popular trust-based routing protocols. It uses the trust-aware routing metric to detect and avoid the malicious nodes, then use distance information to locate the direction to the destination node. We have identified two major problems in current ATSR algorithm i.e., time-insensitive and inflexible weight factors selection problems.

The first problem we have identified in ATSR is time-insensitive in recent attacks. In ATSR, the calculation of trust metric is simply using total number of good experiences divided by total number of experiences. In such case, the malicious nodes can accumulate a large amount of good experiences at the beginning so as to make the algorithm ignore their recent malicious actions. Namely it is time-insensitive on the time scale of the node behaviour records.

The second identified problem in ATSR is its inflexible weight factors selection problem. The ATSR algorithm uses the weight factors to combine the trust-aware metric and distance-aware metric into a final score, and then the algorithm will select the neighbour node with the highest final score as next hop to forward the packet to the destination node. While these weight factors are static values which are correlated strongly to the network scenarios such as topology, traffic demands and attack patterns. It cannot adjust itself automatically to handle the dynamical changes in network scenarios. In other words, whenever the network scenario changes, it has to manually re-calculate these weight factors based on the new scenario. Otherwise, the network performance such as the sensitivity on detecting and avoiding malicious nodes will be much degraded.

Aiming at tackling the first problem of time-insensitivity in ATSR, we propose a novel Trust-based Intelligent Geo Elective Routing (TIGER) by introducing timestamp technique into the routing algorithm. For solving the second problem of inflexible weight factors selection, we have proposed the Dynamic Trust Elective Geo Routing (DTEGR) algorithm by using a two-step selection strategy among the forwarding list of neighbours. The extensive simulation results have proven that the TIGER algorithm becomes more sensitive to detect and avoid the most-recent attacks than the ATSR. In addition, the DTEGR is more capable to handle different network scenarios and solve the inflexible weight factor selection problem. However, for adding new metric such as energy metric into the algorithm, the DTEGR is still using weight factors to combine multiple metrics which motivate us to advance it using fuzzy logic approach, i.e., the

Fuzzy-based Energy Aware Trust Geo Routing (FEATGR) which resolves the multi-metrics integration problem.

1.3 Contributions

This thesis focuses on tackling the research problems of malicious attacks in wireless mesh network infrastructure for Smart Grid environments. We have proposed and developed trust-aware routing algorithms to detect and avoid effectively any malicious attacks as well as utilize geo-graphical information to locate the optimal direction to destination node. Overall, we have three major contributions to this field.

Firstly, we have proposed a Trust-based Intelligent Geo Elective Routing (TIGER) algorithm to tackle the time-insensitive problem in current ATSR algorithm [11]. Comparing to the ATSR, the TIGER introduces two different timestamp methods to prevent the malicious nodes which accumulate a large amount of good experiences at beginning, so as to make the ATSR algorithm ignore their most recent malicious behaviours. The extensive simulation results have confirmed this.

Secondly, we have developed the Dynamic Trust Elective Geo Routing (DTEGR) routing algorithm to resolve the inflexible weight factors selection problem. There are many existing research studies in trust-based routing protocol to use weight factors for synthesizing different metrics into a final score as the routing metric. For example, the ATSR is a typical trust-based geographical routing protocol that it uses the weight factors to combine the trust metric and distance metric into a final score, so it can select the neighbour with the highest final score as the next hop to forward packet until reaching the destination node. However, a static set of weight factors cannot handle all the different network scenarios. For example, in a network scenario that the sources node is surrounded by malicious nodes on the direction to the destination node. In this scenario, the algorithm requires priority on trust by increasing its weight factor to avoid the malicious nodes but might scarifying longer packet delay because it need to detour to the most trustable path not shortest path. While in another network scenario which there are not many malicious nodes allocated between the source and destination nodes, the algorithm thus requires less priority on trust metric but higher priority on distance by increasing the weight factor on distance metric, so the shorter path can be

selected and the packet latency can be minimized as much as possible. Therefore, the DTEGR is proposed to separate the evaluation on the trust metric and distance metric using a two-step strategy. The DTEGR set up a threshold value for trust metric firstly to determine good nodes and malicious nodes, and then generate a trustable neighbour nodes forwarding list. Then the algorithm selects the next hop node only among the nodes listed in the forwarding list by using the extra distance metric. In such case, DTEGR can select the shorter path after blacklisting the malicious nodes, so as to handle different network scenarios. For the network availability concern, there might be a probability that the threshold value is configured too high to find a neighbour node listed in the forwarding list. In such case, the algorithm can automatically degrade the threshold value until the forwarding list is not empty.

Thirdly, we have developed the Fuzzy-based Energy Aware Trust Geo Routing (FEATGR) routing algorithm to advance the DTEGR algorithm. Although DTEGR solved the problem of inflexible weight factor selection between trust metric and distance metric, it still use weight factor to synthesize direct trust and indirect trust metrics for calculating the overall trust value. In addition, if a new metric need to be considered and put into the algorithm such as energy consumption, it will degrade significantly the sensitivity level of existing metrics so as to make the new metric take effect in the algorithm by using the linear weight factor technique. In such case, the fuzzy logic approach can be used to develop a better solution to synthesize different metrics. Therefore, the FEATGR is proposed as a trust-aware routing algorithm by using the fuzzy logic approach. It can add new metric with the minimum sacrifice on the existing metrics in a scalable way. The extensive simulation studies have confirmed that the FEATGR can achieve better or at least similar performance as the DTEGR algorithm.

1.4 Thesis structure

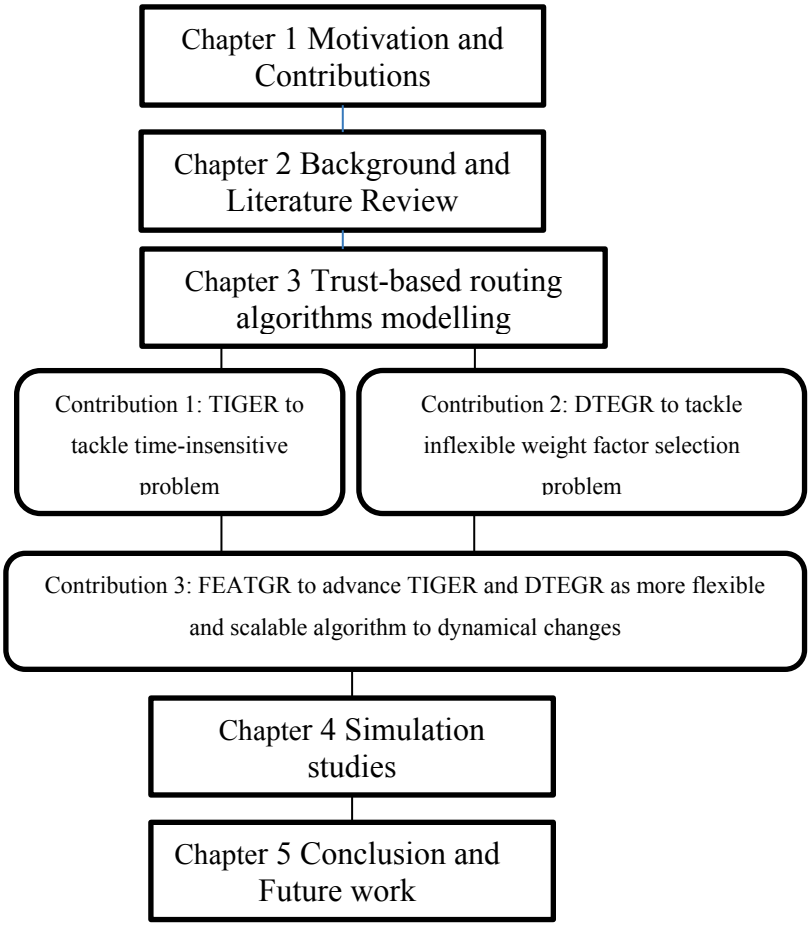

Figure 1.1 - Thesis structure

The thesis's structure is depicted in Figure 1.1 above. The remainder of this thesis contains an introduction of relevant background knowledge, followed by description and comprehensive discussion of three trust-aware routing algorithms. The extensive simulation studies are then presented to validate and compare these three new algorithms. The chapters of the thesis are organized as follow:

Chapter 2 gives a detailed background introduction of Smart Grid technology and its typical network infrastructure. The three major security concerns in Smart Grid and the related cyber-attacks for each concern are then introduced. Furthermore, an overview on the new concept of trust is presented based on the existing literature studies. Specially, the existing trust-based routing algorithms such as the benchmark algorithm of ATSR is introduced and then discussed in details.

Chapter 3 has identified the weaknesses in the benchmark algorithm of ATSR. By tackling these weaknesses, we have proposed three novel trust-based geographical routing algorithms. They are Trust-based Intelligent Geo Elective Routing (TIGER), Dynamic Trust Elective Geo Routing (DTEGR), and Fuzzy-based Energy Aware Trust Geo Routing (FEATGR) algorithms The TIGER algorithm has resolved the time-insensitive problem by using timestamp technique. The DTEGR algorithm can separate the evaluation on the trust metric and distance metric using a two-step strategy, so as to handle different network scenarios. Finally, the FEATGR algorithm uses the fuzzy logic approach to advance the TIGER and DTEGR algorithms, so as to synthesize all the metrics as a final score for routing. It is flexible and scalable to add new metric with minimum sacrifice on the sensitivity level of existing metrics. This makes the routing algorithm more intelligent and has human-like decision making to handle dynamical network scenarios.

Chapter 4 presents the extensive simulation studies and discussion on TIGER, DTEGR, and FEATGR algorithms by comparing with the benchmark ATSR algorithm. The simulation studies have confirmed that the TIGER algorithm can maintain the sensitivity level of detecting malicious activities for a long term and time-sensitive on the recent attacks, where the ATSR was failed to do it. In addition, the DTEGR has been validated to able to maintain performance level to handle different network scenarios, but ATSR has to use different trust weight factors manually to achieve the best performance in different scenarios. Finally, we add energy consumption metric to compare FEATGR and DTEGR algorithms with different network scenarios. The FEATGR has been proven itself as a promising solution for multi-metrics decision making algorithm.

Finally, the contribution and findings are concluded in Chapter 5. In addition, we suggest some possible research directions to advance our algorithms in the future work.

Chapter 2

Background

In this chapter, we give an overview on Smart Grid technology and its typical underlies network infrastructures. The three major security concerns in Smart Grid and the related cyber-attacks for each concern are then introduced and bring out the research problems. Furthermore, an overview on the new concept of trust is presented based on the existing literature studies and associated key technologies. Specially, the existing trust-based routing algorithms such as the benchmark algorithm of ATSR is introduced and then discussed in details.

The remainder of this chapter is organized as follows. In section 2.1, we give an overview of Smart Grid technology and its communication network infrastructure. In addition, this section addresses the major security concerns under Smart Grid environment. Section 2.2 describes the traditional security approaches, and discusses the challenges of these traditional security approaches to be adopted into the Smart Grid communication network. Section 2.3 reviews the existing routing protocols for the wireless ad-hoc network. Finally, the section 2.4 introduces the umbrella concept of trust in wireless mesh network for smart grid environment and presents some existing trust-based routing protocols including the benchmark algorithm of ATSR.

2.1 Overview of Smart Grid

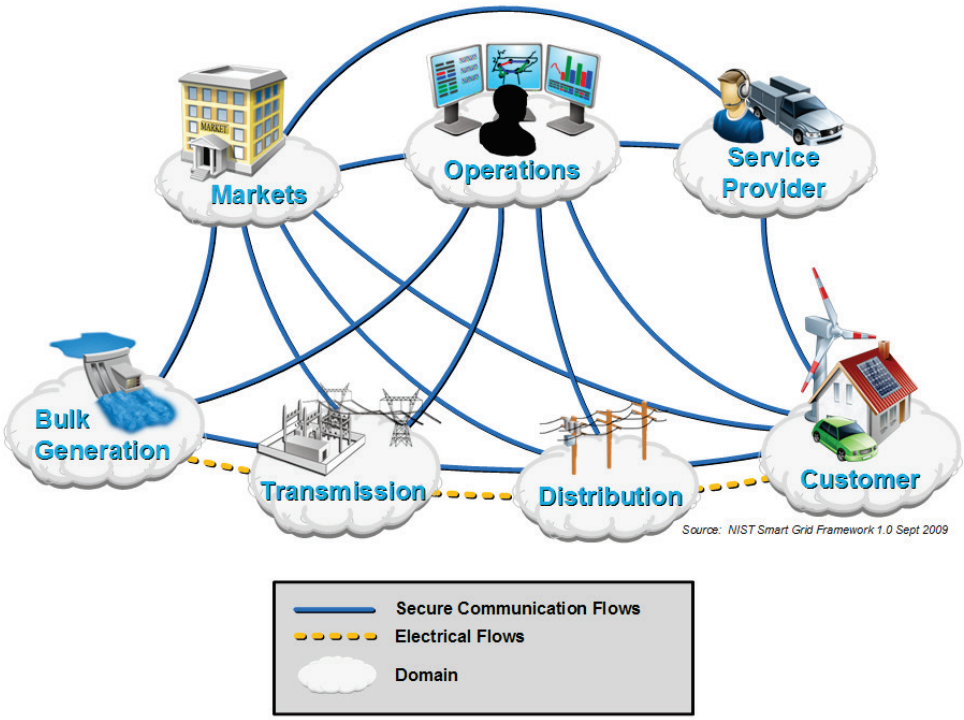

Figure 2.1 - Smart Grid conceptual model [6]

The National Institute of Standards and Technology (NIST) defines Smart Grid standard in seven domains, i.e., market, customer, service provider, bulk generation, distribution, operation, and transmission in [6]. These seven domains interact with each other through the secured Smart Grid communication network. Moreover, bulk generation, distribution, transmission, and customer domain will also involve two-ways power flow between each other to achieve energy efficiency and low carbon footprint by using renewable energy effectively. These seven domains and their relationships are shown in Figure 2.1 as above.

The market domain manages all participants in the electricity market in Smart Grid which it involves retailing, wholesaling, and trading services. It also exchanging information such as billing, customer profiles between different services providers.

The customer domain in Smart Grid is the end-user of the electricity. These end-users can be residential home users, commercial business users, or industrial factory users. Each of these end-users has a smart meter to manage the power flow. For example, a home user installs a solar panel on the roof to collect renewable energy and

sell it back to the service provider. The smart meter also records the electricity consumption and customer usage behaviour information, and exchanges this information with different domains such as service provider.

The service provider domain handles all third parties operation in the Smart Grid such as the two-ways communication between customers and electricity management utilities, electricity demand optimize applications, outage management.

The bulk generation domain is about all the different types of energy sources in Smart Grid. These energy sources can be solar power plant, nuclear power plant, hydro power plant, and anything supply energy. These energy sources can also be classified as renewable energy which the energy is generated from natural source, as well as non-renewable energy which the sources cannot be reproduced and it is consumable. In addition, the non-variable energy is defined as the energy sources able to generate energy consistently, while variable energy is the sources have inconsistency energy generation.

The distribution domain manages the energy distribution between sources and end-users through the power grid coupled with the communication network in smart grid. For example, an end-user can sell back his energy to the service providers, and the consumers in one city can buy the electricity generated by the nuclear power plant located in another city.

The operation domain controls the energy flow and manages the intelligent utilities such as smart meters, substation, etc. through the communication network in Smart Grid. The operation domain provides controlling, supervision, reporting functionality and related services through these intelligent utilities. The last transmission domain is the actual energy and communication data transmission media.

2.1.1 Typical network infrastructures

IEEE has defined three foundational layers for Smart Grid and they are energy and power layer, communication layer, and IT/computer layer in [6]. In this thesis, we focus on the communication layer.

The hierarchical communication structure is the typical infrastructure adopted in Smart Grid communication network. The layers can be defined by geographical size, i.e.

Wide Area Network (WAN), Neighbourhood Area Network (NAN), and Home Area Network (HAN) in [4]. It can also be defined by domain, i.e. HAN, Business Area Network (BAN), NAN, data centre, and substation automation integration system [12].

Smart Grid integrates the bulk power generation sources and then distribute them to the end-users from WAN to NAN, and then from NAN to HAN. At the same time, the end-users can conduct two ways communication to service providers (i.e., data centre or substation automation integration system) through NAN to WAN. In this way, the end-users will be encouraged to better manage their power usage. On the other hand, the service providers can collect end-users usage data to optimize their resources and then allocate power supply in an efficient way to achieve energy efficiency.

2.1.1.1 Home area network

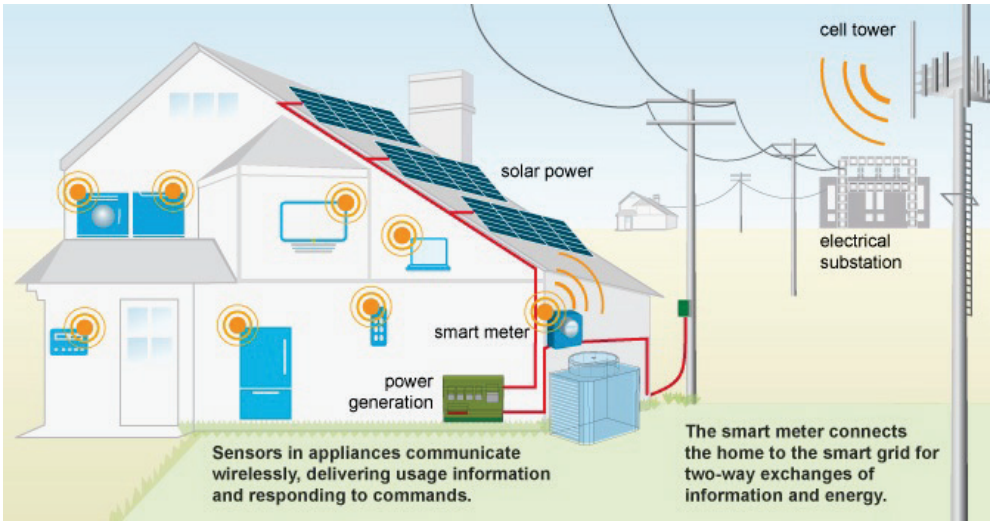

Figure 2.2 - Home Area Network [41]

As show in Figure2.2 above, the bottom layer of Smart grid is called as home area network (HAN). It is the local network within the house area. There are different wireless devices in the house connected to the smart meter so as to collect and report real time usage information. Three modules are consisted in HAN. The service module provides real time power usage information, the metering module records the power usage history, and the last one is management module which manages previous two modules and also in charge of the communication between HAN and NAN [4].

2.1.1.2 Neighbourhood area network

The neighbourhood area network (NAN) is formed by many HANs within a small region. Under the wireless mesh network infrastructure for Smart Grid environment, it

has wireless access point i.e., base station within the NAN for the whole community. The smart meter in HAN sends data to its neighbours and requests its neighbours to forward the data continually and eventually reach the base station (gateway). The base station collects the data then forwards to the service provider through the WAN. On the other way, the base station is also responsible for collecting data from services providers and forwards it to the targeted smart meter in HAN. In such case, the consumers and service providers can have two-way communication. However, for security reasons, the direct interactions among HANs are not allowed. Namely, they have to communicate through the backbone servers. The HANs within the same NAN can be bridged via Zigbee and Wi-Fi connections.

2.1.1.3 Wide area network

The wide area network (WAN) is composited with a number of NANs. Obviously, Zigbee and Wi-Fi are not suitable for WAN due to the limitation of their transmission distances. The power line communication (PLC) has been recommended as a possible solution as it uses the existing power grid infrastructure with the power feeder lines. This layer consists of an energy distribution system which is conducting the overall energy and metering data distribution. The supervisory control and data acquisition (SCADA) is used here to control all the nodes or utilities in smart grid network, and also the energy and service corporations, so as to make a power price judgment [4].

2.2.2.4 Wireless Mesh Network Infrastructure in Smart Grid

The wireless mesh network is one of popular network infrastructure solutions for the Smart Grid communication network due to its reliability, robustness, and cost efficiency [7, 14]. The study in [14] has categorised the wireless mesh network as three different architectures. They are infrastructure wireless mesh network, client wireless mesh network, and hybrid wireless mesh network. In infrastructure wireless mesh network, the nodes are the mesh routers with minimal mobility to provide backbone services for the clients in the network. The client wireless mesh network is having the clients directly communicate with each other like wireless ad-hoc network and actually they are the same. The hybrid wireless mesh network is the combination of the infrastructure wireless mesh network and clients wireless mesh network in a hierarchical form. The hybrid wireless mesh network architecture is the recommended wireless mesh network infrastructure for the Smart Grid communication network.

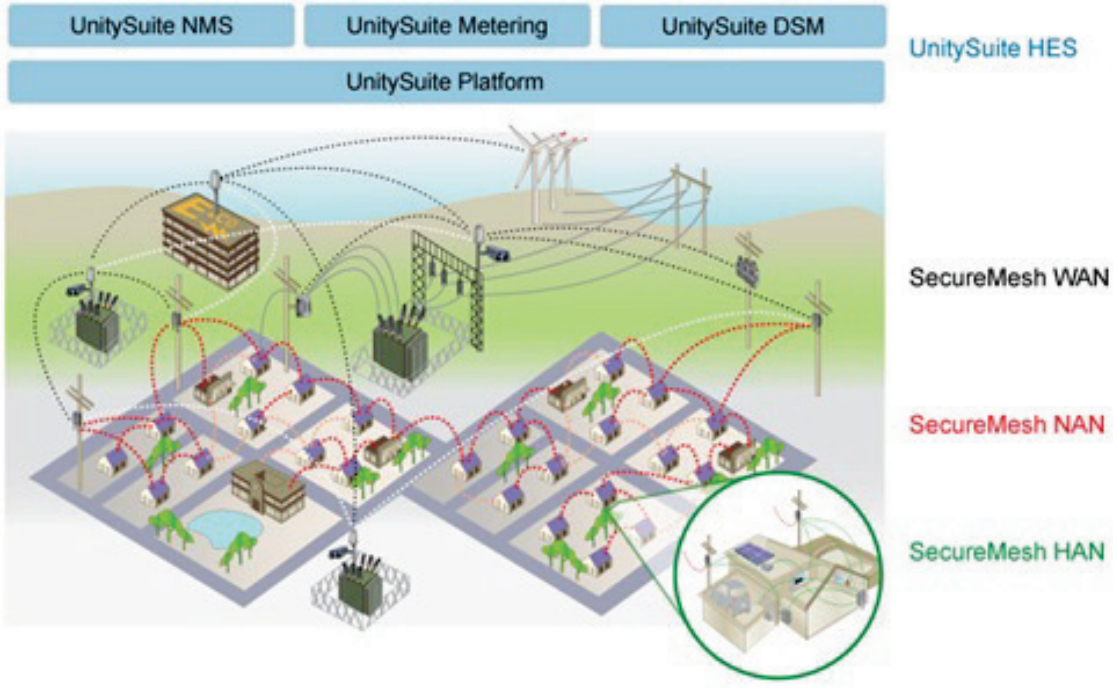

Figure 2.3 - Wireless Mesh Network Infrastructure in Smart Grid [42]

The wireless mesh network topology in Smart Grid can be seen in Figure 2.3 above. Some studies [7, 8] have recommended the wireless mesh network infrastructure for the Smart Grid communication network, and many successful cases are demonstrated in [13]. All of these are showing that wireless mesh network infrastructure is a promising solution to Smart Grid. Therefore, in this thesis, we focus on tackling some problems related to the wireless mesh network infrastructure for Smart Grid communication network.

2.1.2 Major security threats

As communication network is playing a crucial role in Smart Grid, thus its related security concerns should be addressed. The Electric Power Research Institute (EPRI) has pointed out that the cyber security issue is one of the biggest challenges for Smart Grid deployment [15]. There are three major threats in the smart grid network, i.e., network availability, data integrity, and information privacy, which are ordered from the most critical to less critical [1, 2, 5].

2.1.2.1 Network availability

Network availability is the most critical security objective in the Smart Grid communication network, as the power system requires uninterruptable network connection to collect usage information and deliver services for any situations. The loss of connectivity thus the loss of control to the system in an emergency situation could cause critical safety issues [1]. For example, if a transformer is overloaded and need to be shut down immediately, the loss of control to the system will make the transformer stay on overloaded and it may lead to an explosion. For wireless mesh network infrastructure, there are a few typical cyber-attacks can lead to network availability failures. Those attacks are black-hole attack, grey-hole attack and flooding attack [16, 17]. The black-hole attack is a malicious behaviour which it drops all the packets received from others. The grey-hole attack randomly or purposely drops some of the packets rather than all of the packets received. The flooding attack is defined as a malicious node spamming huge amount of non-sense messages to drain up the network resources such as network bandwidth and energy to congest the network. For the black-hole and grey-hole attacks, the studies in [16, 17] suggested to use acknowledgement message to confirm the success packet delivery, while for the flooding attack, normally the routing algorithm need to setup a message spreading threshold to detect such attack.

2.1.2.2 Data integrity

Data integrity is another critical security concern in Smart Grid communication network. As Smart Grid communication network has involved significantly the monetary information and management data, thus the unauthorized manipulation on monetary information could cost huge amount of the financial lost. In addition, the manipulation on management information could cause the grid ownership being handed over to the malicious parties such as terrorists. In such case, the malicious parties can use this information to cause the chaos in the human society. A typical data integrity attack is the wormhole attack [16, 17]. The wormhole attack creates a super speed link between two malicious nodes to attract source nodes to route their data through it to the destination node for the shorter time. For example, the normal link speed is at 10Mbps, but this super speed link use optic fibre connection which the bandwidth is up to 1Gbps. In such case, these two malicious nodes might have the chance to manipulate the data without any awareness by the source and destination nodes. The studies [16, 17] suggested that the source node can send estimated arrival time information with the

packet together to the destination nodes, if the actual arrival time is much less than this estimated time, then this route should be considered as malicious and need further investigation.

2.1.2.3 Information privacy

Information privacy is another critical concern in Smart Grid communication network. As customers' information is available on the Smart Grid in service providers' database, some parties like business adversaries may misuse this information for marketing competition. One typical attack is the replay attack [16, 17]. It is to record the identify information or data from its legitimate neighbour and use it later on to pretend as its neighbours so as to gain the authorization for further malicious actions. The studies in [16, 17] suggested to allocate a timestamp and expire time with the legitimate nodes asymmetric key, so as to prevent malicious node reuse the asymmetric key later on.

In this thesis, we are particular interested on the attacks to the network availability such as black-hole and grey-hole attacks, and they will be the main attack pattern we are investigate for our proposed trust-aware routing algorithms.

2.2 Traditional security approaches

In the traditional security approaches, there are three major concerns, i.e., confidentiality, data integrity, network availability. From this, we can see that the focuses on the traditional security approaches are sort of similar to the security concerns in Smart Grid environment. Moreover, the five fundamental security aspects in wireless ad-hoc networks are defined in [18], other than the three concerns just mentioned, there are also authentication and non-repudiation are added.

Confidentiality is also often referred to information privacy where the concern is to ensure the information should only be accessible to the authorized person or parties. The popular approach to confidentiality is providing data encryption on sensitive data with encryption key distributed only to authorized person or parties [2, 18, 19]. The data integrity is to ensure the data should only be modified by the authorized parties in authorized manner, and also the data corruption due to poor network condition is one of the concerns in data integrity [2, 18]. The most common approach for data integrity concern is having a strict access control of end-users and validate end-users' input by

using hash techniques. The network availability is to ensure the network resource always available when it is needed [2], but in the traditional security approach has less focus on it. The authentication is to identify users to ensure their actions are legitimated, and non-repudiation is refer to security methods that to prove the data integrity or signature's genius [18].

From the above major concerns, it can be seen that the primary mechanisms for traditional security approaches are encryption, key management system, authentication, and access control. So the rest of this section gives an overview of the related work on these mechanisms.

2.2.1 Encryption

Cryptography refers to the secret writing which is believed to be the most powerful mechanism to against many kinds of security threats [20]. There are two basic encryption methods, which are substitution and transposition. The encryption can protect information privacy, and also protect data integrity because it translates the data into an unreadable form. In such case, it makes the unauthorized manipulation almost impossible.

The encryption can be hardware encryption, software encryption, or applying both. The hardware encryptions only apply to particular connection link as it requires the hardware on both end of the connection has encryption and decryption function on them. At the same time, the hardware encryption automatically encrypts the connection to ensure the secured data all the time. While the software encryption is more flexible compare to the hardware encryption when selecting the encrypt connection link, but at the same time, it provides the option to disable the encryption which can be misused by malicious parties.

The cryptographic algorithms normally require a piece of information or parameter as input to encrypt the selected data, and this piece of information or parameter is referred as the encryption key [20]. There are two types of encryption keys, which are symmetric key and asymmetric key. The symmetric key is a single and same key that it is used to encrypt and decrypt the data. Due to the vulnerability problem of key distribution for symmetric key, the asymmetric key is developed in later time. The asymmetric key is actually a pair of keys set which includes public key and private key,

and they are mathematical related. The private key must keep in secret and should only be known by the authorized group as this key is use to decrypt the data. The public key is open to public used to encrypt the data only. The typical symmetric key cryptographic algorithms are Data Encryption Standard (DES) [21] and Advance Encryption Standard (AES) [22]. The typical asymmetric key cryptographic algorithm is RSA [23].

The study in [2] has strongly recommended using data encryption mechanisms to protect the privacy and data integrity in Smart Grid communication network. Depending on the different type of connections such as wireless or wired line connections, there are different degrees of security mechanisms to protect the communication network such as Virtual Private Network (VPN), IPSec technologies, Secure Shell (SSH), etc.

For the cryptographic mechanisms, the key distribution obviously is a problem need to be resolved, so as to ensure the cryptographic mechanisms working probably and securely. In the following, we will have an overview on the key management techniques.

2.2.2 Public Key Infrastructure (PKI)

Cryptographic algorithms can encrypt and decrypt data with cryptographic keys, while obtaining the keys by malicious parties can make the encryption data exposed and unauthorized manipulated. So the distribution of encryption key to authorized parties is a critical issue in cryptography. The asymmetric keys are not only used to encrypt and decrypt data, it can also be used as the identification for the entities. In the typical multicast key management scheme, a third party of trust agency such as Key Distribution Centre (KDC) or Certification Authority (CA) is involved to certify the ownership of the public key by its digital certificates [24]. Trust is the key factor in this scheme as the CA need to be trusted by both sides of the parties in order to exchange keys, so as to make this key management structure running properly. The X.509 [25] is telecommunication standards for PKI which is developed by International Telecommunication Union (ITU), and there are many studies on this aspect [26-29]. All these studies are focus on the trustworthiness issue on CAs. For the Smart Grid security solutions, there are already many studies on how to adopt the existing PKI into Smart

Gird such as [30-33]. These studies have shown the possibility of PKI adoption in Smart Grid Environment, and as one of the authentication techniques to verify entities in Smart Grid. But in Smart Grid environment, there will be millions of devices and across hundreds of organizations which make the PKI must be scalable and very complicated [3], which make the current PKI approaches facing the scalability and complexity issues to be addressed and advanced.

2.2.3 Authentication

Authentication is the technique to verify the identities of each components in Smart Grid, so as to allow authorized parties being granted the access to authorized resources and contents, while keep the unauthorized parties away from the classified data. This can help protecting information privacy and data integrity. The useful techniques can be passwords, biometrics such as fingerprint, security card, digital certificates from CA, etc. The study in [34] has defined three factors for authentication, which are something you know (e.g. passwords), something you have (e.g. security card), or something you are (e.g. fingerprint). Also this study suggested that the authentication shall be the first and strongest defence line for the Smart Grid to against any cyber-attacks.

The study in [35] has introduced a set of design principles for authentication protocol for Smart Grid communication network which is valuable guidelines for researchers who work in the area of authentication protocol for Smart Grid. One example of authentication protocol for wireless sensors network is μTESLA [36] which is the modified version of the TESLA [37] protocol to overcome the high energy consumption problem in TESLA. TESLA is using digital signature which is asymmetric key for authentication. The asymmetric key requires very high computing resources and complex processing which is too much energy consumption for the low battery devices, so the μTESLA give up the asymmetric key and use the symmetric key with expire time on it. For the wireless mesh network infrastructure in Smart grid, this is obviously a need for saving the energy and computing resources. There are also other studies on authentication protocols in Smart Grid environment such as [38, 39], and they all have shown their concerns on the significant consumption of computing resources and power in wireless environment.

2.2.4 Access control

Access control is the policies of enforcement to ensure only authorized parties to be granted the access to what they have been allowed. For example, the customers in Smart Grid are only allowed to check their own power usage and bill, but not others. There are three categories for the access control approach, which are Access Control List (ACL), Mandatory Access Control (MAC), and Role Based Access Control (RBAC) in [41]. The policies in ACL are a set of access rights for users on every protected resource [40]. As there are millions of devices in Smart Grid environment which can make the ACL very complex and resource exhausted. The study in [4] suggested that the MAC policy is not flexible enough thus it is not suitable for the Smart Grid environment, while the RBAC is a better choice for Smart Grid comparing with the other two. The principle for assigning the access right to the users is always allocating them with the minimum privilege. Access control is focus on defending data integrity and information privacy attacks. However, even following the minimum privilege rule, if the misconfiguration on the access control occurs, it could lead to the availability issues because the authorized users cannot access to their authorized resources.

2.2.4 Summary

The previous security mechanisms such as cryptography, public key infrastructure (PKI), and authentication are the traditional ways to address these security concerns after reviewing these traditional security mechanisms in above, it can be seen that in most cases, these mechanisms require the PKI to establish a trust relationship between both sides of the communication. These trust mechanisms are mainly based on the digital certificates in PKI, while it can only interpret some limited information about the trust concept. Therefore, in recent years, there is an emerging research direction to discover the trust and reputation mechanisms, as the alternative solutions to the traditional security mechanisms, to tackle the challenging security problems in wireless mesh network for Smart Grid environment. In this thesis, we focuses on the trust-based secure end-to-end routing solutions for wireless mesh network infrastructure in Smart Grid. In the following sections, we will first review some wireless network end-to-end routing protocols, and then present a detailed introduction and discussion on the trust concept and also trust-aware routing algorithms.

2.3 Routing in Wireless Mesh Network

In wireless mesh network, the nodes are forwarding the packets through their neighbours, and this allows the packets travelling through the network scale beyond the source nodes' radio range. In such case, the end-to-end routing plays an important role in wireless mesh network technology and it strongly relies on the available and cooperative neighbours as the trustable relay nodes to guarantee the success delivering the information to the destination node. The study in [14] suggested that the routing protocols for wireless mesh network should have multi performance metrics such as the scalability, robustness, energy-efficient and end-to end QoS such as packet delay and loss etc. There are many existing routing protocols using only hop count metric which have been proven ineffective in some situation such as for the applications with critical QoS requirements. As the wireless mesh network in Smart Grid is normally huge in utilities number, the network set up and maintenance in such situation can take a very long time. There are two type of routing protocols for ad hoc network, which are on-demand and proactive protocols. In this situation, the proactive routing protocols obviously are not suitable because of the long setup and maintenance time.

There are four different types of routing algorithms have been described in [14] for wireless mesh network infrastructure, i.e., routing with various performance metrics, multi-path routing, hierarchical routing, and geographical routing. The first one is the routing protocol by considering various performance metrics. The metrics such as link quality, hop count, etc. are addressed in the routing algorithm. The multi-path routing approach is to find multi-path for a traffic flow which is useful for load balancing and backup purposes. The hierarchical routing is to select some of the nodes as the regional cluster heads (i.e., gateway) while others as the members under these cluster heads. Whenever a member node needs to communicate with other nodes out of its cluster, it needs to go through the cluster head. The last one is geographic routing which it uses the geographical information to locate the destination node, and also find the proper directional neighbour nodes to forward the packets. The advantage of geographical routing is that it is insensitive to the network topology change [44].

In this section, we introduce three routing algorithms and focus on the Greedy Perimeter Stateless Routing (GPSR) [45] algorithm. As the benchmark algorithm of

ATSR is based on this algorithm by adding extra trust-aware metrics, and our proposed algorithms are also inspired from it.

2.3.1 Greedy Perimeter Stateless Routing (GPSR)

The Greedy Perimeter Stateless Routing (GPSR) is an on-demand geographic routing protocol which uses the geographical information on both source and destination nodes to calculate and determine the directional end-to-end path between them. In GPSR algorithm, the nodes only know one hop neighbours, which mean a node only knows the neighbours within its radio range. In such case, the algorithm is a hop-by-hop algorithm which the algorithm performs the selection of its neighbour node as next hop at each node rather than the source node selects the full end-to-end path to the destination node. As mentioned previously, the advantage of geographic routing is that the network topology change has less impact on its performance. While on the other hand, this geographic routing protocol has a common problem of routing loop. To overcome this problem of geographical routing, there are two strategies are introduced in [45]. The first is greedy forwarding with full network graph and the second is the perimeter forwarding with planar graph.

2.3.1.1 Greedy forwarding

The greedy forwarding is the typical geographic routing approach which selects all of the neighbouring nodes within the radio range and calculates their distance to the destination node. The neighbour with closest distance to destination node is selected as next hop to forward the packet.

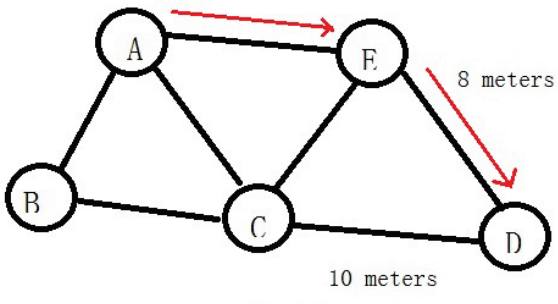

Figure 2.4 – An example of Greedy forwarding

The Figure 2.4 shows an example of greedy forwarding. In Figure 2.4, node A is looking for the path to node D. The nodes B, C, and E are the candidates of next hop for packet forwarding. The GPSR algorithm calculates the distance from destination node to these three nodes, where node E has the closest distance to the destination node. In such case, node A will forward the packet to node E, then node E identify that the destination node D is already within its radio range and then forward the packet to node D directly.

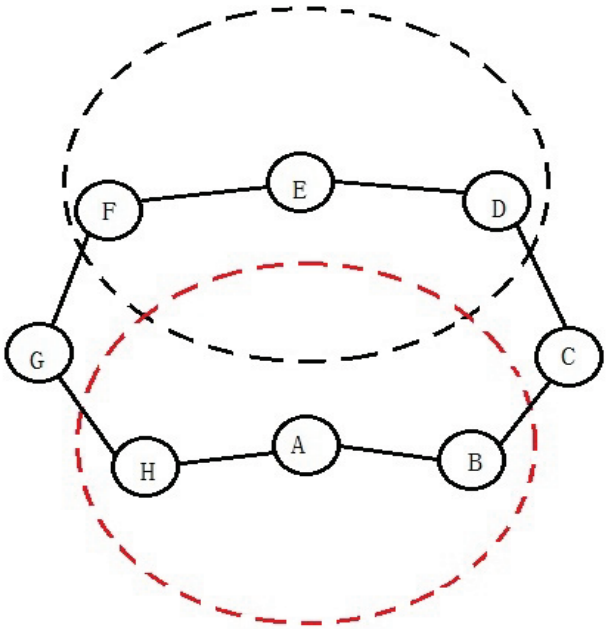

Figure 2.5 – The void problem in greedy forwarding

However, the greedy forwarding is not working effectively in some network scenarios. For example, if there is no neighbour node which has the closer distance than the source node to the destination node. In such case, the algorithm will not know which neighbour to forward. An example is shown above in Figure 2.5. Node A is sending packets to node E. The red dash circle is the radio range of node A, and the black dash circle is the radio range of node E. The black lines are the wireless connection between the nodes. Within node A's radio range, there are two neighbours which are node H and node B. As node A wants to send packet to node E, but node H and node B both have longer path to node E than node A. In such case, the greedy forwarding cannot select node H and node B, which indicates non-reachable to the destination node. But actually there are two possible paths to destination node through nodes H, G, F or nodes B, C, D. Therefore, the second strategy of perimeter forwarding is designed for such situation.

2.3.1.2 Perimeter forwarding

Whenever in a situation that the source node has the closest distance to the destination node which is out of its radio range, the perimeter forwarding algorithm will be conducted. The perimeter forwarding follows the right-hand rule which is inspired from the real life path finding rule when people get lost in the maze. In GPSR, the right-hand rule is described in Figure 2.6 as below.

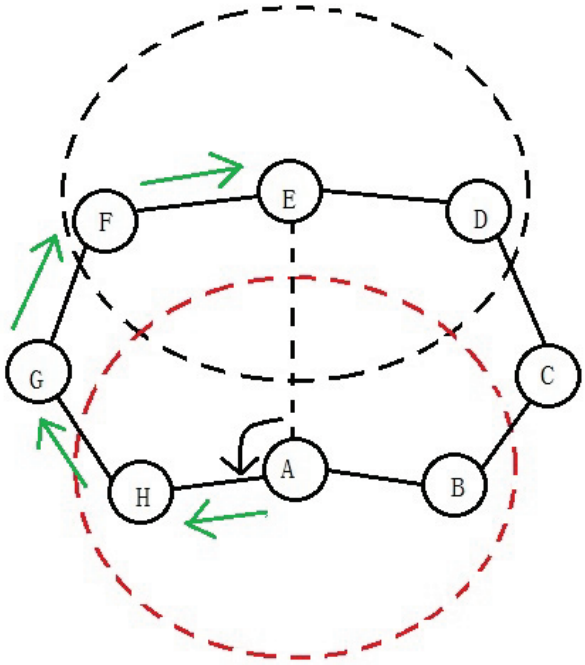

Figure 2.6 – An example of Perimeter forwarding

As shown in Figure 2.6 above, node A wants to send packet to node E. The right-hand rule in GPSR is travelling ahead using counter clock wise from the dash line between node A and node E, which is to select the first neighbouring node appeared in the counter clock wise direction to forward the packets. So the node H is selected between node A and node E, then nodes G, F until reaching node E.

The study in [45] suggested the no-cross heuristic to enforce right-hand rule in GPSR, so as to find perimeter that encloses voids in the region where the edges of the graph cross. The no-cross heuristic in [45] is the most common planar graph such as Relative Neighbourhood Graph (RNG) [46] or Gabriel Graph (GG) [47].

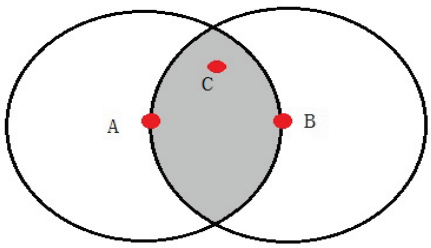

Figure 2.7 - Relative Neighbourhood Graph (RNG)

As shown in Figure 2.7, the two big circles represent the wireless radio range for node A and node B. The grey area is the overlap area by both node A and node B. In RNG, if there is an edge exists between node A and node B, there should not have the third node C in the grey area. Namely, there should not be any other node in it. Otherwise the RNG will exclude this connection between node A and node B in the perimeter forwarding selecting.

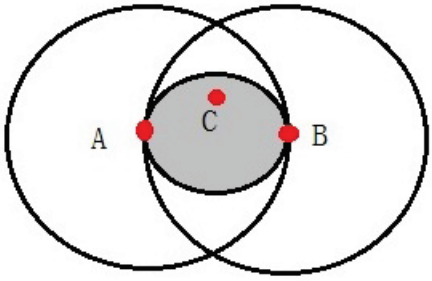

Figure 2.8 - Gabriel Graph (GG)

The Gabriel Graph (GG) is similar to the RNG, node A and node B are the two ends of the diameter in the circle. Within this grey area circle, as shown in Figure 2.8, there should not be node C there, otherwise GG will exclude the edge of node A and node B.

In GPSR, the perimeter forwarding only selects either RNG or GG to exclude the possible cross edges in the network graph, but not both RNG and GG. Moreover, once the algorithm has been switched into perimeter forwarding mode like the situation in Figure 2.6, this mode starts at node A, then the next hop node H will still stay at the

perimeter forwarding mode. This is because the two neighbours of node H are node G and node A, node G's distance to destination is further than node A, where the perimeter forwarding initiates. The perimeter forwarding mode is continue to take place when all possible next hop neighbours' distance are further than the previous node. While the node G will switch back to the greedy forwarding mode, as shown in Figure 2.6, as the node F's distance to destination is closer than node A's distance.

The study in [45] compares the GPSR with Dynamic Source Routing (DSR) [48] algorithm using the simulation studies. It has shown that the GPSR achieve higher packet delivery ratio with lower routing protocol overhead in the same network scenario. The DSR algorithm is introduced in the following session as it is also a popular ad hoc network routing protocol and some existing trust-based routing algorithms are inspired from it.

2.3.2 Dynamic Source Routing (DSR)

The Dynamic Source Routing (DSR) is same type of routing protocol as GPSR which is on-demand routing algorithm. The DSR algorithm is specifically designed for the wireless ad hoc network, which the network node is self-organizing self-configuring, and requiring loose network structure and administration [48].

Using the DSR algorithm, a node uses source routing rather than hop-by-hop routing like GPSR does. Namely the node has completed route information to the destination node, which is stored in its router cache already, rather than finding which neighbour as the next hop node to forward each time. There are two mechanisms used in the DSR algorithm, which are route discovery and route maintenance.

The route discovery mechanism in DSR obviously is used to find the route from source node to the destination node, and this only happens if the route to the destination node is not stored in its route cache. The source node broadcast the route request to all its neighbours, and its neighbours will further broadcast this route request if they do not have a route to destination node stored in their cache until this route request message reach the destination or one of the nodes has the route to destination node stored in cache. Then all the route requests which reach the destination node will report back and update the complete route information in the source node. In case of preventing a huge

amount of route request messages flooding the network, there is a hop count limit on the request message.

The maintenance mechanism is used to detect any network topology change in the network. If a node finds the packet cannot reach the destination node any more through the route information stored in its route cache, the maintenance mechanism removes this route from the route cache and use the backup route. If there is no more backup route left in the route cache, then the discovery mechanism will be activated. The way to determine the route is not working any more is to wait for the confirmation message back from the destination node when the packet is sent out. After a predefined amount of attempts, the route error message will be reported back to source node to inform this route is not valid anymore and be deleted from the cache.

2.3.3 Ad-hoc on Demand Distance Vector (AODV)

The Ad-hoc on Demand Distance Vector (AODV) another popular routing algorithm is designed specifically for ad-hoc wireless network, and it is focus on loop free routing with lower network bandwidth consumption [49]. AODV has two mechanisms which are similar to the DSR, they are discovery and maintenance mechanisms.

The discovery mechanism in AODV is similar to the DSR algorithm, but rather than using source routing, the AODV algorithm is using hop-by-hop routing. Namely, the source node and intermediate nodes only store one hop route in their routing table. For example, the completed route from node A to E is A→B→C→D→E. In node A's routing table, if there is a packet request to node E, it only has the information that B can be the next hop to forward packet to E, and in node B's routing table, it only list the information that C can be the next hop to forward packet to E without a full view on the end-to-end path.

The maintenance mechanism in AODV is an inherence from the Destination-Sequence Distance Vector (DSDV) in [50], but with a new feature that it introduces the expiration time for the routing table, so as to clean up the potential broken and out-of-date link information. Once the AODV finds the broken link, the neighbours of the broken link will send the notification to all the nodes which are affected by this broken link. In this case, it helps the discovery mechanism to eliminate the chance of generating broken link, as the discovery mechanism also accept existing routes information from

intermediate nodes. In DSR algorithm, this can be an issue to make DSR algorithm inefficient.

However, all the AODV and DSDV require the hello message from all the nodes stored in the routing table to maintenance the information of updated network topology, and this could cause the overhead problem in a large scale of ad hoc network. The study in [50] has confirmed that the AODV has generating more routing overhead than the DSR does.

In this session, we give a background introduction on the popular routing algorithms for the ad hoc wireless network. In the next section, a detailed review on the trust concept and some existing trust-based routing algorithms for wireless mesh network are presented and discussed.

2.4 Trust-based Routing

The wireless mesh network is dynamically self-organized and self-configured multi-hop communication network, and these great features make the wireless mesh network easy to extend, flexible, and adaptive to the changes [51]. This is an infrastructure that makes itself different from the traditional tightly structured infrastructure based wireless network. Each node in the wireless mesh network is self-organized and self-configured, namely no central management or administration is needed in wireless mesh network. In such case, some traditional security mechanisms may not suitable for wireless mesh network anymore such as PKI. As the CA is not trustable any more in a peer-to-peer infrastructure Moreover, as wireless mesh network is multi-hop communication network, and it strongly relies the inter-mediate nodes to forward the packets to anywhere in the network. Therefore, any node in the wireless mesh network can be the entry point for malicious attack to gain the access into the network [52]. The study in [53] suggests that there are two sources of threats to routing protocols, one is from external, and another one is from internal. The external threats can be defended by using cryptographic scheme, but for the internal threats the cryptographic schemes can do little and inefficient as internal threats come from the compromised nodes. In addition, in the multi-hop communication network, a source node wants to transmit packet to anywhere in the network has to depend on the reliable and trustable intermediate nodes.

As mentioned previously, there is a lack of CAs in wireless mesh network which disables the most traditional security mechanisms relied on the PKI, such as public key encryption and digital signature. Moreover, in some cases, the nodes in the wireless mesh network behave selfish by not cooperating with others such as refusing to forward the packet, so as to save their own energy and computing resources. This is not a malicious attack which makes the most traditional security mechanisms become inefficient to handle this case. The study in [54] suggests the watchdog mechanism to detect the misbehaviours of the nodes and its effectiveness has been confirmed by the simulation studies. But rather than passively detect misbehaviour nodes every times they conduct, the trust and reputation approach might be a better way to forecast and avoid these misbehaviour nodes. Therefore, it requires an alternative solution to the traditional security mechanism to set up the trust and reputation management system to secure the communications in wireless mesh network.

In this section, firstly, an overview of the definitions of trust in different areas is given, so as to study how researchers describe the trust approach in their areas. Then we will have a review on the existing trust-based routing protocols, so as to identify the research gap and unsolved problems where we can work on and discover for better performance.

2.4.1 Trust concept

Trust is very important term in our everyday life as all relationships rely on the trust in the human society, and each interaction with other people involves trust as well. For example, the customers are shopping in the stores and the staffs recommend the products to them. If customers trust the staff then they will consider their suggestions and might buy the products, but if they don't trust the staff then there is no sale happening. Trust is not only important in human society, but also in the computer security area. Take the public key cryptography as an example, it requires the key only can be accessed by the authorized person otherwise this security mechanism become compromised which involves the trust on both sides of the communications.

While trust is hard to be defined as it can mean different things in different areas. The Oxford dictionary has defined the trust in six words: confidence, belief, strength, goodness, responsibility, and reliability [55]. In details, the trust can give

confidence; trust is a subjective matter which is a strong belief on someone or something in the goodness; trust can give strength; trust can make people relies on someone and something.

There are some other definitions regarding to the more general aspects of trust. The study in [62] defines trust as the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other party will perform a particular action important to the trustor, irrespective of the ability to monitor or control the party. In [63], Josang interprets trust as the belief that it will behave without malicious intent for passionate entities (e.g., human), and as the belief that it will resist malicious manipulation for rational entities (e.g., system). Denning also gives more explanation about trust in [64]. He claims that trust cannot be treated as a property of a trusted system but rather an assessment based on the experience that is shared through networks of people.

These three definitions can be concluded as the expectation on the other parties' performance and without malicious intent, and also this expectation will become the experiences shared through the networks of people. It can be seen that this include two parts: the first part has defined the direct trust experience, and the second part can interpreted as the word of mouth which is about the reputation i.e., indirect trust.

In psychology area, most studies they focus on are how the trust is setup on someone or something. The studies in [56-57] both suggested that the past experiences have great impact on building the trust. The study in [58] points out a property of trust: the trust is not reciprocal. In other word, node A trust node B, but node B is not necessary trust node A back. The study in [59] has defined the trust by giving an example. An individual is in front of an ambiguous path which can either lead to an even of beneficial or harmful and this is depend on another person's behaviour. This individual perceives could be more harmful than beneficial on this path. If this individual decide to take the ambiguous path then this individual is decide to trust, otherwise this individual is decide to distrust. This example implicates trust can affect the individual's decision, and the definition is more related to a belief on someone or something in the goodness.

Trust in sociology is considered as a foundation of relationship between people. The studies in [60-61] have both defined trust as the dynamic probability of someone

will behave as expected. The study in [59], Luhmann suggested the concept of trust is a means of reducing of complexity in society. This is saying that when people are making decision, there are always assumptions for the situation so as to make the trusting decision. These assumptions can be understood as trust.

In the field of ad hoc wireless network routing, the most popular definition for trust is the probability of an individual node will behave as expected, and the studies in [11, 65 - 69] are all use the value from 0 to 1 to represent the level of trust. The study in [73] defines the trust in communication network environment as "a set of relations among entities that participate in a protocol. These relations are based on the evidence generated by the previous interactions of entities within a protocol. In general, if the interactions have been faithful to the protocol, then trust will accumulate between these entities." This definition has highlighted two points. First, they define trust in communication network is being built up with previous interactions, namely experiences. Secondly, the probability is used in the approach to describe and model the trust level. Moreover, these definitions in wireless network on trust are more relating to the behaviours of the entities. The studies in [70, 71] have defined five properties of trust in wireless ad hoc network environment, which are dependent, asymmetric, dynamic, subjective, and not transitive.

For example, the node A can be trusted in packet forwarding performance but might be the malicious at data integrity aspect. The node B can be trusted in data integrity but might be fail to pass the authentication. These are the context dependent property in trust. As mentioned before in [58], the trust is not reciprocal. Node B forwards the packets for A every time, so node A trust in node B. But node B does not necessary need to trust node A back, this is the asymmetric property in trust. The definitions in studies [60, 61] described that the trust as dynamic probability which state that the trust is a dynamic value. For example, node A trusts node B previously, but node B betrays node A in a later stage, which makes node A no longer trusts node B, which is the dynamic property in trust. The study in [72] has defined the trust is a particular level of subjective probability that different person might have different level of trust on the same person or things, and this is the subjective property in trust. The last property of not transitive is, for example, node A trusts in node B, and node B trusts in node C, in such circumstance, it does not indicate that node A should trust in node C. Moreover, the study in [71] has defined the sixth property which trust is a measure of

uncertainty. It can be seen that the trust has been defined most frequently as the probability in [11, 60, 61, 65-69]. When the probability value is not either 0 i.e., not possible at all, or 1 i.e., definitely happen, that makes the uncertainty on a particular behaviour occurs or not.

In the next section, we will review the existing trust-based routing algorithms, and study how they defined and measured the trust in the wireless mesh network environment.

2.4.2 Existing Trust-based routing algorithms

From the definition of trust in above, we have concluded that there are two components in the trust. They are direct trust, and reputation i.e., called as indirect trust. Many existing studies in [11, 60 - 75] agree with these two components and adopt them into the practical modelling and calculation of the trust. The direct trust is the personal direct trust experience to other entities and the reputation i.e., indirect trust is more like a trust recommendation received for other entities from the third parties [75]. In such case, to measure the trust on a particular entity requires the integration of direct trust and indirect trust as one final trust value for decision making. The survey study in [74] has classified the integration approaches into two major mechanisms, such as the discrete model and the fuzzy logic model.

In the following, we will have a detailed discussion on the existing trust-based routing algorithms based on these two integration models, and the ATSR will be introduced in detail as it is the benchmark algorithm used to compare with our proposed trust-aware routing algorithms in this thesis.

2.4.2.1 Trust-based routing algorithms using discrete model

The discrete model approach is to use linear addition function to combine two or more different metrics together as one final value for making routing decision. In most trust-based routing protocols which use discrete model, the linear weight factor technique is popular used to combine the direct trust, indirect trust and other metrics together.

The Ambient Trust Sensor Routing (ATSR) in [11] is one of the typical trust-based energy away geographical routing algorithms by using the liner weight factors. The ATSR uses the direct trust i.e., t_{direct} and indirect trust i.e., $t_{indirect}$ to model and

measure the trust for each node in the wireless mesh network, so as to evaluate and detect any predefined malicious activities in the network and thus avoid these malicious nodes by routing or re-routing the packets to the more trustable end-to-end paths.

There are eight performance metrics considered for the evaluation on the direct trust. The first metric is related to the packet forwarding, which is the successful packet forwarding ratio through its neighbours. The network layer acknowledgement (ACK) metric is the ratio of received confirmation message from the destination, which can be used to detect the black-hole and grey-hole attacks. The data integrity metric is defined as the ratio of packets arrived destination without any manipulated. The authentication metric is to confirm legitimate neighbours. The confidentiality metric is used to check whether the neighbours support encryption or not so as to confirm the connection is secured. The reputation response metric is defined as the ratio of the reply for the reputation request from neighbours. The reputation validation metric is to protect against the bad-mouthing attacks. The last one is the energy metric which is the residue battery level in each node. The bad-mouthing attack is the malicious nodes spreading out the fake reputation messages of particular neighbours, so as to damage these neighbours' trust reputation or hype up the trust reputation so that they can have further malicious attacks [76]. This sort of bad-mouthing attack can be considered as the flooding attacks as well. These eight metrics reflect the context dependent property in trust which has been defined in studies [70, 71]. The ATSR algorithm calculates these 8 metrics by using the probability method. The ATSR is using the total number of satisfied interactions divided by the total interactions, the equation is shown as below.

$$T_i^{A,B} = \frac{S_i^{A,B}}{S_i^{A,B} + F_i^{A,B}} \tag{2.1}$$

In the Equation 2.1, node A is evaluating the trust on node B. The $S_i^{A,B}$ represents the total number of satisfied interactions, and the $F_i^{A,B}$ represents the total number of malicious interactions, and i is one of the eight metrics for the direct trust. The ATSR then use the linear addition function, i.e. weight factors to combine all of these eight metric into a final direct trust value, and this is shown in the following equation.

$$t_{direct} = \sum_{1}^{8} (W_i \times T_i^{A,B}) \tag{2.2}$$

In the Equation 2.2, the W_i is the weight factor for the *i*th metric of the eight metrics counted in direct trust. By using the weight factors in ATSR algorithm, this allow the leverage among the different metrics by weighting up the most critical metrics with higher weight factors while weighting down the less important metrics with lower weight factors. The next step is to synthesize the t_{direct} and $t_{indirect}$ using the linear function again. In ATSR, these factors are called as confidence factors, and the equation is shown as below.

$$t_{final} = C^{A,B} \times t_{direct} + (1 - C^{A,B}) \times t_{indirect} \qquad (2.3)$$

In Equation 2.3, the $C^{A,B}$ represents the confidence factor for the direct trust. The t_{direct} and $t_{indirect}$ can be combined into the final trust value using the confidence factor. In addition, the ATSR is also a geographical-aware routing algorithm, which it uses the geographical information to calculate the direction to destination node. The ATSR algorithm uses the greedy forwarding mechanism in the GPSR [45]. The distance metric in ATSR algorithm interprets the distance from the neighbours to the destination node into a value within the range between 0 to 1, where 0 is furthest and 1 is closest to the destination node. Then the ATSR algorithm combines the final trust value with distance by using the weight factors. The combination of trust and distance metric is using the same way as the combination of direct trust and indirect in Equation 2.3. The final score is thus generated after this combination with a suitable set of weight factors. The ATSR algorithm will selects the neighbour with the highest final score as the next hop for forwarding the packet until reaching at the destination node.

There are two major problems in the current ATSR algorithm. It can be seen in the Equation 2.1, it does not consider the dynamic trust property. In such case, the malicious nodes can accumulate a significant number of satisfied actions to have a high value in $S_i^{A,B}$ over the beginning period as a very trustable node. When the $S_i^{A,B}$ becomes large, the $F_i^{A,B}$ is relative small, this can make the ATSR algorithm ineffective to detect and avoid the most recent malicious attacks. We call this first problem in ATSR as time-insensitive problem.

The second problem we have identified in ATSR is its inflexible weight factor selection problem. The ATSR algorithm need to use the weight factors to combine the eight metrics for direct trust, and also the confidence factor to combine the direct trust

and indirect trust, as well as a set of weight factors to synthesize the trust metric and distance metric into the final score for decision making. There are three sets of these weight factors need to be configured and setup manually every time to a specific network scenario. A static set of weight factors obviously cannot handle all the different network scenarios such as various topologies, traffic demands and attack patterns. For example, in a situation that the source node is surrounded by many malicious nodes in the direction from the source to the destination node. In this condition, it requires the higher priority on the trust metric by using a higher trust weight factor, so as to enforce the ATSR algorithm can detect the malicious attacks and then avoid the malicious nodes instantly but might select a longer but more trustable path. On the other hand, if there are not many malicious nodes between the source and destination node, the priority might give to the distance metric, so as to find a shorter path to the destination node with better packet delay and energy consumption.

There are some other studies on the trust-based routing algorithms. The study in [77] selects the most powerful such as better CPU, and trustable nodes as the trusted sensors. These trusted sensors will collect performance evidences for trust builders. The nodes selected as trust builders will evaluate the collected evidences, and the nodes of reputation manager act as an indirect trust consultant for the nodes which are inquire the trust level on other targeted nodes. While there still a security concern that how to ensure the trusted sensors and trusted builders are not compromised. The study in [78] is using the similar way to measure the trust value as ATSR, but rather than blacklist the particular malicious nodes, it blacklist the area where those particular malicious nodes are located. As they believe that in a certain area where the malicious nodes are located, the adversaries will probably select other nodes to compromise and attack again. In other words, the malicious nodes would most likely geographically locate together. The study in [79] introduced an algorithm called as the Trust-Aware routing protocol (TRAP). This protocol is also use direct trust and reputation management scheme, specially takes the link quality into account in trust evaluation and also considers the dynamic property of trust by using reputation checkpoints, so as to ensure the reputation information is up-to-date. The link quality here refers to the stability and throughput of the communication media. All of these studies are using discrete model i.e., linear weight factor technique to combine various metrics. They all have the common problem on how to select the most suitable weight factors for the best performance as well as

how to handle the reconfiguration manually on these weight factors whenever the network scenario changes.

2.4.2.2 Trust routing protocols using the fuzzy logic approach

The fuzzy logic is a computing method that provides the general concept of linguistic description and mathematical measurement. The study in [80] describes the fuzzy logic as a unique technique with the ability to represent subjective or linguistic knowledge in term of a mathematical model, so as to emulate human decision making processes. The fuzzy logic approach is more suitable to model the trust than the discrete model, as the trust is linguistic concept, which those linguistic variables is supported by fuzzy logic in [74].

The study in [81] has proposed a fuzzy based trust management algorithm for wireless ad hoc networks. The algorithm has two steps to evaluate the trust level on a particular node. It uses four input membership functions set to calculate the final metric of trust evaluation, which are the ratio of packet drop, replay packets generated, false routing messages generated, and the ratio of packets to the wrong destination. Each membership function set consists with three linguistic variables which are less, few, and many. There are three linguistic variables for the output memberships which are low, medium, and high. The algorithm defines the fuzzy rules table through the four input linguistic results to determine the trust level of a node, so as to determine the node is malicious or normal one. Moreover, the study in [81] suggested that not all of the packet drops are caused by the misbehaviours, and they have defined four valid reasons for packet drop which are empty battery, destination unavailable, broken link, and buffer overflow. Every time the nodes fail to pass the packet, a log file will record the last ten failure reasons. The second step of the algorithm will do the same fuzzy logic step again like step one, but this time the input membership function sets are the four valid reasons. It defines another fuzzy rules table and determines the major cause of the high packet drop. If the failure reason falls into one of the four valid reasons, the algorithm will set the ratio of packet drop from many to less. While there is one major problem of this trust algorithm that it does not consider the dynamic property of trust in it. Moreover, this algorithm does not have reputation system for the indirect trust, and the fuzzy algorithm does not have the defuzzification, which it only has the fuzzy rules table to roughly determine the linguistic output. The study in [84] has proposed a similar fuzzy trust evaluation approach as in [81]. While it uses the centre of gravity defuzzification

method to combine the metrics into a précis value. The study in [82] also proposed a fuzzy based trust routing protocol based on the AODV, named as FAODV. The trust evaluation is using the similar approach as fuzzy algorithm step one in [81]. In addition, the FAODV set up a trust threshold value to determine whether the neighbour is malicious or not, then the algorithm filter out the malicious nodes and let AODV algorithm to find a route to the destination node among the remaining legitimate nodes. There is a problem that this algorithm only uses trust to detect malicious behaviours during the discovery stage. Namely, once the route to the destination node is selected, and start transmitting the packet, the most updated malicious behaviour most likely to be ignored, so it does not consider the dynamic property of the trust. The study in [83] is based on the studies [82, 85], which has concluded that the direct trust evaluation should include a timestamp on the evaluation on each behaviour of targeted nodes to make sure the evaluation is up-to-date. For the direct trust evaluation, it also involves the energy consumption, computing resource, and also memory buffer as the factors by using the weight factor to combine all of these metrics. As this is a fuzzy based trust routing algorithm in [83], so they remove the weight factors and has these four factors as input for the fuzzy logic to be combined into one final output for the trust evaluation at a particular time.

Except the trust metric, we are also interest on the energy consumption behaviour in our routing algorithms, therefore a review of energy aware routing algorithms are presented in the following session to set up the background knowledge for energy efficacy.

2.4.3 Energy aware routing

In wireless mesh network for Smart Grid environment, there are some smart utilities are relying on the battery. In such case, an energy load balancing mechanism is important in the routing algorithm to ensure the energy efficiency of the overall network. Moreover, the awareness of the energy level left in the neighbouring utilities is useful for predicting the reliability of neighbouring node as if the residue battery is lower than the threshold means this node can die in anytime. This is especially critical for wireless mesh network infrastructure as the communication strongly relies on the neighbouring and intermediate nodes.

There are some existing studies on the trust-based routing protocol which is also involving the energy consumption as one of the metric. The ATSR algorithm put energy consumption as one of the factors into the direct trust evaluation. It uses the weight factors to combine energy metric with the rest of other metrics into a final output value as the direct trust value. The simulation studies in [11] have confirmed that it can achieve lower energy consumption when the energy weight factor is at 0.8. This is because that the energy metric can enforce the algorithm achieve energy load balancing by selecting multi-path to the destination node. But at the same time when the energy weight factor is configured as 0.8, this means that the rest of other security metrics are degraded which cause the algorithm less sensitive to detect the malicious activities, and the simulation results in [11] has addressed this problem. In other words, if the ATSR algorithm wants to achieve the energy load balancing, it has to sacrifice the sensitivity on detecting and avoiding malicious behaviours. This is also a typical problem on using discrete model to combine the different metrics in trust based routing algorithm. The study in [83, 84] use fuzzy logic to combine the energy metric with other metrics, but these two studies do not have a simulation study to verify the energy metric performance in their proposed protocol. The study in [81] also includes the energy metric into their fuzzy based routing algorithm. However, rather than using the energy factor to achieve energy load balancing, it uses this energy factor to confirm whether the malicious behaviours are caused by the flat battery or not, so as to judge the nodes are malicious or legitimate upon high packet loss condition.

2.5 Summary

In this chapter, we first give a detailed introduction on the seven domains of Smart Grid and their relationship to each other. These interactions require a secured and reliable communication network. There are three major security concerns in Smart Grid communication network, which are network availability, data integrity, and information privacy. The wireless mesh network infrastructure is one of the promising infrastructure solutions for Smart Grid communication network, where the wireless mesh network can provides reliable connection, low cost scalability, and flexibility. While the wireless mesh network infrastructure has some security concerns due to its nature and should be addressed. There are some traditional security mechanisms such as cryptography, PKI,

authentication, etc. to address the above security concerns, and we are discovering an alternative solution to use trust and reputation approach to tackle this problem.

In addition, we conduct a review on the routing algorithm in wireless ad hoc network and also the existing trust-based routing algorithms. The trust-based end to end routing algorithm is the alternative solution, to the traditional security mechanisms, to provide trust service in wireless mesh network for Smart Grid environment.

Finally we take a look on the energy aware algorithms as some of the nodes in wireless mesh network strongly rely on the battery to conduct operations. In next chapter, firstly we will propose two trust-aware routing algorithms, which are TIGER and DTEGR to tackle the identified time-insensitive and inflexible weight factor sections problems ATSR algorithm. Moreover, we propose the FEATGR algorithm which it uses fuzzy logic mechanism to combine all metrics into a final output value for the evaluation, instead of the discrete approach to solve the trade-off problem among weight factors.

Chapter 3

Trust-based Routing Algorithm Modelling

In previous chapter, we reviewed the major security concerns with wireless mesh network under Smart Grid environment. In this chapter, we have proposed three trust based energy aware geographical routing algorithms to tackle network availability problem in wireless mesh network for Smart Grid environment. Inspired from the existing trust-based routing algorithm ATSR, the first two new routing algorithms are proposed to overcome the shortcomings of the current ATSR. They are Trust-based Intelligent Geo Elective Routing (TIGER) and Dynamic Trust Elective Geo Routing (DTEGR). In addition, to tackle the weight factor selection problem identified in the discrete model trust-based routing algorithm in the previous chapter, we have proposed a Fuzzy-based Energy Aware Trust Geo Routing (FEATGR) algorithm to advance the TIGER and DTEGR algorithms. The FEATGR algorithm is fuzzy-based algorithm which is more flexible and scalable comparing with the previous two algorithms. Moreover, the fuzzy logic approach can emulate the process of our human's dynamic and intelligent decision making, so as to enable this algorithm more intelligent to detect malicious activities occurred in the network and then make an optimal decision to select a secured as well as QoS guaranteed path to the destination.

The remainder of this chapter is organized as follows. First, the TIGER algorithm is presented to resolve the time-insensitive issue on the historical records in ATSR algorithm, and then we have proposed DTEGR to resolve the trust weight factor selection issue in ATSR algorithm. Finally, FEATGR algorithm is proposed to overcome the weight factors selection problem in the previous two discrete models.

3.1 Trust-based Intelligent Geo Elective Routing (TIGER)

We have identified that one of the shortcomings of the current ATSR algorithm is the time-insensitive issue on the historical records explained in the following paragraph. Inspired from the ATSR algorithm by considering the Smart Grid communication network infrastructure, we proposed Trust-based Intelligent Geo Elective Routing (TIGER) routing protocol to tackle this time-insensitive shortcoming in ATSR. It enables the trust metric to precisely evaluate the transverse nodes behaviours timely by using timestamp mechanism.

In ATSR, the equation of the trust metric evaluation is defined as Equation 2.1 described in the previous chapter 2.

$$T_i^{A,B} = \frac{S_i^{A,B}}{S_i^{A,B} + F_i^{A,B}}$$

It is a general average function over the whole span of communication historical records, where $S_i^{A,B}$ is the total number of satisfied interactions between node A and B, and $F_i^{A,B}$ is the total number of malicious interactions. For a long-term period evaluation, the $S_i^{A,B}$ and $F_i^{A,B}$ will be accumulated and are growing into a very large value. In such case, a small amount of the recent malicious behaviours will be hard to be counted and thus has impact on the overall rating of trust. In other word, the sensitivity of the malicious behaviours detection is decreasing over the time. For example, in the human society, a fraud can make 30 successful trades so as to build up the trust relationship among people, and then the fraud takes the advantage of this historical trust relationship to defraud people significantly in the last couple of trades. Therefore, people need to always be sensitive and stay alerted to avoid defraud timely. It is the same to the trust algorithm design. It needs to maintain the sensitivity level over the time. To tackle this existing problem in ASTR, a timestamp technique is introduced here to assign a timestamp on each evaluation record for activity. The format of evaluation record is described as:

{Timestamp ID, evaluation record}, for example, {t1, success}, {t2, fail}, {t3, success}…etc.

By using the timestamp, we can identify the time sequence of the records, which is useful to differentiate that which records are out-of-date, and which records are up-to-date. We are targeting to calculate the trust metric by valuing more on the up-to-date records while fade away the out-of-date records. Here we are proposing two different styles of timestamp to advance ATSR: attenuation window and constant window.

The first approach of Attenuation Window (AW) is based on an exponential equation as following:

$$f(x) = e^{-(\frac{x}{c})}, c > 0 \,\&\, x \geq 0 \qquad (3.1)$$

Where the *e* is Euler's constant that it is equal to 2.71828, the *x* is the timestamp of the particular evidence, and *c* is the coefficient to adjust the speed of decreasing in the result of *f(x)*. For the Equation 3.1, while *x* is growing bigger and bigger, the value of *f(x)* will become smaller and smaller, and finally it is degraded unlimitedly to 0. Moreover, the bigger value of coefficient *c*, the slower in speed of decreasing slopes of the value in *f(x)* between 0 and 1. The curve of this equation 3.1 is shown as figure 3.1 below.

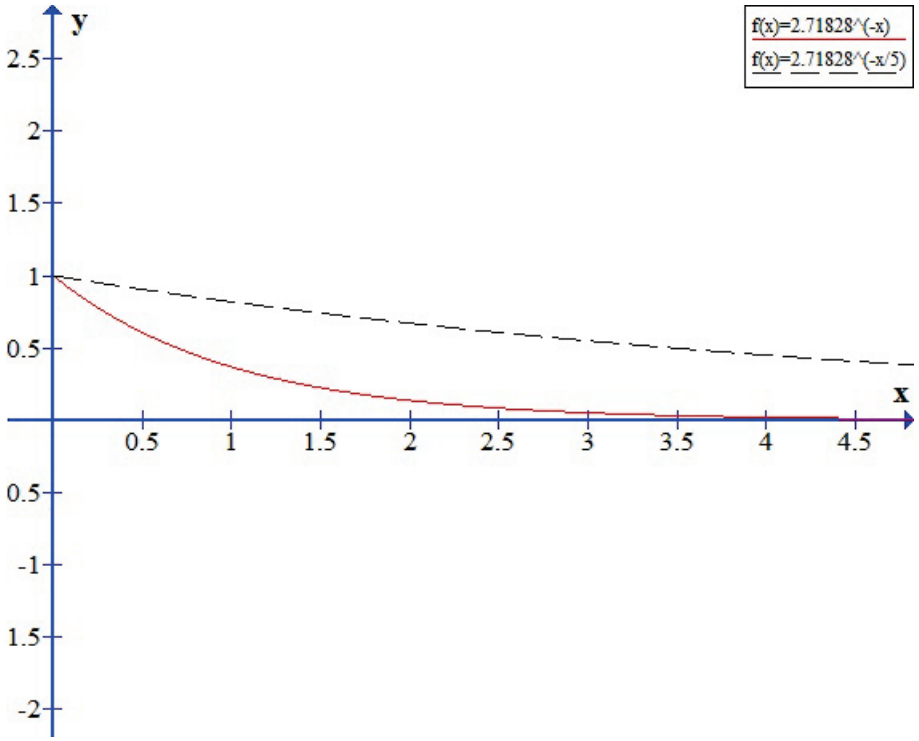

Figure 3.1 - Graph of equation 3.1 (c=0 vs. c=5)

There are two curves in figure 3.1, where the red curve is when *c* = 1, and the black dash curve is when *c* = 5. As can be seen that while *c* is equal to 5 the value of *f(x)*

is dropping much slower than while c is equal to 1. In such way, AW timestamp can emphasize the most up-to-date records and fade away the out-of-date records by the speed controlled by the coefficient *c*.

To apply the AW approach into the direct trust evaluation, here we advance the Equation 2.1 as following Equation 3.2.

$$S^{A,B} = \sum_{k}^{n} e^{-(\frac{n+m-x}{c})}$$

$$F^{A,B} = \sum_{j}^{m} e^{-(\frac{n+m-x}{c})} \qquad (3.2)$$

It can be seen that in the above Equation 3.2, node A is conducting the trust evaluation on node B, and here A is called as the trustor and B is called as the trustee. The $S^{A,B}$ is denoted as the final score of satisfied interaction counted on one of the eight metrics which is using the attenuation windows calculation by node A on B, and $F^{A,B}$ is the final score of dissatisfied interaction counted on one of the eight metrics which is using the attenuation windows calculation by node A on Node B. The *n* and *m* are the total number of satisfied interactions and dissatisfied interactions counted respectively. The *k* and *j* are the most out-of-date timestamp ID for satisfied interaction and dissatisfied interaction on one of the eight metrics. For example, as node A is conducting a direct trust evaluation on node B. There are 3 satisfied interaction records regarding to the packet forwarding metric but 2 dissatisfied interaction records because node B is dropping packets, i.e. n=3 and m=2. Here the satisfied interactions timestamp ID set are x= {1, 3, 5} and dissatisfied interactions timestamp ID set are x= {2, 4}. Thus the k=1 and j=2. As the timestamp ID *x* is from ascending order that it reflects from oldest to latest in time sequence. In such case, *(n+m-x)* of Equation 3.2 will enable the timestamp ID going descending order, so as to fit the *x=(n+m-x)* in Equation 3.1.

AW approach uses the Equation 3.1 to fade away the out-of-date records slowly so as to prevent the accumulation of the out-of-date records. This can make the TIGER algorithm time-sensitive to target's behaviour performance so as to resolve the time-insensitive problem of ATSR's.

The second proposed timestamp mechanism is constant window (CW) which it is using a limited window size to collect the necessary number of evaluation records.

For example, CW=5 which means the most recent 5 records are needed to be collected and considered for the evaluation as shown in figure 3.2 below.

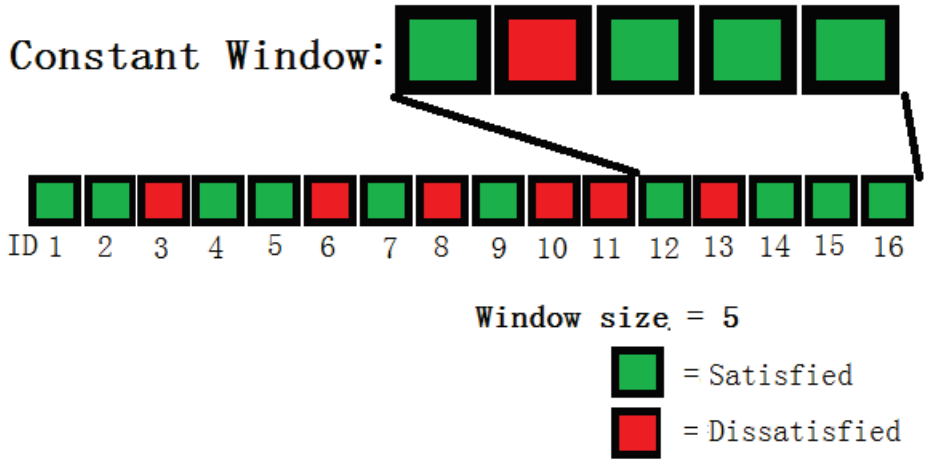

Figure 3.2 - Constant Windows with size 5

As shown in Figure 3.2 above, there are total 16 previous records collected on particular metric with timestamp ID on. The red square is denoted as the dissatisfied evidence and the green square is denoted as satisfied evidence. The constant window is set as to only count the most recent 5 evaluation records which are 12 to 16, and drop the historical records with timestamp ID less than 12. In such case, the TIGER algorithm is able to maintain the performance level of malicious behaviours detection in different periods of time.

Additionally, in the current ATSR algorithm, the trust evaluation is include 8 metrics to measure the performance of target nodes, and then sum up these 8 metrics together with linear weight factors. As so many metrics here are taking into account, the sensitivity of different malicious behaviours will be distracted by the weight factors and also cause extra calculation complexity and overhead. In the TIGER, rather than summarizing all of 8 metrics with different weight factors, we synthesize all the metrics except energy metric into one misbehaviour metric. Every time if any malicious behaviour is detected, TIGER judges it as misbehaviour rather than specifying the malicious behaviour into different metrics and then sum up. In such case, the extra processing of the priority selection will be excluded, such as encryption path has higher priority. The trust concept tries to cover all the misbehaviours that we use the misbehaviour metric to cover all these misbehaviours. Moreover, different kinds of

malicious cyber-attacks are treated as misbehaviours and contribute to the trust value calculation. Therefore, the direct trust evaluation is defined as follow.

$$DT^{A,B} = w_{energy} \times E^{B} + (1 - w_{energy}) \times T^{A,B} \qquad (3.3)$$

In the Equation 3.3, the w_{energy} is the weight factor for energy metric, E^{B} is the residue energy percentage of node B. The $T^{A,B}$ is node B's misbehaviour metric result evaluated by node A. It can be seen in Equation 3.3, it synthesize the energy metric and misbehaviour metric into the direct trust value $DT^{A,B}$ by using the configurable weight factors according to the priority.

Furthermore, the TIGER will calculate direct trust and indirect trust metrics to the final trust score. The equation is shown as below:

$$FT_{final}^{A,B} = w_{direct} \times DT^{A,B} + (1 - w_{direct}) \times IT^{A,B} \quad (3.4)$$

Where w_{direct} is the weight factor of direct trust, and $DT^{A,B}$ is the direct trust value by A on B. The $IT^{A,B}$ is the indirect trust value by A on B and the $FT^{A,B}$ is the final trust value which A is scoring B.

Additionally, the distance metric is added as same as ATSR in TIGER to look for the direction to the destination, as shown below.

$$D^{B} = \sqrt{\left| v^{destination} - v^{B} \right|^{2} + \left| y^{destination} - y^{B} \right|^{2}}$$
$$DM^{B} = 1 - \frac{D^{B}}{\sum_{1}^{p} D^{Z}} \qquad (3.5)$$

As seen in Equation 3.5 above, the D^{B} is the distance from node B to the destination. ($v^{destination}$, $y^{destination}$) is the longitude and latitude of destination, and (v^{B}, y^{B}) is the longitude and latitude of node B. DM^{B} is the value of distance metric of node B, and the value range is between 0 and 1. The bigger value of the distance metric, the closer distance it is to the destination. The *p* represents the total number of neighbours within the node A's radio range, and D^{Z} is the particular neighbour's distance to destination.

Finally the value of trust metric and distance metric value are synthesized by summarizing them with weight factors just like the equations 3.3 and 3.4. The equation is as shown as below:

$$Finalscore = w_{trust} \times FT^{A,B} + (1 - w_{trust}) \times DM^{B} \qquad (3.6)$$

Where $FT^{A,B}$ is the final trust value by A on B, and DM^{B} is the distance value to help TIGER look for the direction to the destination. The TIGER algorithm selects the neighbour with highest final score as next hop to forward the packet until reaching the destination node.

3.2 Dynamic Trust Elective Geo Routing (DTEGR)

In previous section, we have proposed TIGER algorithm to tackle time-insensitive issue on historical records. In this session, we propose the Dynamic Trust Elective Geo Routing (DTRGR) protocol to tackle another shortcoming of the current ATSR algorithm. The ATSR algorithm uses static weight factors between distance metric and trust metric to calculate the final score for each neighbour node. If the weight factor of trust is too high, the distance metric will barely affect the final score, so as to send the packet forwarded to the right destination. On the other hand, if the distance weight factor is too high, the algorithm will hardly detect malicious nodes. Therefore, there is a weight factor selection problem between the detection on malicious node and distance metric. It is obvious that the static weight factor technique cannot tackle all the dynamic network scenarios.

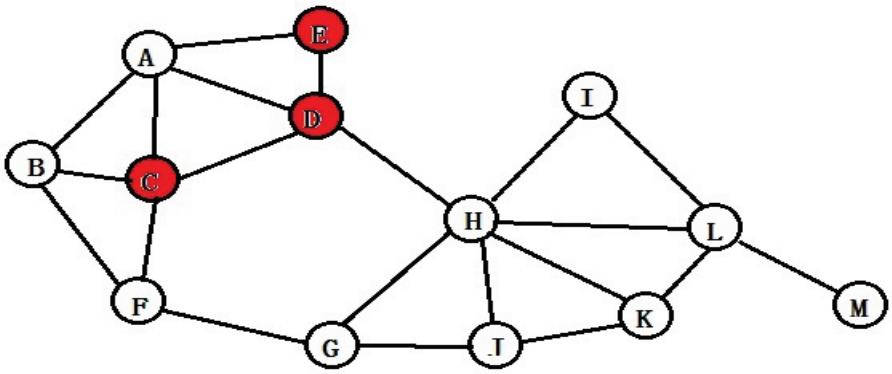

Figure 3.3 - Wireless mesh network with malicious nodes 1

For example, the node A wants to send a packet to node M, and the nodes C, D, and E are all malicious nodes. In this situation, the trustable route is from A to M is A→B→F→G→H→L→M. The ATSR algorithm requires higher weight factor on trust metric, so as to have the trust metric obtain higher priority to enforce the algorithm to ignore the distance metric. In such way, the ATSR algorithm can selects node B as next hop rather than nodes C, D, or E which they have closer distance to destination. This can be seen in the figure 3.3. After the packet is transmitted as A→B→F→G, now node G wants to forward the packet to node M where there is no malicious node in between at all. In node G's radio range, there are neighbours F, H, and J. Node H's distance to destination node M is slightly closer than node J's. Node J's final trust value is slightly higher than node H's in node G, but they are both legitimate node without being compromised. In such case, ATSR algorithm will requires higher weight factor on distance metric to make the algorithm ignore the slightly different on the trust values of node J and node H, so as to have node G select node H as next hop to forward the packet. Though node G select node J as next hop will not cause any packet loss, it certainly selects a long route to cause longer packet latency. In this example, under the same network scenario, it requires two different sets of weight factors to have the algorithm malicious attack and select a shorter path to destination.

To tackle this sort of problem, we propose the DTEGR protocol to split the evaluation process into 2 steps. In the first step, the DTEGR will setup a threshold *h* on the trust value to determine whether this neighbour is malicious or not. If a neighbour node's trust value is higher than *h*, this neighbour will be considered as a trustable node and it will be added in the trustable neighbour forwarding list. In the 2nd step, it selects the neighbour with closest distance to the destination as next hop from the trustable nodes in the forwarding list. An example of DTEGR is illustrated in the figure 3.4 below and the Equation 3.7 is used to generate the safe forwarding list. The initial trust threshold value is the average final trust value of the legitimate nodes in the network which will be presented in later.

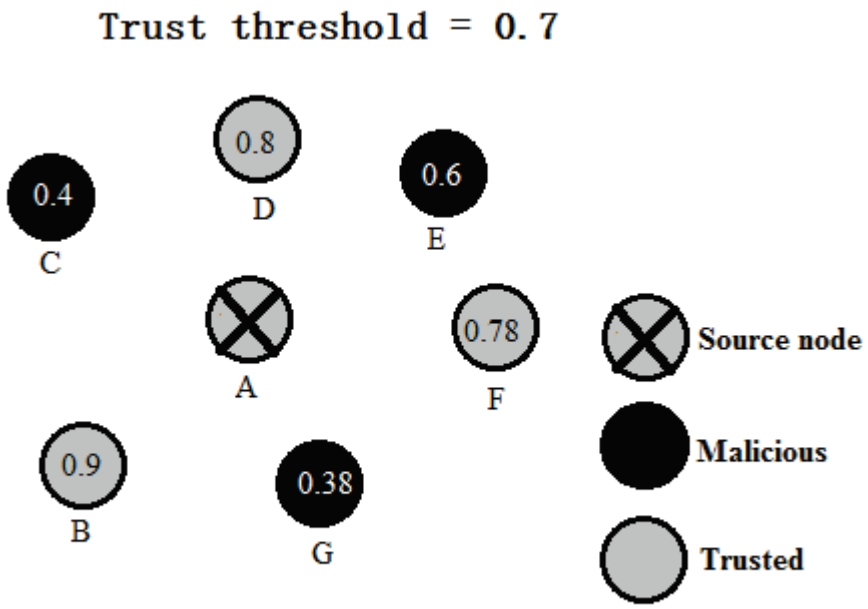

Figure 3.4 - Safe forwarding list

$$FT^{A,B} \geq h \Rightarrow B \in l^{A} \tag{3.7}$$

In Equation 3.7, node B is a neighbour of node A, $FT^{A,B}$ is the trust value of node A on node B , h is the trust threshold value, and l^{A} is the safe forwarding list generated by node A. As shown in figure 3.4, in node A's safe forwarding list, node B, D, and F are selected as their trust values in node A are above 0.7.

By using these two steps strategy, the weight factor selection problem between trust and distance is no longer existed. While now how to configure the threshold value is an issue. If the threshold is too high, it might cause none of the neighbours are selected in the forwarding list. On the other hand, if it is too low, it might include the malicious nodes in the list as well. To further tackle this problem, we enrich a dynamic threshold approach in DTEGR. It ensures at least one neighbour in the forwarding list to guarantee the basic network availability. The maximum value of threshold needs to be determined by the mean of all good nodes' trust value. The equation is as follow:

$$h_{\max} = \left(\frac{1}{\eta}\sum^{\eta} t^{Z}\right) - 0.1 \tag{3.8}$$

This h_{max} is the default trust threshold value for all the nodes in the network, η is the number of selected neighbour nodes with good behaviour, and t^Z is the trust value of one of the well behaviour nodes. In such case, with minus 0.1 of the average trust value this threshold can make sure all the neighbour nodes with good behaviour in the safe forwarding list. While there is a possibility that some of the good nodes might conduct bad performance accidently, and thus safe forwarding list size could be decreased over time. In such case, DTEGR will guarantee sufficient choices on nodes in the list and also granting the second opportunity to the nodes which have poor performance previously. For example, when the threshold is equal to h_{max} and the safe forwarding list size is less than 30% of number of neighbours, threshold value will be deducted by 0.1. This is used provide the second opportunity to those nodes have bad records before. If those nodes perform well again, their trust values should increase back to standard level which is above the default trust threshold value. While if after the first deduction on threshold value, the safe forwarding list is still less than 30% then nothing will happen until the list is empty. When the list is empty, DTEGR will drop the threshold again by 0.1 until the list is not empty. The diagram 3.5 illustrates the logic flow of the DTEGR algorithm.

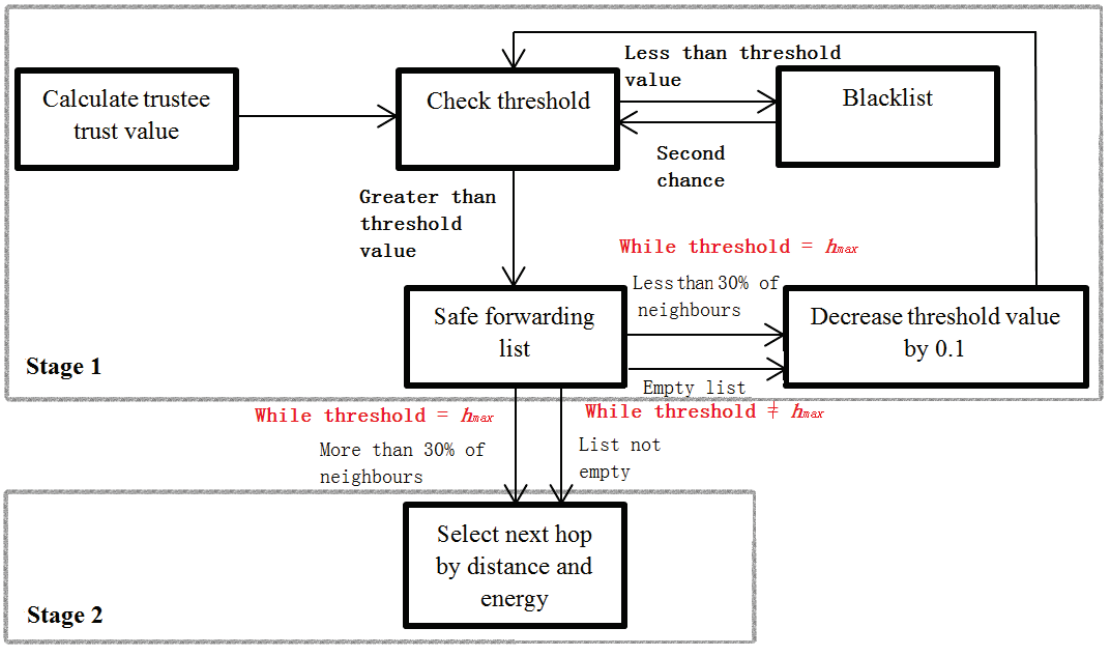

Figure 3.5 - DTEGR algorithm flow

In the first step, it is to calculate the trust metric which consists with direct trust metric and indirect trust metric. The direct trust metric equation is listed as below.

$$T_i^{A,B} = \frac{S_i^{A,B}}{S_i^{A,B} + F_i^{A,B}}$$

Rather than including 8 metrics into the direct trust evaluation like ATSR algorithm, the DTEGR algorithm synthesize all the metrics except energy metric into one direct trust metric. In this direct trust metric, any malicious attack on trustee node will be treated as misbehaviour record. In addition, the indirect trust is calculated from other trustor node which has the direct trust value on the trustee node. The final trust value can be calculated using the Equation 3.3 which was presented in the previous TIGER section. It uses the weight factor technique to integrate the direct trust and indirect trust together.

In DTEGR algorithm, it also synthesizes the energy metric with distance metric together with a weight factor set. Rather than having the energy metric integrated with trust metric like ATSR and TIGER algorithms, we synthesize the energy with distance metric. As ATSR and TIGER algorithms are using the linear weight factor to summarize the trust and distance metric together, the change on the residue energy level will be very sensitive to the final score of trustee. In such case, these two algorithms will keep switching to achieve energy load balance. While in DTEGR case, the energy metric embeds with trust metric, the change on residue energy level will affect the algorithm very slowly as it has to be low enough to make the final trust value below the trust threshold. Once a node has been put into the blacklist by the trustor, it would be very difficult to have this node back to the safe forwarding list again which causes this algorithm very insensitive to the energy metric change. Moreover, energy metric embedded with trust metric in DTEGR is more likely will empty the safe forwarding list which is not good. In such case, with energy metric embeds with distance metric, DTEGR algorithm does not need to sacrifice trust metric sensitivity level of malicious behaviour detection.

The distance metric will be calculated as Equation 3.5 in the previous TIGER section. Then DTEGR algorithm integrates the distance metric and energy metric with a suitable weight factor, and the equation is in below.

$$Finalscore = w_{energy} \times E^B + (1 - w_{energy}) \times D^B \qquad (3.9)$$

After the stage 1 of DTEGR algorithm completed which the safe forwarding list is generated, DTEGR algorithm then proceed the 2nd stage that it selects the neighbours

only from the safe forwarding list which with highest final score and this will keep happen until the packet reach the destination.

3.3 Fuzzy-based Energy Aware Trust Geo Routing (FEATGR)

In the previous section, we have proposed the TIGER algorithm to resolve the time-insensitive problem on historical records of ATSR algorithm by using timestamp mechanisms. Then we proposed the 2-step DTEGR algorithm to tackle the weight factor selection problem between trust metric and distance metric. While in DTEGR algorithm, distance and energy metric are still using the weight factor to be integrated in 2^{nd} step, which means the weight factor selection problem still is existed.

To tackle the weight factor problem, we have proposed a Fuzzy-based Energy Aware Trust Geo Routing (FEATGR) algorithm which is an energy aware trust based geographical routing scheme using fuzzy-logic approach. It is original inspired from previous ATSR and DTEGR routing protocol. As ATSR is using the discrete model for the trust evaluation, thus there is a question on how to determine the weight factor between direct trust and indirect trust? In addition, the ATSR also uses weight factors to combine the trust and distance metrics into one final score, there is a problem on how to determine the weight factors between trust and distance for the best performance. Obviously the fixed weight factors cannot satisfy all the network scenarios. Actually the trust threshold value introduced in DTEGR algorithm can be considered as a linguistic variable in the fuzzy logic mechanism. If the trust value higher than trust threshold, then the linguistic variable value is a trusted node, otherwise the linguistic variable value is a malicious node. Therefore, it might be a better way to setup thresholds for every metric in DTEGR algorithm, and then use the fuzzy logic approach to synthesize all these metric together, rather than using the linear addition function. The fuzzy logic is an effective approach to emulate human intelligent decision making process, and it can synthesize and optimize all the impact factors into a final output score.

In the DTEGR algorithm, there are four metrics involved to evaluate the trustee, which are direct trust, indirect trust, distance, and energy. In FEATGR algorithm, we synthesize these four metrics by using fuzzy logic approach. Namely, there are four

input variables have been defined. Moreover, the direct trust and indirect trust variables have the same structure, as the indirect trust is calculated based on the neighbour's direct trust on the same trustee.

For the direct trust evaluation, we calculate the ratio of satisfied interactions in the most recent 30 times experienced in the node, using the constant window (CW) timestamp technique in TIGER (TIGER-CW). In addition, if the total number of interactions with target node is less than ten, the direct trust is treated as full mark. We call this is warm up period for the algorithm. Once the interactions with target node are more than ten, the direct trust evaluation uses the satisfied interaction ratio as the score. This is to make sure that the enough experiences are collected before determining the target node is good or malicious node. The equation to calculate direct trust is below:

$$b_{good} + b_{malicious} < 10 \Rightarrow t_{direct} = 1$$
$$10 <= b_{good} + b_{malicious} <= 30 \Rightarrow t_{direct} = \frac{b_{good}}{b_{good} + b_{malicious}} \qquad (3.10)$$

In Equation 3.10, b_{good} is the total number of satisfied interactions, and $b_{malicious}$ is the total number of dissatisfied interactions. Indirect trust is the neighbour's direct trust value regarding to its evaluation on the same target node. In addition, the distance metric is calculated as below:

$$d_{distance} = 1 - \frac{(d_i - (d_j - d_r))}{2d_r} \qquad (3.11)$$

When node A is evaluating node B, we call node A as trustor, and node B as trustee. In equation 15, as figure 3.6 shown in below, d_i is the distance from trustee to destination, d_j is the distance from trustor to destination, and d_r is the radius of radio range of the nodes in the network.

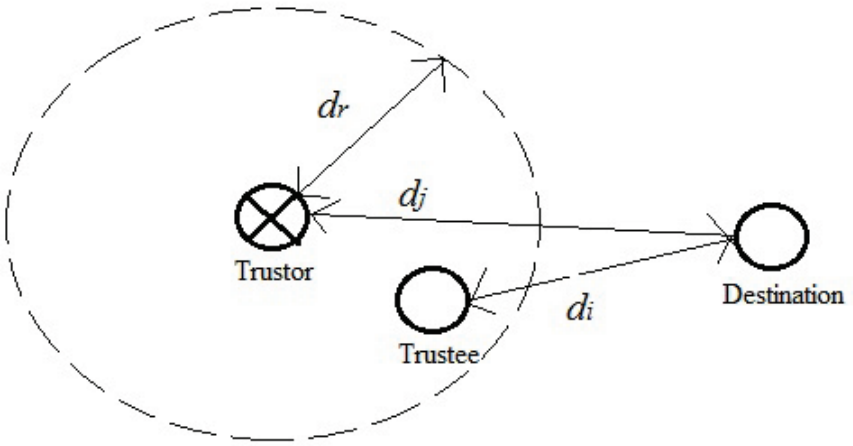

Figure 3.6 - FEATGR distance metric calculation

The fuzzy membership functions for direct trust, indirect trust, energy level, and distance metrics are defined in tables 3.1 – 3.3 and figures 3.7 – 3.9 as below.

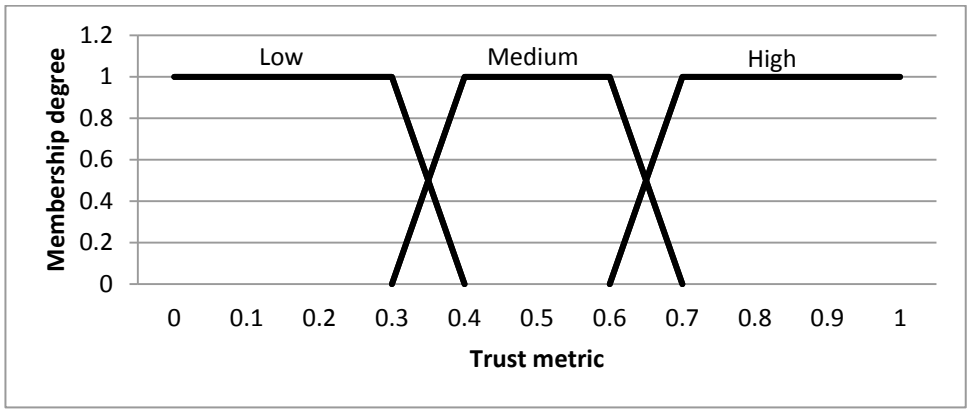

Figure 3.7 - Direct & Indirect trust membership functions

Table 3.1 - Direct and indirect trust linguistic variables

	Lower Bound	*Upper Bound*
Low	0	0.4
Medium	0.3	0.7
High	0.6	1

As shown in figure 3.7 above, the y axes are the degrees of the linguistic variables and x axes are the trust values for direct trust and indirect metric. There are 3 ranges of trust value defined for both direct trust and indirect trust metrics. This is actually three trust threshold values have been defined for both trust metrics like DTEGR algorithm did, but DTEGR only has one dynamic threshold. Rather than

judging the trustee is trustable node or a malicious node, it setups three linguistic variables which are High, Medium, and Low. The details are listed in table 3.1. When the trust value is between 0 and 0.4, the trustee is considered as low in trust level for both indirect trust and direct trust. When the trust value is between 0.3 and 0.7, the trustee is considered as medium in trust level. Finally, when trust value is between 0.6 and 0.7, trustee is considered as High trust level. It can be seen that, when trust value is between 0.3 and 0.4, the trustee is considered as either low or medium in trust, where this produces two possible outputs while the trust value is in these two ranges in each trust metric.

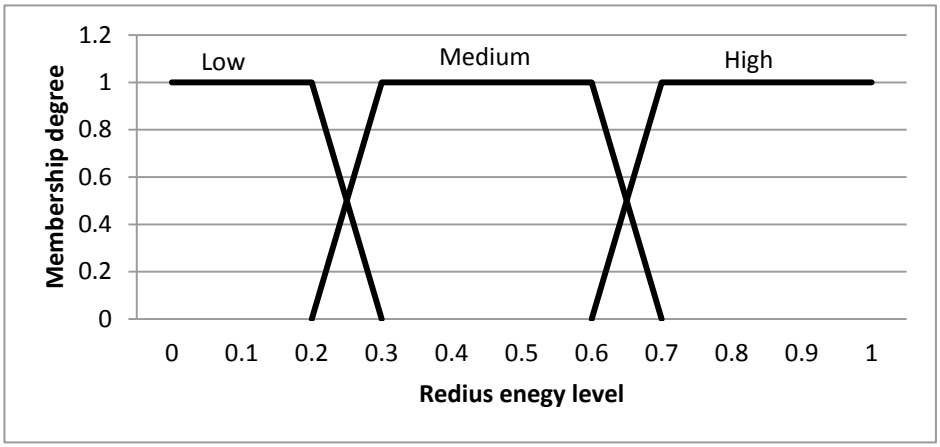

Figure 3.8 - Energy level membership functions

Table 3.2 - Energy level linguistic variables

	Lower Bound	***Upper Bound***
Low	0	0.3
Medium	0.2	0.7
High	0.6	1

In figure 3.8 above, the y axes are the degrees of the linguistic variables which are low, medium, and high, and x axes are the residue energy percentage for energy metric. In the energy level membership functions, we set the low linguistic variable is between 0 and 0.3 which is slightly smaller range compare to trust metric. This is because once the trustee's residue energy level is defined as low, it will be soon judged by the algorithm as an unreliable node and try to avoid route through it. Here we assume that, even 20% of the residue energy level should be enough for the nodes to perform well on any kind of task.

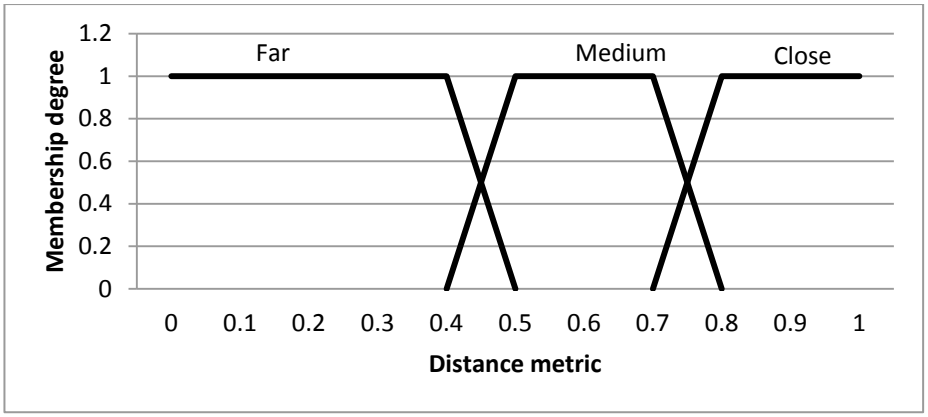

Figure 3.9 - Distance membership functions

Table 3.3 - Distance linguistic variables

	Lower Bound	***Upper Bound***
Close	0.7	01
Medium	0.4	0.8
Far	0	0.5

In figure 3.9, the y axes are the degrees of the linguistic variables which are close, medium, and far, and x axes are the values between 0 and 1 for the distance metric where 0 is with furthest distance and 1 is with closest distance to the destination. While the trustee with distance metric value less than 0.5, that means this trustee have further than trustor to the destination. In such case, we set between distance metric value 0 and 0.5 is far.

After linguistic variables and ranges are defined, we synthesize these 4 membership functions results together as one final value, the final equation as follows.

$$Min(\mu_t(t_{direct}) \cap \mu_r(t_{indirect}) \cap \mu_e(E) \cap \mu_d(d_{distance})) \quad (3.12)$$

In the equation 3.12, $\mu_t(t_{direct})$ is the membership function for direct trust metric, $\mu_r(t_{indirect})$ is for indirect trust metric, $\mu_e(E)$ is for the residue energy level metric, and $\mu_d(d_{distance})$ is for distance metric. The final score is decided by the smallest degree value of linguistic variable among these 4 membership functions results.

The output linguistic variables and membership functions are defined as the table 3.4 and Figure 3.10 in below.

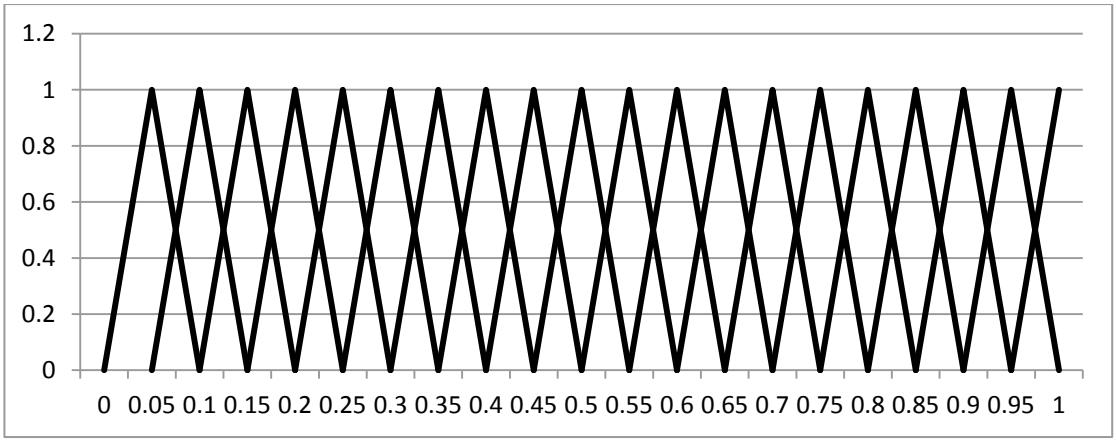

Figure 3.10 - Final output trust membership functions

Table 3.4 - Final output linguistic variables

	Lower Bound	***Upper Bound***
1	0	0.1
2	0.05	0.15
3	0.1	0.2
4	0.15	0.25
5	0.2	0.3
6	0.25	0.35
7	0.3	0.4
8	0.35	0.45
9	0.4	0.5
10	0.45	0.55
11	0.5	0.6
12	0.55	0.65
13	0.6	0.7
14	0.65	0.75
15	0.7	0.8
16	0.75	0.85
17	0.8	0.9
18	0.85	0.95
19	0.9	1
20	0.95	1

There are 4 linguistic variables input in FEATGR algorithm and each linguistic variable has three possible values, so there are total eighty-one possible combinations for the output linguistic variable for each node. For giving expected reactions for each of these eighty-one possible combinations, we classified 20 output linguistic variables as shown in figure 3.10 and table 3.4 below. This output list can provide more precise and accurate result for the FEATGR algorithm. To determine the final output membership, we defined the fuzzy rules table as shown in table 3.5 in below.

Table 3.5 - FEATGR fuzzy rules table

Direct trust	*Indirect trust*	*Energy*	*Distance*	*Output*
H	H	H	C	20
H	H	H	M	18
H	H	H	F	16
H	H	M	C	19
H	H	M	M	17
H	H	M	F	15
H	H	L	C	17
H	H	L	M	15
H	H	L	F	13
H	M	H	C	12
H	M	H	M	10
H	M	H	F	9
H	M	M	C	11
H	M	M	M	9
H	M	M	F	7
H	M	L	C	8
H	M	L	M	6
H	M	L	F	4
H	L	H	C	3
H	L	H	M	2
H	L	H	F	1
H	L	M	C	3
H	L	M	M	2
H	L	M	F	1
H	L	L	C	3
H	L	L	M	2
H	L	L	F	1
M	H	H	C	12
M	H	H	M	10
M	H	H	F	8
M	H	M	C	11
M	H	M	M	9
M	H	M	F	7
M	H	L	C	8
M	H	L	M	6
M	H	L	F	4
M	M	H	C	11
M	M	H	M	9
M	M	H	F	7
M	M	M	C	10
M	M	M	M	8
M	M	M	F	6
M	M	L	C	8
M	M	L	M	6

M	M	L	F	4
M	L	H	C	3
M	L	H	M	2
M	L	H	F	1
M	L	M	C	3
M	L	M	M	2
M	L	M	F	1
M	L	L	C	3
M	L	L	M	2
M	L	L	F	1
L	H	H	C	3
L	H	H	M	2
L	H	H	F	1
L	H	M	C	3
L	H	M	M	2
L	H	M	F	1
L	H	L	C	3
L	H	L	M	2
L	H	L	F	1
L	M	H	C	3
L	M	H	M	2
L	M	H	F	1
L	M	M	C	3
L	M	M	M	2
L	M	M	F	1
L	M	L	C	3
L	M	L	M	2
L	M	L	F	1
L	L	H	C	3
L	L	H	M	2
L	L	H	F	1
L	L	M	C	3
L	L	M	M	2
L	L	M	F	1
L	L	L	C	3
L	L	L	M	2
L	L	L	F	1

As shown in the above table with 81 possible outputs, there are a few fundamental priority rules to refine the output for the fuzzy rules table. First of all, when direct trust metric and indirect trust metric is low, the output variable value should only between 1 and 3. For the distance metric, when it is close, the output is set as 3, and medium for 2, and far for 1. The rest of the case will follow the rules as the highest value is 20, direct trust value at medium will minus 8 from it, this is same as indirect trust as well. This is because trust has most priority in the algorithm so as to avoid the

malicious attack that the rest of metrics combination output results in the table require 8 different output values to differentiate. Then energy metric degrades will loss 1 while degrade from high to medium as the energy metric has least priority in the algorithm. But when the energy metric value is low, the output value will be set to lowest in the same direct and indirect trust level. Distance metric will loss 2 for the degradation as it has higher priority than energy metric so as to find the direction to the destination.

For the four metric inputs, there are up to two possible membership degree values for each membership function. In FEATGR there are four membership functions with total 16 possible membership degrees and linguistic values combination for the final output. Each linguistic value has 81 possible outputs by the fuzzy rules table which described in the table 3.5. Here we use the centroid gravity defuzzification method [87] to combine these 16 values into one final output value. The equation for centroid defuzzification is defined as below:

$$Finalscore = \frac{\sum \mu_o(x) \times x}{\sum \mu_o(x)} \tag{3.13}$$

In Equation 3.13, $\mu_o(x)$ is the membership functions of the output, o is the linguistic variable, x is one of the 16 possible output values. The FEATGR algorithm selects the neighbour node with highest final output value as next hop to forward the packet to the destination node.

There is a possibility that more than one neighbour node have the same highest final score at the same time. In such case, the FEATGR select the first neighbour node coming into the algorithm which obtain the highest final score as next hop to forward the packet rather than the one which has closer distance to the destination. Therefore, in the circumstance of more than one neighbour node have the same highest final score, we setup one more step that the algorithm will select from these neighbours with same highest final score by closest distance to the destination as next hop. To achieve, the Equation 3.11 is used again here to determine the shortest distance.

3.4 Summary

In this chapter, we have proposed three trust-based geographical routing algorithms. The TIGER algorithm was inspired from the logic of ATSR algorithm and introduces

the timestamp mechanisms to resolve time-insensitive issue on historical records in ATSR. The DTEGR algorithm advances the ATSR algorithm into two steps, which it uses the trust factor with a dynamic threshold value to generate a safe forwarding list first, then it uses distance factor as routing metric to decide the next hop from the safe forwarding list. In such way, the weight factor selection problem between distance and trust in ATSR algorithm can be resolved. Finally, the FEATGR was proposed to advance the previous two algorithms i.e., TIGER and DTEGR algorithms. FEATGR algorithm use fuzzy logic mechanism to emulate the process of human decision making, so as to make the algorithm more intelligent, flexible, and scalable comparing with TIGER and DTEGR algorithms.

In the next chapter, we will conduct extensive simulation studies to validate and evaluate these three proposed algorithms. The goal is to confirm that the effectiveness and performance of timestamp mechanism in TIGER algorithm, DTEGR able to resolve the weight factor selection problem between trust and distance metric in ATSR, and FEATGR has better flexible adaptive performance than TIGER algorithm and DTEGR algorithm according to dynamic network scenarios.

Chapter 4

Simulation studies

In this chapter, we have conducted extensive simulation studies on the three proposed trust-based geographical routing algorithms by using J-Sim tool [86]. J-Sim is standing for JavaSim, which is a Java component-based and compositional network simulator. The proposed TIGER, DTEGR and also FEATGR algorithms will be validated and studied through different network scenarios for their efficiency, stability and adaptability performance on detecting and avoiding malicious nodes attacks in the network. In addition, the energy efficiency of three proposed algorithms has been investigated. First of all, the TIGER algorithm's performance will be given. And next we look at the second algorithm DTEGR on the adaptability through different network scenarios with the benchmark algorithm, and testify the stability through different malicious attack levels scenarios. Then through different network scenarios, we confirm energy load balance by the energy metric in DTEGR algorithm and the sacrifice on the performance in DTEGR algorithm. Finally, we compare the FEATGR algorithm with TIGER and DTEGR algorithm in different network scenarios for the efficiency of detecting malicious nodes and the stability on performance. Then through the energy side of simulation studies to confirm the better scalability and flexibility of the FEATGR algorithm compare to TIGER and DTEGR algorithms.

The ATSR algorithm has been used as the benchmark algorithm to compare with our proposed algorithms to verify their performance on detecting and avoiding malicious nodes. In addition, we also use GPSR algorithm as the benchmark algorithm to show the extra energy consumption for enabling security features on our proposed routing algorithms. The ATSR algorithm is embedded security features by using trust based routing on top of the GPSR algorithm. This is same to TIGER, DTEGR, and FEATGR algorithms as well. In such case, GPSR algorithm becomes the perfect benchmark to find out the energy consumption for the security features in our proposed algorithms.

In the next section, we will introduce the three performance metrics evaluated for our three proposed routing algorithms in the simulation studies.

4.1 Performance Metrics

In the simulation studies, we have selected three performance metrics to be evaluated regarding to the malicious node's attacks. They are packet loss ratio, mean packet latency, and energy consumption.

The packet loss ratio is the percentage of the packet loss over total packets transmitted in the transmission. This metric is effective to measure the sensitivity performance on detecting and avoiding the malicious behaviour. For example, how fast the algorithm can detect then avoid the malicious behaviours by analysing how many packets are dropped by black/grey-hole attack and thus not successfully delivered to the destination nodes. The more of the packet loss indicates worse performance in malicious node detection.

The mean packet latency is the average packet delivery time from the source to destination nodes. This metric is effective to measure the route finding capabilities of the algorithms, which the less mean packet latency indicates a shorter path is found.

Finally the energy consumption obviously is used to measure the energy efficiency feature of the algorithms. The algorithm aware the residue energy level change in the nodes that it base on residue energy level to select multi-route to the destination so as to achieve energy load balance. This energy load balance feature can help nodes with longer battery life.

In the next section, we will have a detail introduction of the structure organized and purposed of each simulation scenario in this chapter.

4.2 Simulation studies structure

The structure of simulation studies in this chapter is shown in Figure 4.1 as following.

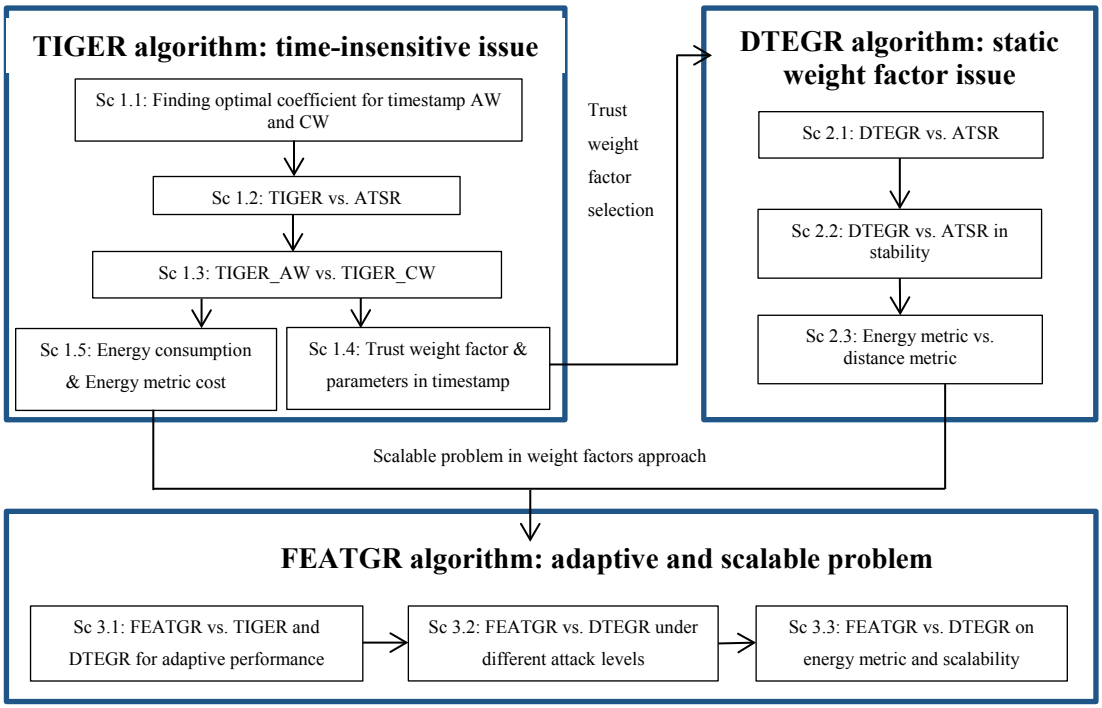

Figure 4.1 - Chapter 4 flow chart

As can be seen in the Figure 4.1, we will first have TIGER algorithm case study with ATSR algorithm as benchmark. There are five scenarios for the TIGER algorithm case study which are listed in below:

- Scenario 1.1: Finding optimal coefficient for timestamp AW and CW
  - ❖ Timestamp AW has coefficient *c* and CW has windows size need to be selected, so as to achieve the best packet loss and latency results in the network scenario.
- Scenario 1.2: TIGER versus ATSR
  - ❖ TIGER algorithm with optimal coefficient setup is able to resolve the time-insensitive issue of ATSR.
- Scenario 1.3: TIGER_AW versus TIGER_CW
  - ❖ The comparison between AW and CW timestamp in the TIGER algorithm through different random network scenarios indicate AW algorithm perform better under focus attack such as black-hole attack, and CW perform better under random attack such as grey-hole attack.
- Scenario 1.4: Trust weight factor & parameters in timestamp

- ❖ Trust weight factor in TIGER algorithm affect the selection of optimal coefficient of timestamp. If trust weight factor increase, the optimal coefficient will be increase accordingly as well.

- Scenario 1.5: Energy consumption & Energy metric cost
  - ❖ The security feature on TIGER algorithm has the same energy consumption as ATSR algorithm. In addition, the energy metric in TIGER has sacrificed the trust metric performance to achieve energy load balance.

From the scenario 1.4, we find out the problem of trust weight factor selection in TIGER algorithm. In such case, we propose the DTEGR algorithm to come over this problem. There are three scenarios in the DTEGR case study and they are listing in below:

- Scenario 2.1: DTEGR versus ATSR
  - ❖ In the same network scenario, the packets were sending from different source nodes to different destinations, DTEGR able to maintain the some performance level in packet loss and latency without selecting any weight factor where ATSR required different trust weight factors to be able to detect and avoid the malicious nodes efficiently. Namely, DTEGR avoid the trust weight factor selection problem in ATSR.

- Scenario 2.2: DTEGR versus ATSR in stability
  - ❖ With the extensive simulation studies under different attack levels, we found out DTEGR algorithm is performing better under heavy attack level, while ATSR performs better under light attack level with optimal trust weight factor selected.

- Scenario 2.3: Energy metric versus Distance metric
  - ❖ In DTEGR algorithm, energy metric able to achieve energy load balance without sacrificed the performance of trust metric. But instead, it sacrificed the performance of distance metric. When the energy metric is high, it can make DTEGR cannot find the direction to the destination.

In the scenario 2.3 for DTEGR algorithm case study, it reveals the scalability problem on weight factor approach to combine different metrics in the algorithm. Though DTEGR algorithm avoid the trust weight factor selection problem in TIGER algorithm, it still have weight factor between energy and distance, also the direct trust and indirect trust. TIGER algorithm has the same problem as well, such as the weight factor between energy and direct trust. To overcome the scalability problem on weight factor approach, we proposed FEATGR algorithm which use fuzzy logic mechanism to substitute weight factors. There are three scenarios for FEATGR algorithm case study as following:

- Scenario 3.1: FEATGR vs. TIGER and DTEGR for adaptive performance
    - ❖ Under the same network scenario, the packets were sending from different source nodes to different destinations, FEATGR algorithm able to maintain the performance level in packet loss and latency like DTEGR did without adjustment on any parameters like TIGER algorithm required.
- Scenario 3.2: FEATGR versus DTEGR under different attack levels
    - ❖ FEATGR algorithm able to achieve better performance in packet loss in overall while under light and medium attack level compare to DTEGR algorithm. While under heavy attack level, FEATGR able to achieve similar performance in packet loss with DTEGR.
- Scenario 3.3: FEATGR versus DTEGR on energy metric and scalability
    - ❖ Both FEATGR and DTEGR are able to achieve energy load balance in the simulation so as to extend the battery life time for the nodes in the network. But for DTEGR to achieve the better result in energy load balance, it sacrificed the distance metric performance in DTEGR so as to make the algorithm lost direction in the network while the nodes start drain up their batteries. While FEATGR do not have such problem at all. This show the scalability of the FEATGR algorithm, where TIGER and DTEGR require additional weight factor value to have new metric include in the algorithm. FEATGR only require the adjustment on fuzzy rules table to include new metric in the algorithm.

4.3 Case study 1: TIGER vs. ATSR

In this section, we have conducted the simulation study on the new proposed TIGER algorithm in Smart Grid wireless mesh network infrastructure. The main goal is to validate that our proposed trust-based routing algorithm TIGER has resolved the time-insensitive issue on historical records in current ATSR algorithm. To verify this, we will first study the timestamp mechanisms in TIGER algorithm which are attenuation window (AW) and constant window (CW) compared with ATSR algorithm. Both AW and CW in TIGER algorithm have a parameter which is coefficient *c* in AW and window size in CW need to be setup for the mechanisms. Firstly, we have studied the sensitivity of these parameters on the network performance for these two timestamp mechanisms. By using the optimal parameter value for AW and CW in TIGER algorithm, we have compared with ATSR algorithm for the time-sensitive of detecting and avoiding the malicious nodes over time. Next we compare the AW and CW timestamp mechanisms in TIGER algorithm in stability performance regarding to 10 random malicious nodes deployment scenarios. In addition, the TIGER algorithm is using weight factor to combine the trust and distance metric together, and a further study on how these weight factors' change has impact on the performance of detecting malicious nodes is performed. Finally we test the stability of the TIGER algorithm with AW and CW regarding to 10 random malicious nodes deployment scenarios.

4.3.1 Simulation scenario setup

We assume a wireless mesh network topology with 100 wireless smart meters been allocated in a 10x10 grid as shown in the Figure 4.2 below. There are 26 malicious nodes were deployed in the network, so as to deploy heavy malicious attacks to block the routes between source node 1 to destination node 99. These attacks are represented in the nodes which are highlighted in red colour. The node 12 conducts black-hole attacks (B_H) which is used to evaluate the sensitivity level of malicious behaviours detection. While the rest of nodes conduct grey-hole attacks (G_H) during the simulation period. In each simulation run, there are 900 traffic sessions to be proceeded, and the interval is 4 second. Each session also forwards 1 UDP packet with 31 bytes data, and the time to live (TTL) is 128 milliseconds.

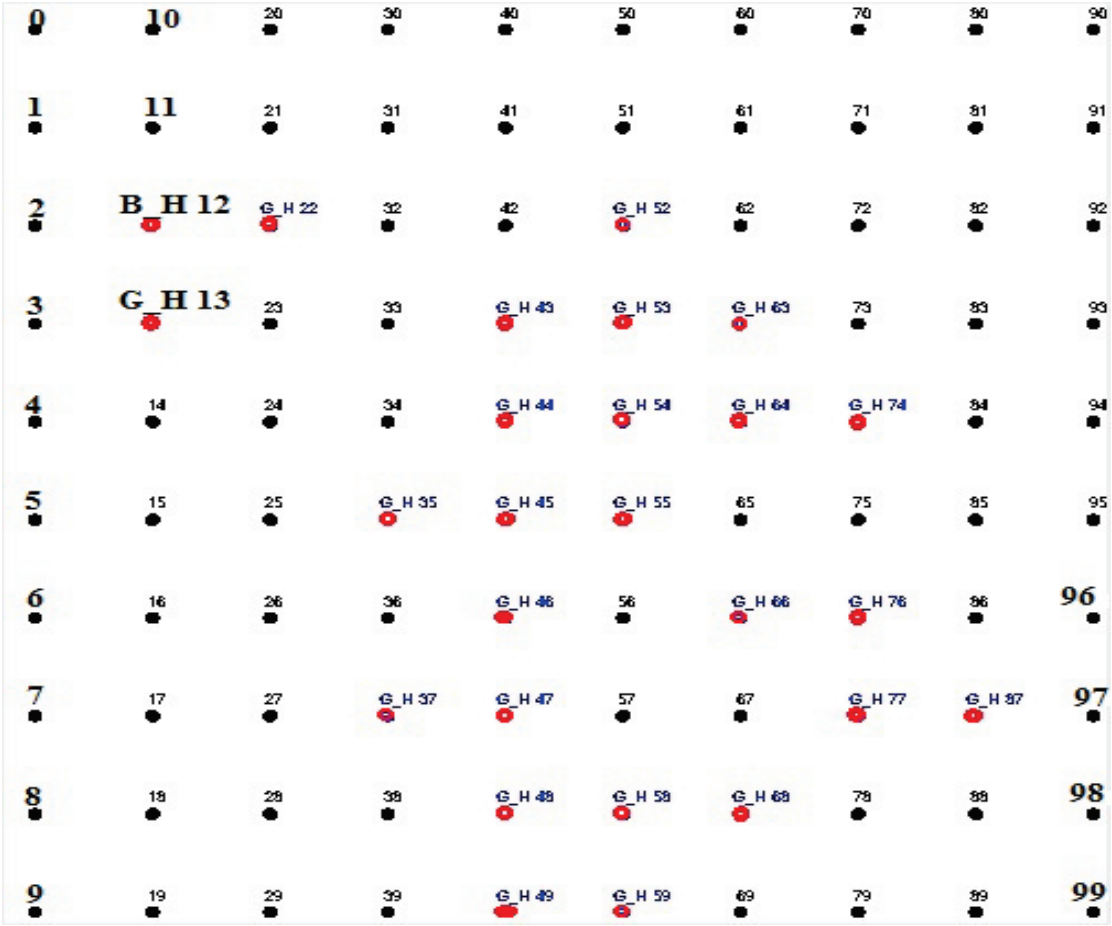

Figure 4.2 - The 10x10 network topology.

Grey-hole attacks are occurred after 30 sessions proceeded by the malicious nodes, as well as after 60 sessions by the malicious nodes, the grey-hole attacks are terminated. Then grey-hole attacks start again after 100 sessions by the malicious nodes, and stop again after 130 sessions are preceded. Finally, the grey-hole attacks start again after 200 sessions. The details are shown in the Figure 4.3 in below. The grey-hole attacks will randomly drop 50% of the packet received by malicious nodes. For example, a grey-hole attack starts after 30 sessions were proceeded and stop after 60 sessions, this will approximately cause 25% of packet loss which make hard for the algorithm without timestamp mechanism to detect. Same way for the attacks starts after 100 sessions and stops 130 sessions, then start again after 200 sessions. In such case, malicious nodes start the attacks at different time one by one rather than all start from beginning. We use these scenarios to compare the detection sensitivity between the TIGER and ATSR algorithm.

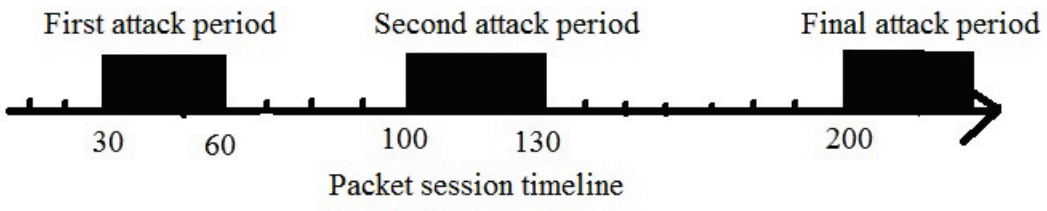

Figure 4.3 - Grey-hole attack pattern

The weight factors for trust metric and distance metric are setting as 0.5 and 0.5 respectively, and there is no weight factor on energy metric for both algorithms in this section as we focus on studying the performance on trust and distance metrics first. Moreover, in ATSR algorithm, only packet forwarding metric which is detecting packet forwarded ratio by the neighbours will be turn on as only black/grey-hole attacks are performed in the simulations. For TIGER algorithm, there are two different approaches to setup the timestamp: the first approach is TIGER_AW (i.e., attenuation window) and the second one is TIGER_CW (i.e., constant window). Both algorithms have a coefficient to be configured so as to adjust the performance of the timestamp. Therefore, these two coefficients should be determined before conducting the simulation.

4.3.2 Finding optimal Coefficients

In the first simulation scenario, the node 12 starts conducting the black-hole attacks after 200 packets have been forwarded by the malicious nodes. We use the TIGER_AW algorithm with coefficient *c*, which is introduced in the Equation 3.1 with value set {5, 15, 20, 25, 30, 35, 40} to investigate which value has the lowest number of packet loss. The result is shown in the Figure 4.4 and Table 4.1 below.

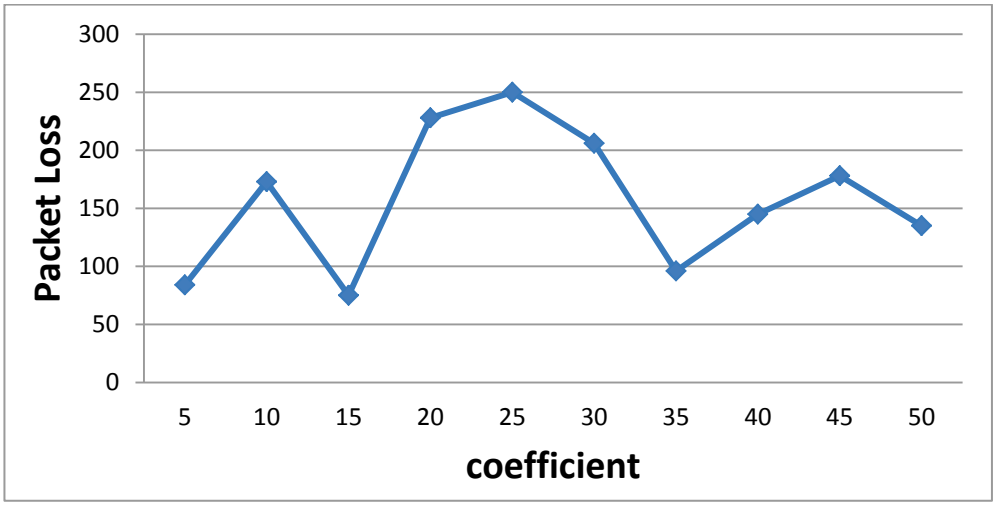

Figure 4.4 - TIGER_AW coefficient impact on packet loss.

Table 4.1 - Packet sacrificed to detect node 12 black-hole attack in AW

TIGER_AW	*Coefficient c*									
	5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
Packets sacrificed	2	3	4	5	6	7	8	9	10	11
Mean Latency (ms)	17.51	15.55	13.17	14.52	14.75	15.42	13.11	13.12	14.06	14.75

According to the results in Figure 4.4, we can see that the values of 5, 15, and 35 are most effective coefficient of c, and the value of 15 is the most suitable for this topology as it has least packets loss. In the table 4.1, the packets sacrificed number is indicating the number of packets sacrificed to detect and avoid node 12's black-hole attacks. It also shows the sensitivity of detecting malicious behaviours change according to different coefficient values. The sensitivity is decreasing as the c is increasing, because the packets sacrificed number to detect a malicious node was increasing accordingly.

The increase in *c* the slope of the curve is decreased in Equation 3.1.

$$f(x) = e^{-(\frac{x}{c})}, c > 0 \,\&\, x \geq 0 \tag{3.1}$$

As a consequence, the sensitivity on the time for the evidences is decreasing, which cause the detection sensitivity on malicious behaviours decrease. From the least packet loss result achievement, the value of *c* equals to 5 is best. This explains why the packets loss is increased after *c* is configured as 35. While considering the packet loss and also latency as a whole, *c* is configured as 15 is best for this scenario. It can be seen that, the coefficient value of *c* with 5 has best detection sensitivity and lowest packet loss, but it has highest mean packet latency comparing with the others. This is due to the low value of *c* that it makes the trust metric over sensitive, so as to misjudge some of the good nodes as malicious nodes. In such case, the TIGER algorithm will select a further route to the destination so as to avoid these misjudged nodes. In addition, the misjudgement of good behaviours nodes will cause TIGER algorithm reselect route even the secured route is found. The more route reselection happens that the more chance to encounter malicious nodes while finding the alternative route. This also explained why the packet loss is so high when the coefficient is 10, 20, 25, and 30. The decreased on sensitivity level is another factor for the high packet loss number. The TIGER algorithm is conducting more historical records to make a judgement whether the nodes are malicious while *c* is increasing. In such case, TIGER will most likely

achieve more packet loss results at the end, such as when *c* is 35 the algorithm achieved less packet loss comparing to when *c* is 40 and 50. Therefore, we reveal that these two factors actually influence each other to have a better sensitivity on grey-hole attacks. While *c* is too low, trust metric in TIGER is more likely to misjudge good nodes as malicious that it reselects the route frequently to cause more packet loss. On the other way, while *c* is too high, trust metric will take more historical records to determine whether a node is malicious, so as to create more packet loss as well. Obviously an optimal *c* is required to achieve best performance in packet loss and latency. In this scenario, 15 is the optimal value for *c* to achieve best result in packet loss and latency.

In addition, we also study TIGER_CW with different size of time windows. The results are shown in Table 4.2 and Figure 4.5 below.

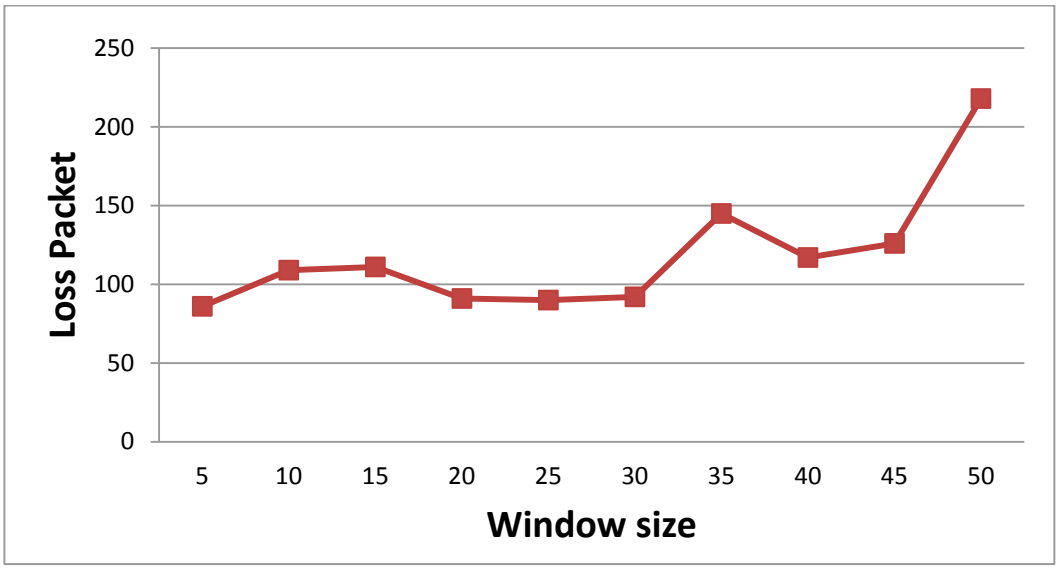

Figure 4.5 - TIGER_CW coefficient evaluation on packet lost

Table 4.2 - Packet sacrificed to detect node 12 black-hole attack in CW

TIGER_CW	***Timestamp window size***									
	5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
Packets sacrificed	2	3	4	5	6	7	8	9	10	11
Mean Latency (ms)	15.81	15.63	14.33	13.63	14.61	13.12	13.07	12.89	13.13	12.79

According to Figure 4.5, while the timestamp windows size at 20, 25, and 30 are achieving the similar results in packet loss. While the window size at 30 is slightly better as it also has lower mean packet latency. Moreover, the sensitivity level change through the window size in table 4.2 is similar as TIGER_AW which it decreases as the window size increases. The reasons for the results in Figure 4.5 are the same as

TIGER_AW. While the windows size is too small, the trust metric might over sensitive to misjudge some of the good nodes as malicious, so as to cause path reselection happen frequently to result in more packet loss. On the other hand, while the window size is too big, the trust metric can insensitive to the malicious behaviours in the network which will result in more packet loss as well. Now we have selected the optimal TIGER_AW coefficient and TIGER_CW timestamp window size which are used in next study to compare these two TIGER algorithms with ATSR.

Though the previous scenarios only have one run for each parameter set, we will have more random network scenarios in the next few sections.

4.3.3 TIGER vs. ATSR

In this study, we are targeting to investigate the detection sensitivity on malicious behaviours for the TIGER_AW and TIGER_CW. The total of 10 simulations run for each algorithm is conducted to show the trend of sensitivity level of malicious behaviours detection change, so as to reveal the time-insensitive problem of ATSR and confirmed the solution by TIGER. In each simulation, the node 12 starts conducting the black-hole attacks in different time period in each run, so as to study the behaviour of sensitivity level affected by the attacks regarding to the different time period. The time period is determined by the number of sessions generated, such as the following time samples in {0, 30, 60, 90, 120, 150, 180, 210, 240, 270} sessions. Results are shown in Figure 4.6 as below.

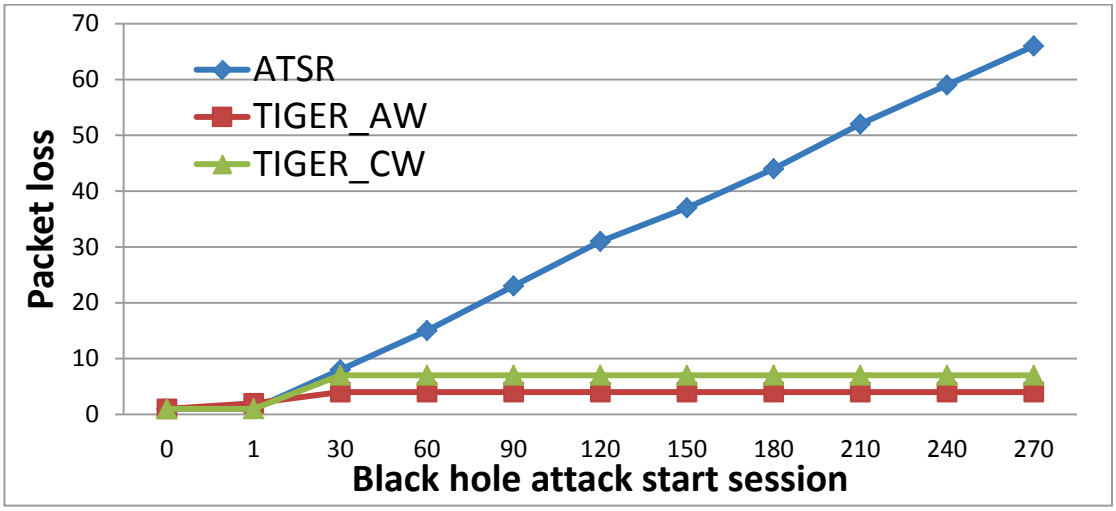

Figure 4.6 - The comparison of detection Sensitivity of ATSR, TIGER_AW and TIGER_CW over time.

It can be seen that, in Figure 4.6, the later of the black-hole attacks are performed by node 12, the worse of the detection performance of ATSR algorithm. While the TIGER_AW and TIGER_CW can detect the attacks instantly and only 4 and 7 on the packet loss over the time. This is because every interaction record with neighbours has timestamp on it, so as to let the TIGER algorithm only conduct up-to-date historical records. In such case, the trust evaluation by TIGER is always up-to-date as well. This has confirmed that the timestamp technique is effective on increasing the detection sensitivity of malicious behaviours for both short-term and long-term period.

Table 4.3 - ATSR vs. TIGER in packet loss and packet latency

Overall performance	***Algorithms***		
	ATSR	**TIGER_AW**	**TIGER_CW**
Mean Latency (ms)	12.33	13.17	13.12
Packet loss ratio	48.78%	8.33%	10.22%

In addition, the Table 4.3 above gives an overall performance view on these algorithms. In the same scenario, we have node 12 start perform black-hole attack after 200 packets are forwarded and the rest setups are the same. Both the TIGER_AW and TIGER_CW algorithms have similar mean packet latency at 13.17 and 13.12 milliseconds respectively, which are all about 1 millisecond higher than ATSR. It was because that TIGER_AW and TIGER_CW are all using timestamp techniques to increase the detection sensitivity level to effectively avoid the malicious node and select a more trustable route with sacrificing a longer distance to reach the destination node (1ms). On the other hand, ATSR has 48.78% of packet loss which can lead to a network collapse, while the TIGER_AW and TIGER_CW were able to maintain it at 8.33% and 10.22%. Obviously this 1 milliseconds sacrifice on each packet delivery can protect the network from collapse, thus it is worthy to be sacrificed.

The reason of ATSR algorithm has such high packet loss ratio is because it is insensitive to the most updated attacks. In other words, the malicious nodes can accumulate satisfied historical records so as to make the algorithm ignore their recent malicious behaviours.

4.3.4 TIGER_AW versus TIGER_CW

We now have more different malicious nodes deployment network scenarios to verify the stability of these two timestamp mechanisms which are AW and CW in TIGER

algorithm. We still use 10 x 10 wireless network grid, but with 50% of malicious nodes random deployment. There are two different scenarios, first scenario is all malicious nodes perform Grey-hole attack, and second scenario is all malicious nodes perform Black-hole attack. We have these two network scenarios run 10 times for each and for each algorithm (TIGER_AW and TIGER_CW) as well. These 10 random malicious nodes deployments are the same for both algorithms. We also change the session interval time from 4 seconds to 1 second so as to shrink the simulation time. The trust weight factor for both algorithms is 0.5. The simulation results are as following figures and tables.

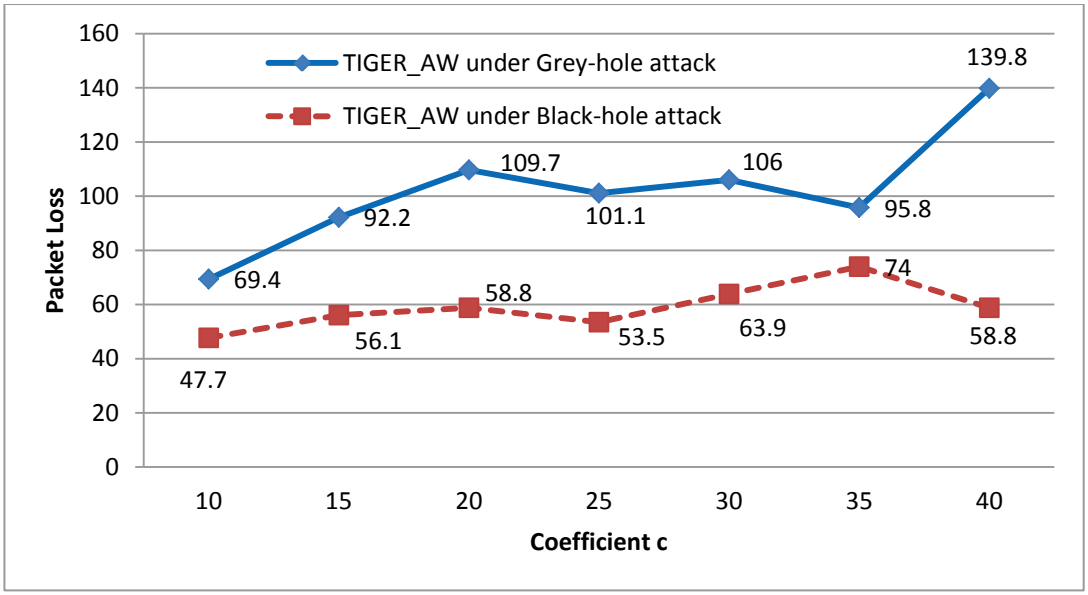

Figure 4.7 - TIGER_AW with 10 random scenarios

Table 4.4 - Means packet latency for TIGER_AW 2

TIGER_AW	*Coefficient c*						
	10	**15**	**20**	**25**	**30**	**35**	**40**
Means packet latency (ms) under Grey-hole attack	14.09	13.66	13.72	13.3	13.53	13.82	13.29
Means packet latency (ms) under Black-hole attack	14.43	14.13	13.91	13.78	13.54	13.81	13.57
Standard Deviation for latency (ms) under Grey-hole attack	1.66	1.48	1.83	1.38	1.54	1.54	1.38
Standard Deviation for latency (ms) under Black-hole attack	2.43	2.11	1.45	1.46	1.76	1.43	1.75
Standard Deviation for packet loss under Grey-hole attack	54.28	66.16	75.93	86.12	81.1	91.75	82.62
Standard Deviation for packet loss under Black-hole attack	35.99	37.75	28.61	29.11	41.37	48.09	29

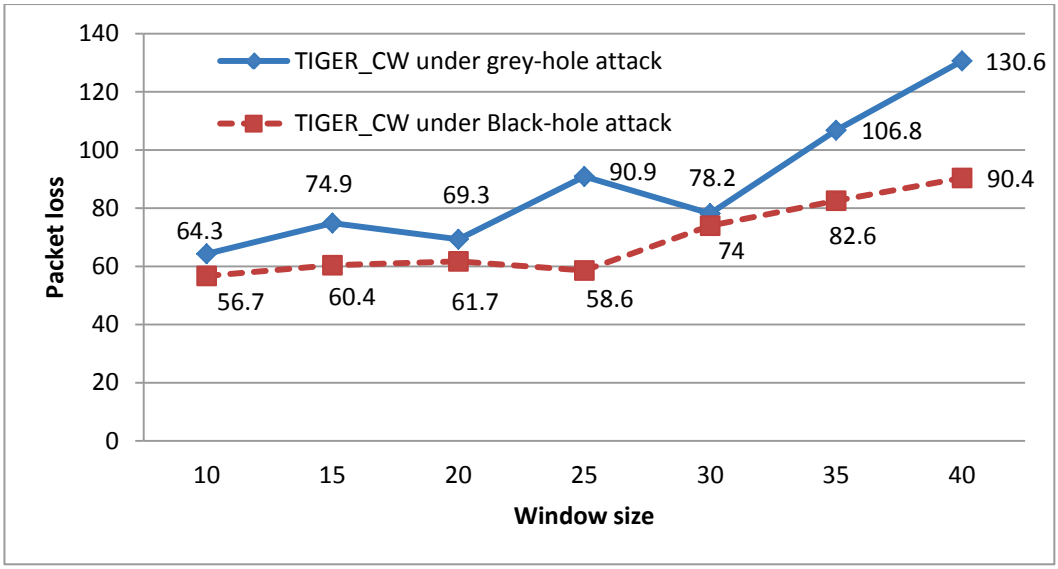

Figure 4.8 - TIGER_CW with 10 random scenarios

Table 4.5 - Means packet latency for TIGER_CW 2

TIGER_CW	*Window size*						
	10	**15**	**20**	**25**	**30**	**35**	**40**
Means packet latency (ms) under Grey-hole attack	14.38	14.43	13.51	13.42	13.7	13.74	13.1
Means packet latency (ms) under Black-hole attack	14.45	14.16	13.61	13.84	14.09	14.26	13.9
Standard Deviation for latency (ms) under Grey-hole attack	1.71	2.03	1.67	1.81	1.41	1.35	1.84
Standard Deviation for latency (ms) under Black-hole attack	1.83	1.97	1.74	1.66	1.74	1.77	1.86
Standard Deviation for packet loss under Grey-hole attack	36.42	44.89	36.34	75.84	28.55	44.19	88.72
Standard Deviation for packet loss under Black-hole attack	40.86	38.97	30.39	21.88	34.11	35.8	43.83

As shown in Figures 4.7 and 4.8, the packet loss number have similar trend to increase following by the increment of coefficient *c* in AW and window size in CW while under both grey-hole attack scenario and black-hole attack scenario. As the coefficient c and window size increase in AW and CW, the sensitive of malicious behaviour is decreasing that it causes more packet loss. While under the grey-hole attack scenarios, TIGER_CW can always achieve better packet loss than TIGER_AW in general speaking, as less packet loss overall in TIGER_CW's produced results in these 140 scenarios (10 random network scenarios for each coefficient c and window size samples set which is 7). The packet latency results were similar between these two algorithms from the view of table 4.4 and 4.5, but the standard deviation for packet loss

reflect that TIGER_CW has more stable performance under grey-hole attack in these ten random network scenarios, as its standard deviation in packet loss was almost half of TIGER_AW's. But while TIGER algorithm under black-hole attack, the results were showing TIGER_AW was achieving better results in packet loss and stable performance in overall. Moreover, while TIGER under black-hole attack, the packet loss achievements were better than while it was under grey-hole attack. This is because black-hole attack will drop all the packets it received which TIGER algorithm can quickly detect it and avoid the attacks. But for the grey-hole attack, as it drops the packet randomly, TIGER algorithm need to collect more historical records to detect it and avoid the attacks.

In summary, the AW timestamp in TIGER algorithm has better performance to detect focus attacks like black-hole attacks, and CW timestamp has better performance to detect random attacks like grey-hole attacks. This is because AW timestamp uses attenuation equation to let the trust metric in TIGER algorithm focus on the latest historical record and fade away older historical. For example, if the coefficient *c* in AW is 10 and there are 15 historical records of the neighbour so far, then latest record is about 0.9, then 2^{nd} latest is 0.82, and 3^{rd} latest is 0.73, and etc. In such case, the latest record should weight 12.6% of the final mark for the trust evaluation, and 2^{nd} latest record weighted more than 11.5%, and so on. For CW timestamp, it conducts certain amount of latest records for the evaluation which described in the Figure 3.2 in chapter 3. For example, if the window size is 10, then each record in the constant window equally weighted 10% of the final mark for the trust evaluation. In such case, while the same type of records are coming in a row, such as bad feedbacks, AW timestamp will judge the neighbour as malicious neighbour in a faster time comparing to CW timestamp. This is because the 12.6% + 11.5% in AW is greater than 10% + 10% in CW which mean AW has faster speed in decrease the trust evaluation mark. But when the good feedbacks are coming in random order in between the bad feedbacks, like a good feedback comes in after the bad feedback, we can calculate as 12.6% - 11.5% in AW is greater than 10% - 10% in CW. This is meaning good feedback as latest record that it is increasing the final trust evaluation mark in AW rather than like CW to keep the same mark. In such case, CW can detect the random malicious attacks in a faster speed.

4.3.5 Trust weight factor with TIGER_AW and TIGER_CW

In the ATSR algorithm, we can see that the trust weight factor has significant impact on the packet loss performance under different attack patterns. Therefore, here we will investigate how the trust weight factor impacts the two algorithms' behaviours. In this scenario, we are using the same network setup as previous section 4.3.4. The results are shown in the following figures and tables.

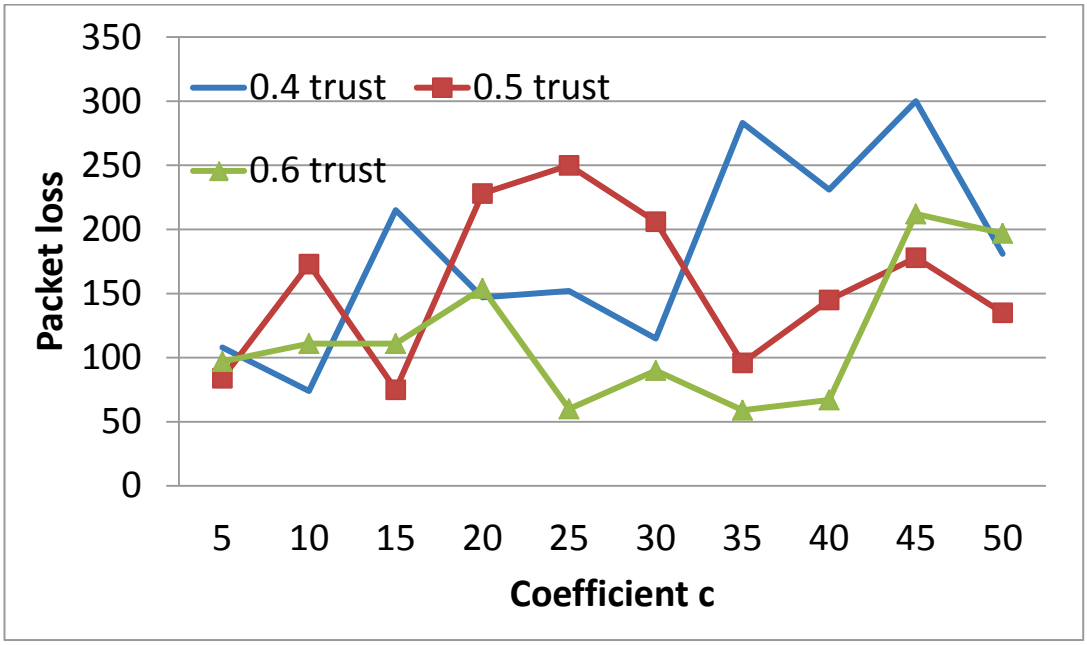

Figure 4.9 - TIGER_AW with different trust weight factors

Table 4.6 - Means packet latency for TIGER_AW 1

TIGER_AW	*Coefficient c*									
	5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
0.4 trust (ms)	13.36	13.37	13.56	13.08	14.08	13.05	12.9	12.93	12.99	13.01
0.5 trust (ms)	17.51	15.55	13.17	14.52	14.75	15.42	13.11	13.12	14.06	14.75
0.6 trust (ms)	18.39	15.4	15.16	13.75	13.41	14.49	13.29	15.53	14.86	15.62

As shown in Figure 4.9 above, as the trust weight factor is at 0.4, the TIGER_AW algorithm can achieve best packet loss with coefficient *c* equals to 10. When trust weight factor is at 0.5, the TIGER_AW algorithm was able to achieve best packet loss result with *c* of 15. Finally when the trust weight is at 0.6, the TIGER_AW algorithm was able to achieve best packet loss result with the coefficient *c* was equal to 25. From this we can see that, as the trust weight factor value was increasing in TIGER_AW algorithm, the optimal coefficient *c* value was increasing as well. This is because the higher value in trust weight factor, the more sensitive the algorithm can detect the malicious activities in the network. At the same time, the lower value in

coefficient *c*, the more sensitivity the algorithm to the malicious activities in the network as well. A certain level of sensitivity of the trust metric in TIGER_AW algorithm can help achieve better performance in packet loss, the higher value on trust weight factor requires the higher value of *c* to maintain the sensitivity level. But on the other hand, the incensement on trust weight factor makes more sacrifices on the distance metric which probably cause higher means packet latency, and this was confirmed on the results in table 4.6. As can be seen in the table 4.6, when trust weight factor was at 0.6, the mean packet latency generally higher than when it was at 0.4.

Additionally, we have run the TIGER_CW algorithm using the same scenario with trust weight factors of {0.4, 0.5, 0.6}. The results of the simulation are shown in table 4.7 and figure 4.8 in below.

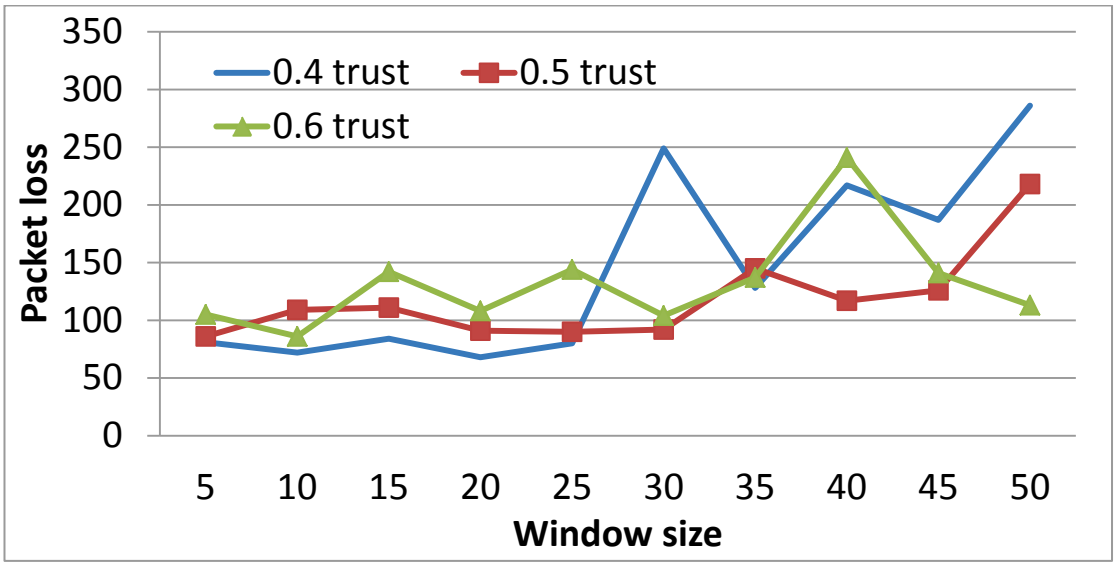

Figure 4.10 - TIGER_CW with different trust weight factors

Table 4.7 - Means packet latency for TIGER_CW 1

TIGER_CW	*Window size*									
	5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
0.4 trust (ms)	15.56	13.64	13.14	13.09	13.15	13.77	12.95	13.01	12.5	11.63
0.5 trust (ms)	15.81	15.63	14.33	13.63	14.61	13.12	13.07	12.89	13.13	12.79
0.6 trust (ms)	22.86	16.35	15.62	13.74	16.46	14.2	16.63	13.07	15.23	13.24

From the Figure 4.10, when the trust weight factor was at 0.4, and the window size between 5 and 25, the packet loss results were very well and the mean packet latency were reasonably low as well. When window size was at 5, the packet loss was a little high in latency. This is because the window size is so small that the trust metric is over sensitive to any activities in the network. This has caused some misjudgement on some of the legitimate nodes as well. In such case, the algorithm travel through a

further route that makes the latency high. Moreover, from the window size 30, the packet loss number was increase dramatically. This is due to the sensitivity level of malicious behaviours detection was dropping down to a stage which the algorithm ignored the malicious activities on some of the neighbours just because they were close enough to the destination. When these nodes close enough, they can gain enough points from the distance metric to win the higher position in the final mark than other legitimate nodes. Also, when trust weight factor was at 0.5, the performance of packet loss was quite stable as described previously. And the trust weight factor was at 0.6, the performance in packet loss was averagely high, as well as the packet latency. This is because the trust weight factor was too high in this scenario and at the same time the distance weight factor was low, that makes the algorithm a little bit get lost to the destination as it can be seen in the simulation the algorithm keep selecting alternative paths, so as to have more chances to encounter the malicious nodes and thus cause more packet loss.

From the weight factor simulation studies of TIGER_AW and TIGER_CW, we can see that high trust weight factor can increase the sensitivity level of malicious nodes detection. Moreover, when a suitable coefficient c and window size values for AW and CW are selected, the increase of trust weight factor should have these two parameters increase accordingly to maintain the same best performance of the TIGER algorithm. As these two parameters in the timestamp mechanism is to maintain certain level of sensitivity of malicious behaviour detection over time.

4.3.6 Energy consumption

The energy metric in TIGER_AW and TIGER_CW algorithms are inspired the logic in ATSR algorithm. In such case, the timestamp mechanism in TIGER algorithm should not affect the calculation of energy metric. Moreover, to confirm whether the timestamp mechanisms are affecting the TIGER algorithm in energy consumption, we measure the energy consumption of TIGER algorithm with ATSR algorithm. So first of all, we run the TIGER_AW, TIGER_CW, and ATSR on a 10 x 10 grid wireless network topology without any malicious attack, and compare the energy consumption of each. Same as previous scenarios, in each simulation run, there are 300 traffic sessions to be proceeded from node 1 to node 99, and the interval is 1 second. Each session also forwards 1 UDP

packet with 31 bytes data, and the time to live (TTL) is 128 milliseconds. We setup each node contain 1000 units of energy, and the transmission cost 50 units of energy per second, and receive cost 30 units of energy per second. The simulation results as follow figure and table. Moreover, the GPSR algorithm will be used as benchmark algorithm here to find out the energy cost for security features on the algorithms.

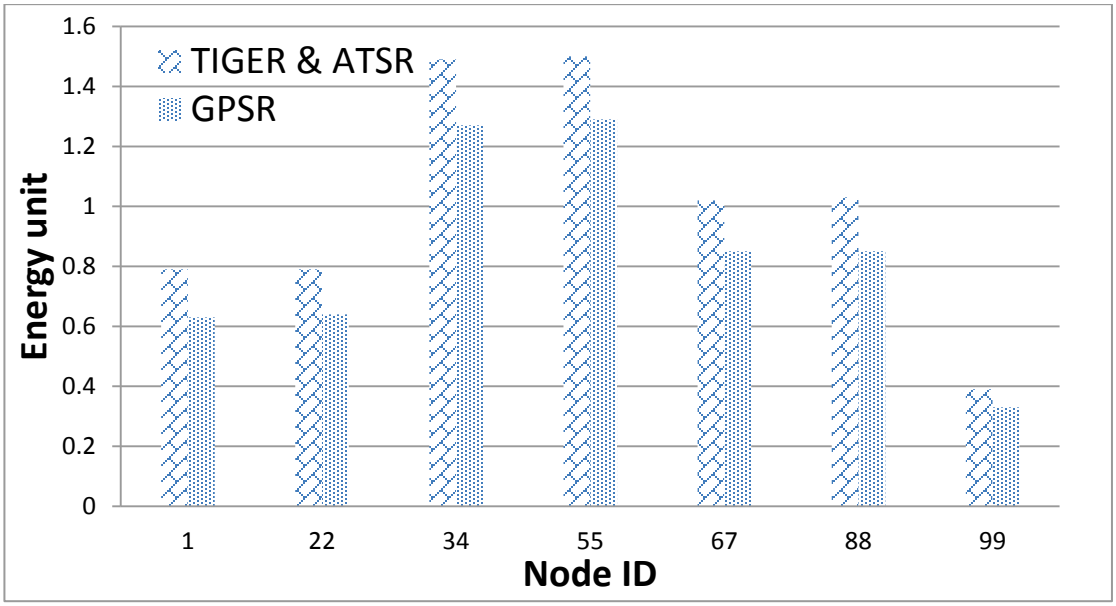

Figure 4.11 - TIGER vs. GPSR on energy consumption

Table 4.8 - TIGER vs. GPSR on energy consumption

Nodes	*1*	*22*	*34*	*55*	*67*	*88*	*99*
GPSR (per sec) Beacon exchange	0.43	0.43	0.96	0.96	0.59	0.59	0.19
TIGER (per sec) Beacon exchange	0.43	0.44	0.96	0.96	0.58	0.58	0.2
GPSR (per sec) Packets & Beacon	0.63	0.64	01.27	1.29	0.85	0.85	0.33
TIGER (per sec) Packets & Beacon	0.79	0.79	1.49	1.5	1.02	1.03	0.39

In the results, ATSR, TIGER_AW, and TIGER_CW have the exact same energy consumption in different situations which confirmed that the timestamp mechanisms were not affecting their energy consumption upon no attack scenarios. Additionally, we compare the TIGER algorithm with GPSR algorithm, which is a sole geographical routing protocol without any security awareness. As shown in Figure 4.11, it is obviously that the TIGER algorithm is consuming more energy units than GPSR. Moreover, in the table 4.8, while there was no data transmission occurred, the energy consumption is the same, while the nodes are transmitting packet, TIGER algorithm consumed 0.2 units per second more than GPSR for the additional security features, i.e. trust management. In additional, the nodes at the edge or corner of the network had less energy consumption. For example node 1 and node 22 were consuming less energy per

second than the nodes in the centre position such as node 55. This is because, those nodes at the corner or edge of the network have fewer neighbours to broadcast messages so that they receive and process less beacon messages.

After the finding of energy consumption for TIGER algorithm in the network, in this scenario we increase the traffic sessions from 300 to 900, so as to drain up the nodes batteries during the simulation. We first turn of the energy metric in the TIGER algorithm, and then set the energy weight factor as 0.2, 0.4, 0.5, 0.6, and 0.8 so as to investigate how the weight factor of energy metric has impact on the nodes battery life in the network by monitoring node 55 flat battery time. Node 55 is one of the forwarding hops between node 1 and node 99. The results are shown in the table 4.10 in below.

Table 4.9 - TIGER with energy metrics

Energy weight factor	*0.0*	*0.2*	*0.4*	*0.6*	*0.8*
Life time on node 55 (sec)	671.59	670.62	676.64	681.36	680.85
Mean Packet Latency (ms)	11.03	11.56	11.99	12.53	13.95
Packet loss	13	7	8	1	10

As can be seen in the table 4.9, follow the increment on energy weight factor, the life time of node 55 was increasing accordingly. Interestingly, as the energy weight factor is at 0.2, the life time on node 55 was decreasing a little. This is because the energy metric is too small to affect the algorithm for rerouting for energy load balance. Moreover, the mean packet latency was increasing accordingly too. This is due to the energy metric in TIGER algorithm try to select alternative routes to destination rather than a single route, so as to achieve energy load balance to extend the batteries life for the nodes in the network. On the other hand, the packet loss was decreasing accordingly due to the energy metric affect the algorithm to avoid the low battery nodes before they die. But the last one, when energy weight factor was at 0.8, the 10 packets loss were due to the algorithm lost the direction a little to the destination while the nodes' batteries in the network become empty as the energy weight factor was too high in this case.

In next study, we will put the 26 malicious nodes into the network and perform grey-hole attacks and black-hole attacks as shown figure 4.2. The attacks pattern is same as previous section: grey-hole attack will be performed between 30 and 60, 100 and 130, and after 200 packets which have been forwarded by the malicious nodes. The black-hole attack will be performed after 200 packets have been forwarded. There are 900 traffic sessions proceeded in the simulation. We run both TIGER_AW and

TIGER_CW algorithms with final trust weight factor with distance at 0.5 and energy weight factor with direct trust at 0.0, 0.2, 0.4, 0.6, 0.8. TIGER_AW coefficient *c* is set at 20, and TIGER_CW windows size is set at 20. As the ATSR algorithm achieved huge amount of packet loss in this scenario with energy metric turn on, so we didn't include the ATSR results here. The results of the simulation are in below figure and table.

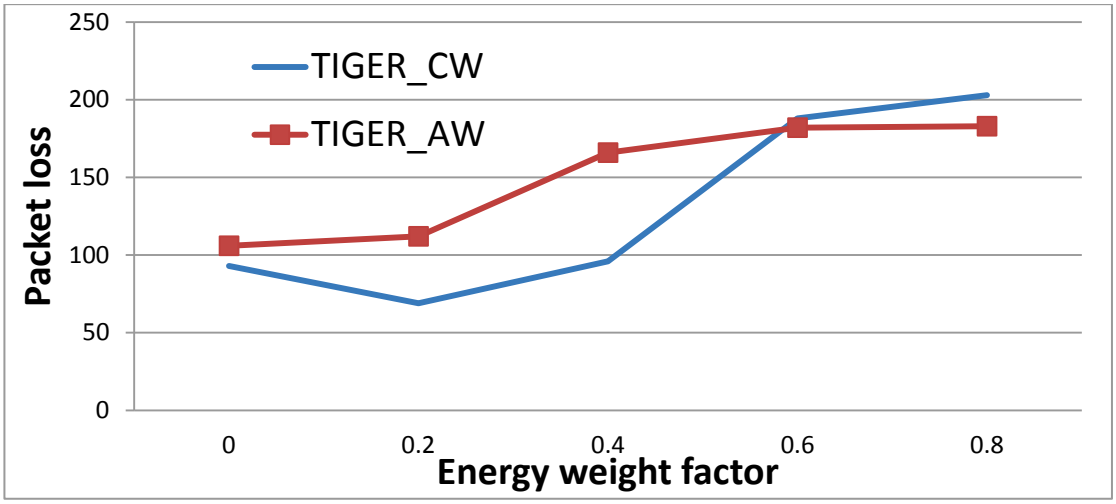

Figure 4.12 - Energy metric cost in TIGER packet loss

Table 4.10 - Means packet latency for TIGER_CW and TIGER AW

Means packet latency (ms)	***Energy weight factor***				
	0.0	**0.2**	**0.4**	**0.6**	**0.8**
TIGER_CW	14.85	13.39	13.51	13.41	14.59
TIGER_AW	15.2	13.58	13.25	13.97	14.02

As shown in Figure 4.12 upon malicious attacks, as the energy weight factor was increasing, the packet loss number was increasing accordingly as well. Because the energy metric is sacrificing the trust metric performances for the energy aware function. In TIGER_CW algorithm, as energy weight factor was at 0.2, it can achieve lower packet loss which even lower than the energy metric was turned off. This is because the window size was at 20, and its algorithm was too sensitive for this scenario that misjudged some of legitimate nodes as malicious nodes. The misjudgement can be clearly see during the simulation which the algorithm keep avoiding some of the legitimate nodes. In such case, the algorithm was keep switching the route which causes more chance to encounter the malicious nodes to increase packet loss. On the other hand, the packet loss was increasing as energy weight factor was increasing. The first reason is that the more priority on energy metric to achieve the energy efficiency, the more it sacrifices the trust performance. The second reason is the energy load balance feature

enforces the algorithm switching the route frequently, and this route switching creates more possibilities to be attacked by the malicious nodes. In addition, as shown for the mean packet latency in the table 4.10, the two algorithms have achieved higher latency when the energy metric was turn off. This is because in this network scenario in Figure 4.2, the energy metric affecting the TIGER algorithm on node 21 switch the route to a further neighbour which was node 42 to achieve energy load balance. The switch route to node 42 as next forwarding hop was actually a shorter route to the destination compare the one from node 33.

4.3.7 Summary

In summary, from the Figures 4.6 and Table 4.3, it can be seen that TIGER algorithm resolve the time-insensitive problem on historical records of ATSR algorithm. It is able to maintain the sensitivity level of malicious behaviours detection in the network over time. As there are two approaches for the timestamp for TIGER algorithm, which are AW and CW. We have extensive simulation studies to compare the AW and CW timestamp approaches. We found that AW timestamp has better performance in packet loss and stable performance while the network is under focus attacks such as black-hole attacks. CW timestamp has better performance in packet loss and stable performance while the network is under random attacks such as grey-hole attacks. Then next we conducted the impact of changing trust weight factor in the TIGER algorithm through simulation studies, and we found that the change of trust weight factor is affecting the selection of optimal coefficient value for both AW and CW timestamp mechanisms. While the trust weight factor is increasing, the optimal coefficient value should increase accordingly for both AW and CW timestamp mechanisms. This reveals the important of selecting an optimal trust weight factor, so as to have TIGER algorithm achieve best performance in different network scenarios and topologies. Finally, we have studied energy consumption on both TIGER_AW and TIGER_CW algorithms, and compare with the GPSR to indicate the extra energy consumption for the trust management. Moreover, the weight factor of energy metric can enforce the algorithm achieve energy load balance for energy efficiency, but this feature requires the sacrifice on the trust metric performance which will cause more packet loss upon malicious nodes attacks.

4.4 Case study 2: DTEGR vs. ATSR

In previous section, we have conducted an extensive study on the TIGER algorithm which introduces the timestamp techniques to tackle time-insensitive problem on historical records of ATSR algorithm. While we have found that the trust weight factor has significant impact on the network performance. As the network scenarios change, the trust weight factor needs to be adjusted to have the algorithm achieve best performance result. Therefore, in this section, we proposes a trust-based routing algorithm Dynamic Trust Elective Geo Routing (DTEGR) to resolve the optimal trust weight factor selection problem in ATSR algorithm by using two stages strategy . To validate our algorithm work, we first compare the DTEGR with ATSR algorithm, in terms of network performance such as packet loss and mean packet latency through different network scenarios, to investigate whether the DTEGR algorithm able to maintain the same performance without bothering on the selection of weight factors problem. Next, we discover the trade-off problem between energy metric and distance metric, and how their weight factors have impact on the algorithms' performance.

4.4.1 Simulation scenario setup

The topology of wireless mesh network is assumed as a 10x10 grid network, namely 100 nodes (i.e. smart meters) as shown in the figure 4.11 below. There are 26 malicious nodes have been deployed in the network, which can be seen in the figure 4.11 with red dots. All the malicious nodes will perform grey-hole attacks of dropping the received packets randomly, except for node 12. The node 12 will drop all the received packets to perform black-hole attacks. All malicious nodes will perform malicious attacks at the time when the first packet they have received. In the simulation, each scenario has 300 sessions will be preceded, and each session interval is 4 seconds. Each session will forward 1 UDP packet with 31 bytes data, and packet's time to live (TTL) is 128 milliseconds.

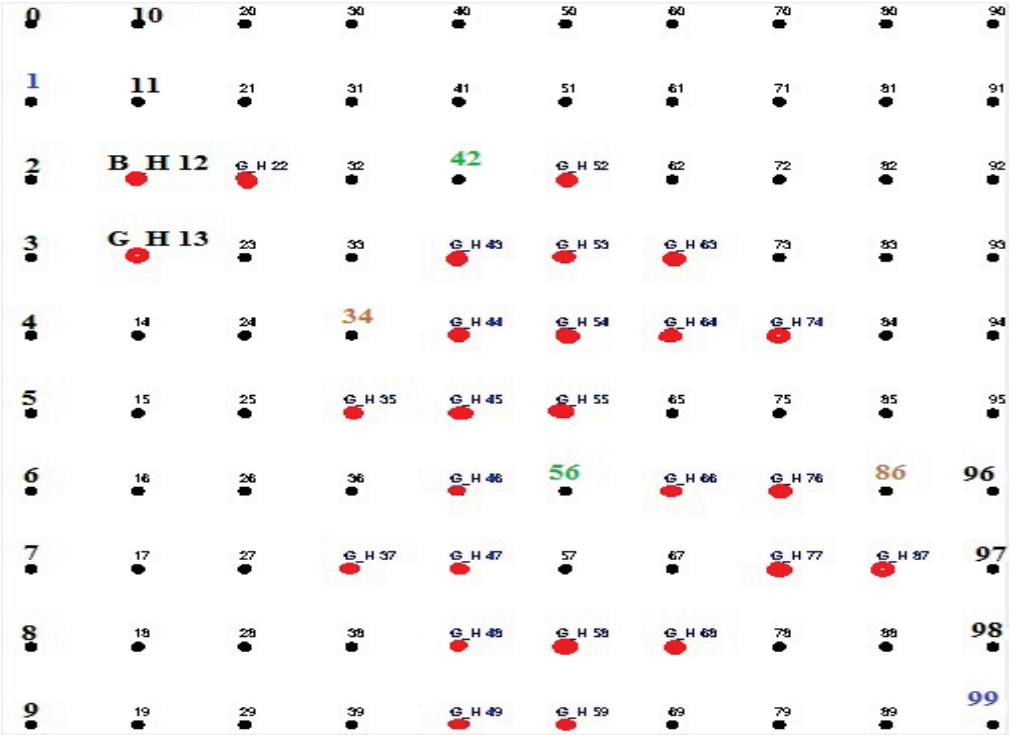

Figure 4.13 - 10x 10 wirelesses network topology for DTEGR simulation.

4.4.2 DTEGR vs. ATSR

We set up three network scenarios to compare DTEGR with ATSR algorithm, and each scenario has different source-destination node-pairs, so as to confirm DTEGR algorithm able to resolve the trust weight factor selection problem of ATSR and TIGER algorithms. We have run ATSR algorithm 7 times with different trust weight factor from 0.2 to 0.8, namely the distance weight factor is from 0.8 to 0.2. The trust weight factors {0, 0.1, 0.9, 1} are not included because they are either disable the trust metric or distance metric. The DTEGR is only need to be run once as it does not have weight factor problem and the threshold will be adjusted automatically by the algorithm. The initial threshold value for DTEGR is 0.7 as the average trust values of good behaves nodes is 0.8 which is calculated by using Equation 3.8 in chapter 3. We have the threshold value a little bit lower than the average trust value of good behaviour node so as to guarantee the sufficient neighbours in the trust forwarding list.

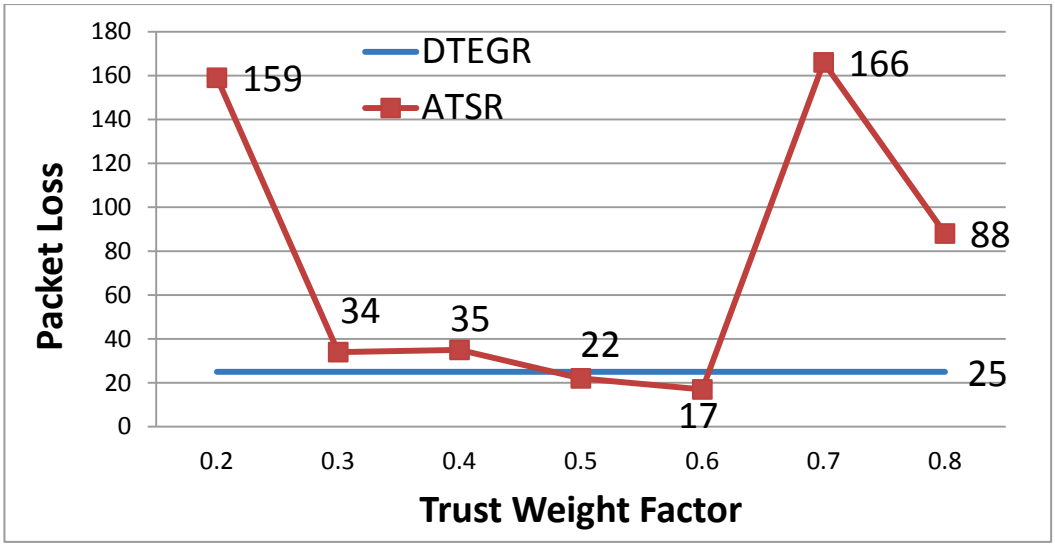

Figure 4.14 - Packets loss vs. Trust Weight Factor

Table 4.11 - Mean packet latency vs. trust weight factor

ATSR	*ATSR Trust Weight Factor*							*DTEGR*
	0.2	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	
Latency (ms)	11	13.3	15.88	15.76	15.86	34.58	16.87	13.42

In the first scenario, the source-destination node pairs is node 1 and node 99, and the simulation results are shown as above in table 4.11 and figure 4.12. According to the simulation results in Figure 4.14, the ATSR perform well when the trust weight factor is in the range from 0.3 to 0.6. When trust weight factor is set at 0.2, the impact of trust metric is too little to alert the nodes from grey-hole attacks. On the other hand, when the trust weight factor is set at 0.7 and 0.8, it made the distance metric too small to affect the algorithm select the shortest route to destination node. In this scenario, ATSR with trust weight factor at 0.6 has the best result as it has the least packets loss, and the DTEGR had lost 25 packets out of 300. Compare to the ATSR, when the trust weight factor at 0.6, the packet loss result achieved by DTEGR was slightly higher. However, if we look at the latency metric in table 4.11, DTEGR has less delay of 13.42 milliseconds, to ATSR of 15.86 milliseconds. Therefore, it can be concluded that the DTEGR algorithm can select shorter path (less average hop count) to the destination comparing to ATSR. In conclusion, the DTEGR had similar performance on packet loss with ATSR with optimal trust weight factor selected compare to ATSR which need extra process to find out the optimal weight factors.

In scenario 2, the source destination node pair is (56, 42). The result is shown as below in figure 4.15 and table 4.12.

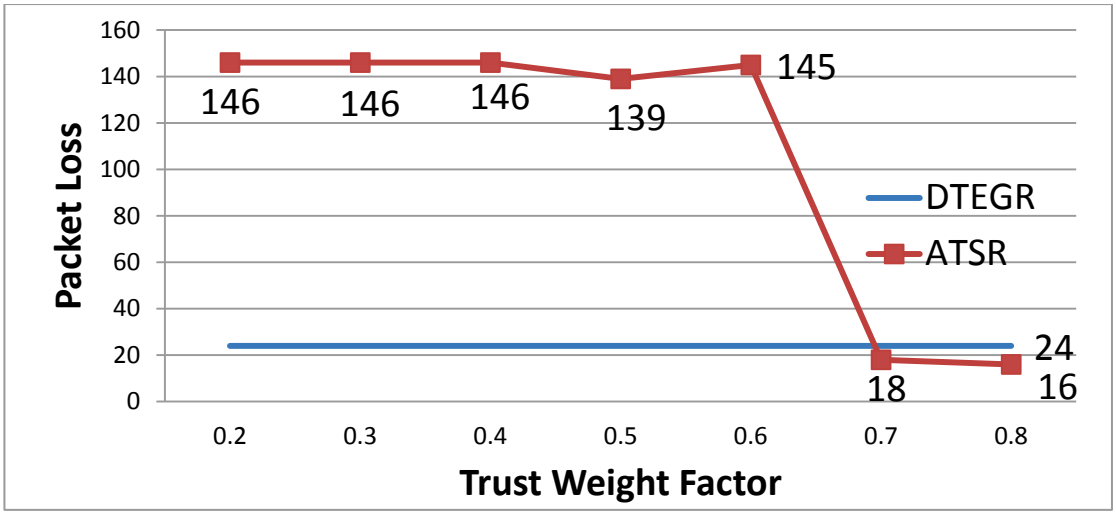

Figure 4.15 - Packets loss vs. trust weight factor 2.

Table 4.12 - Mean packet latency vs. trust weight factor 2

ATSR	*ATSR Trust Weight Factor*							*DTEGR*
	0.2	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	
Latency (ms)	**2.83**	**2.83**	**2.83**	**2.87**	**4.37**	**4.46**	**6.32**	6.3

According to the Figure 4.15, ATSR with trust weight factor from 0.2 to 0.6 had more than 40% of packets loss in the scenario 2, which can disable the network availability. The reason for high percentage packet loss was because node 56 was surrounded by malicious nodes, thus the distance metric take the higher priority and always search for the nearest path which are all malicious nodes. In such case, those malicious nodes were close enough to the destination to obtain enough point in the final evaluation score, so as to ignore trust metric and won the first position of the final score in the ATSR evaluation. When trust weight factor is increased to 0.7 and 0.8 in ATSR, it finally made the value of trust metric high enough over the distance metric so the malicious neighbours can be avoided. In the scenario 2, node 56 and node 42 are only 2 hops away that ATSR was still able to find a short path to the sink node even the distance metric became less priority than trust metric in the algorithm. Comparing to the scenario 1, when the trust weight factor is at 0.7, ATSR was trying to select a further route to the sink node just because trust metric has higher priority in the algorithm. But in scenario 2, ATSR has achieved the best result in packet loss when the trust weight factor is 0.8. DTEGR had lost 24 packets in this scenario which is slightly higher than ATSR, and its mean packet latency is at 6.3 milliseconds which indicate that it selected longer path to the destination node. In scenario 2, there were many malicious nodes appearing between node 56 and node 42, thus the secured paths were either traverse to

the left or right so as to get around the malicious nodes and reach the destination. The left path was the shorter path, but the neighbour with closer distance to the destination was on the right hand side. As the DTEGR algorithm only know one hop neighbours information. This was why DTEGR selected the right hand side neighbour which was the longer path rather than the left hand side neighbour which was the shorter path.

In scenario 3, we select node-pair (34, 86) as source and destination nodes respectively. The results are as following in Figure 4.14 and Table 4.13.

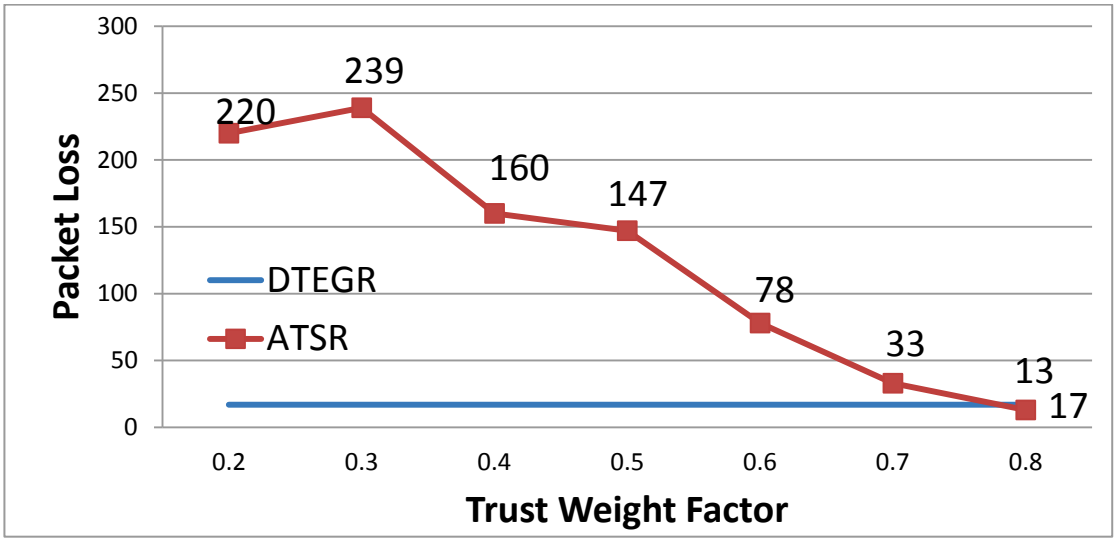

Figure 4.16 - Packets loss vs. trust weight factor scenario 3.

Table 4.13 - Mean packet latency vs. trust weight factor 3

ATSR	*ATSR Trust Weight Factor*							*DTEGR*
	0.2	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	
Latency (ms)	4.58	4.55	4.59	4.61	6.29	10.4	6.43	6.54

According to the Figure 4.16, the most suitable trust weight factor for ATSR was at 0.8 for this scenario as it cost the least packet loss and less mean packet latency. The packet loss is over 70% when the trust weight factor was at 0.2 and 0.3. This was because the distance metric has higher priority to make the algorithm ignored those malicious behaviours on the two of malicious nodes between source node and sink node. In such case, ATSR algorithm selected a shortest path to the sink node rather than a secured route. As the trust weight factor in ATSR was increasing, the packet loss was dropping significantly. It was the same reason for the scenario 2, because source node and sink node were close enough to each other that the algorithm did not get lost when distance metric become less important. DTEGR had 17 packets lost out of 300 in the

scenario 3, and mean packet latency was at 6.54 milliseconds which had similar performance to ATSR, but less packets loss.

From the three scenarios evaluated as above, it can be seen that ATSR is sensitive on the weight factors and always need to be reconfigured with different trust weight factors for different network scenarios such as 0.6 in scenario 1, 0.7 in scenario 2, and scenario 3 with 0.8. As the trust and distance weight factors in ATSR are static factors, it cannot use the same weight factor to fit all the scenarios. For example in figure 4.17, in scenario 1, 0.6 was the optimal trust weight factor that to achieve best result, but if used 0.6 in scenario 2 or 3, which will cause the network collapse as there are over 40% of packet loss. While the new DTEGR has can overcome the problem by using the two steps strategy. In all 3 scenarios, DTEGR was adaptable automatically to maintain the good network performance while ATSR require changes the weight factors manually to obtain the best performance.

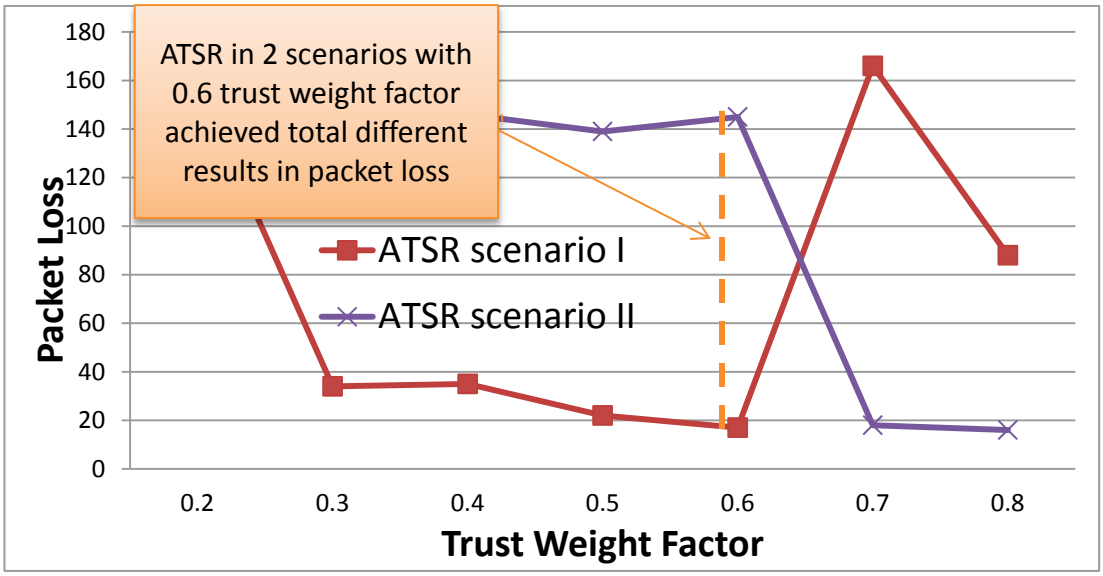

Figure 4.17 - Weight factor vs. scenario in ATSR

4.4.3 The stability of DTEGR

To further investigate the stability of DTEGR algorithm's performance, we setup different malicious attacks scenarios in the network, i.e. 30%, 40%, 50% malicious nodes of the network to compare with ATSR trust weight factor at 0.5. The 20% or less of malicious nodes is not sufficient to block the route between node 1 and node 99 to cause any packet loss in most of the scenarios as we have tried 10 random simulations

where 6 has 0% packet loss. While the 60% or more of malicious nodes in the network is too many that it can completely block the route from the node 1 to node 99 in most of the scenarios. We also have tried the 60% scenario ten times, which 7 of the simulations have the malicious nodes completely blocked the path to sink node. We deploy the malicious nodes in random positions for ten times for each level of attacks. For each scenario, there are 900 traffic sessions to be proceeded as we are testing the performance stability for the DTEGR algorithm, and the interval is 1 second. The traffic is travel from node 1 to node 99.

First of all, we setup 50% of the malicious nodes deploy in the network randomly with ten simulations, and the simulation results are shown in below.

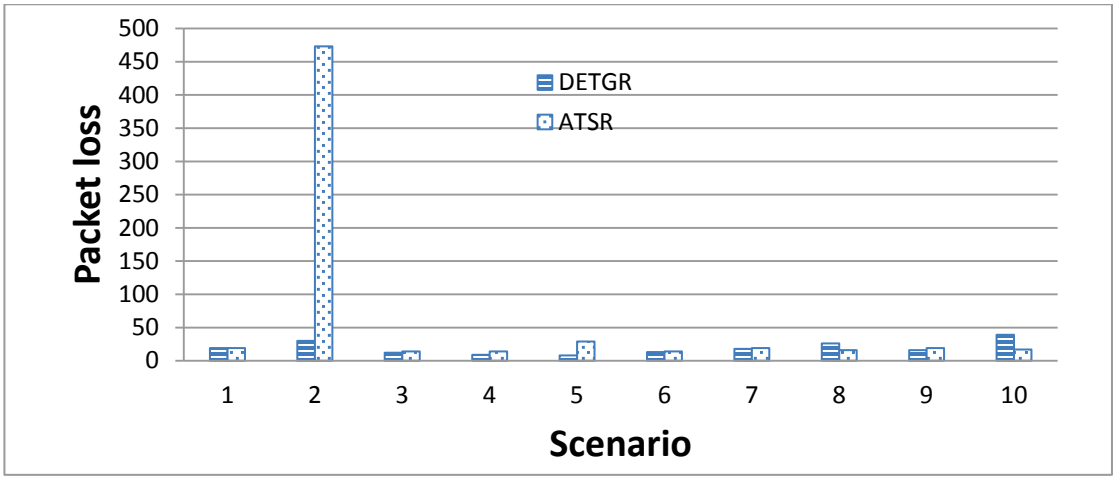

Figure 4.18 - DTEGR vs. ATSR under 50% malicious nodes attacks

Table 4.14 - Mean packet latency for DTEGR vs. ATSR under 50% attacks

50% attacks	*Network Scenario*									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
DTEGR packet latency (ms)	13.36	14.33	13.34	10.97	11.09	13.41	13.35	15.91	13.4	13.36
ATSR packet latency (ms)	13.36	12.7	13.38	13.5	10.99	13.5	16.1	13.48	13.97	13.45

From the figure 4.18, we can see the performance results in packet loss for DTEGR and ATSR algorithm were similar as the 0.5 for trust weight factor is an optimal trust weight factor for these random scenarios, except for scenarios 2, 5, and 10. In the scenario 2, ATSR has huge amount of packet loss was due to the trust weight factor was not high enough to enforce the algorithm detour to a longer route so as to avoid the malicious nodes attacks. This can be also seen in the table 4.14 packet latency, ATSR achieved lower packet latency compare to DTEGR it is because it selected a

shorter route to the destination but cannot exclude the malicious nodes from the route. In scenario 5, ATSR algorithm resulted in a higher packet loss in the simulation. This is actually the same reason as in scenario 2, the trust weight factor was low that it sacrificed so many packets to make the algorithm decided to select a longer but secure route to the destination. Finally in the scenario 10, DTEGR algorithm has a higher packet loss compare to ATSR algorithm, this was because DTEGR algorithm is using trust threshold to determine whether the nodes are malicious or legitimate. In such case, the nodes have to bad enough to fall out the safe forwarding list so the algorithm can select another neighbour as next hop which with shorter distance to the destination. In ATSR algorithm case, if there are two nodes have similar distance to the destination, the change on trust metric value will make the ATSR algorithm switch the route very quickly. For example, node A and B has the same distance to the sink node, DTEGR and ATSR both first select node A as next hop to forward the packet. Node A start dropping packets after a while, ATSR will quickly switch the route to node B as node A and B has the same distance to sink node, as long as trust metric decrease, ATSR algorithm will switch. But for the DTEGR case, it has to wait for node A to drop more packets so as to have the trust metric value below the threshold, then DTEGR can exclude node A from the list and switch the route to node B. From this example, ATSR algorithm is actually detect and avoid the malicious quicker than DTEGR algorithm, and thus ATSR has less packet loss.

Next we decrease the attack level from 50% of malicious nodes in the network to 40% with the same network setup as last scenario. The results of this scenario are shown in below.

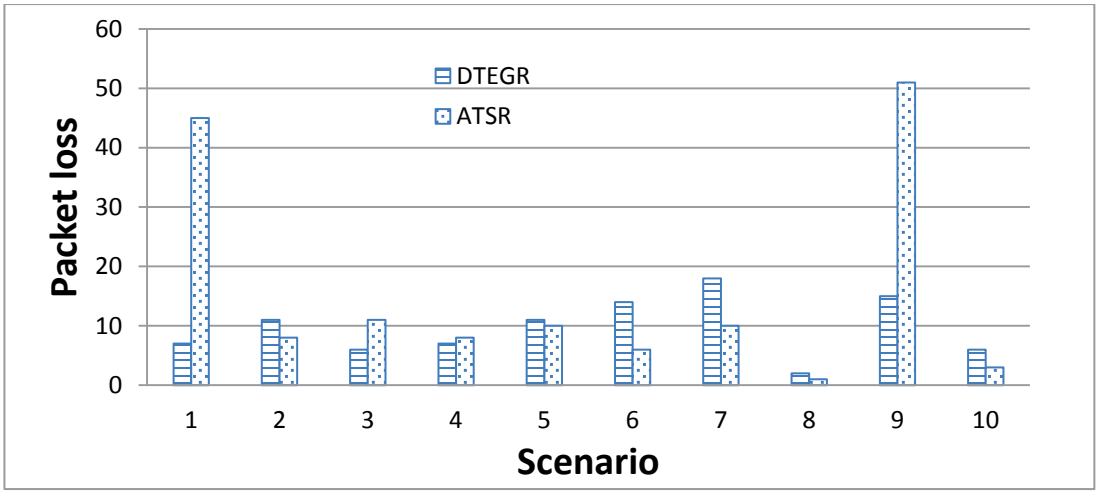

Figure 4.19 - DTEGR vs. ATSR under 40% malicious nodes attacks

Table 4.15 - DTEGR vs. ATSR in mean packet latency under 40% attacks

40% attacks	*Network Scenario*									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
DTEGR packet latency (ms)	12	13.25	11	11.26	13.28	13.44	13.47	11.05	13.31	11.06
ATSR packet latency (ms)	13.52	13.59	11.4	10.98	13.52	13.49	13.41	11.03	13.3	11.07

We can see that from Figure 4.19, the packet loss results obviously were decreased in general while under 40% of attack level rather than 50% of attack level. In the scenarios 1 and 9, ATSR had high packet loss number compare to DTEGR algorithm's performance in packet loss. Again, this is because the trust weight factor value not high enough, so it selects the shorter path but insecure route which causes large amount of packet loss. For the scenario 6 and 7, DTEGR algorithm had higher packet loss number compare to ATSR algorithm, this is the same reason in scenario 10 when under 50% malicious nodes attacks. DTEGR algorithm normally takes longer time to switch the neighbour as next hop compare to ATSR algorithm when this neighbour has similar distance as current next hop neighbour to the destination. But in overall performance for DTEGR and ATSR algorithm, DTEGR algorithm has more stable and better performance rather than like ATSR algorithm has high packet loss in scenarios 1 and 9.

Finally, we reduce the attack level again from 40% malicious nodes attacks to 30%, and the results are shown in below.

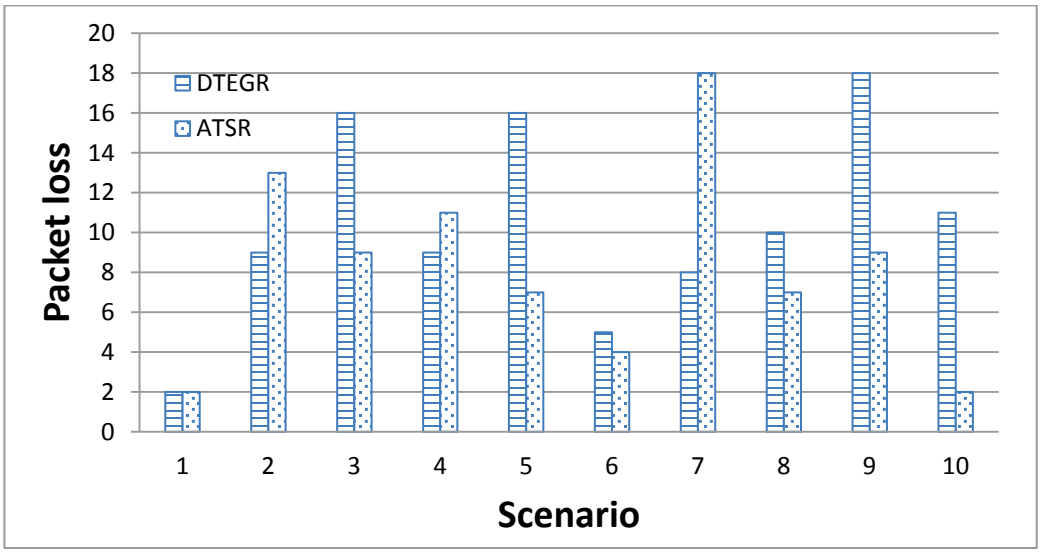

Figure 4.20 - DTEGR vs. ATSR under 30% malicious nodes attacks

Table 4.16 - DTEGR vs. ATSR in mean packet latency under 30% attack

30% attacks	*Network Scenario*									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
DTEGR packet latency (ms)	12	13.25	11	11.26	13.28	13.44	13.47	11.05	13.31	11.06
ATSR packet latency (ms)	13.52	13.59	11.4	10.98	13.52	13.49	13.41	11.03	13.3	11.07

From the figure 4.20, the scenario 7 had higher packet loss number with the same reason again as previous scenarios, which was the trust weight factor was not high enough. Also, for the DTEGR algorithms, in scenarios 3, 5, 9, and 10, packet loss number were high compare to ATSR's performance, it was also the same reasons as previous scenarios. But from these different scenarios' results under different levels of malicious nodes attacks, ATSR algorithm with trust weight factor at 0.5 was performing better and better as the attack level decrease.

Table 4.17 - DTEGR vs. ATSR on packet loss

DTEGR/ATSR	*Attack level*		
	30%	**40%**	**50%**
DTEGR standard deviation	5.06	4.95	13.26
ATSR standard deviation	5.01	17.58	143.99
DTEGR average	10.4	9.7	19
ATSR average	8.2	15.3	63.4

As can be seen in the table 4.17, as the attack level decreased, the standard deviation for packet loss on ATSR is getting better and better compared to DTEGR. This also has been reflected on the average packet loss such as when attack level was at 30%, ATSR actually perform slightly better than DTEGR algorithm. In such case, it can tell that ATSR trust weight factor at 0.5 can have better performance when attack level is low. Namely, trust weight factor at 0.5 is optimal value for the ATSR algorithm while the network is under low level of malicious attacks. As when attack level is low, it has higher probability of same distance alternative route to the destination, in such case ATSR algorithm do not require a high priority on trust metric to select a further route so as to avoid the malicious nodes to final reach the destination. When the attack level is high, it will require higher trust weight factor value, otherwise it has worse performance in scenario 2 upon 50% attack level. While DTEGR able to detect every malicious node in the route and avoid them without adjusting any parameter. DTEGR is performing better when under heavy attack as the trust threshold clearly defines the malicious nodes and legitimate nodes, then avoid the malicious nodes. But ATSR algorithm do not have a clear definition on which one is malicious and which one is legitimate, it just select

the neighbour with highest final score which come from distance and trust metric together. In such case, ATSR algorithm is more sensitive to the change of trust metric value so it can detect and avoid the malicious nodes in an earlier stage to save the packet loss. There are two factors to affect the results of packet loss number for both algorithms. First is mention before which is the speed of detect and avoid the malicious nodes, obviously, the faster it happen, the less packet loss will happen. The second factor is the attack level, when the attack level is low, while the algorithms are finding the alternative route, they have less chance to encounter the malicious nodes and cause the packet loss. On the other way round, when the attack level is high, the algorithms are more likely to encounter more malicious nodes to cause packet loss while they are finding the alternative route. In As ATSR will switch the route quicker once encounter malicious attack compare to DTEGR, and under light attack level in the network, the switch of the route is less likely to encounter the malicious nodes again. In such case, ATSR algorithm is more likely to achieve less packet loss at the end compare to DTEGR. On the other hand, while the network is under heavy attack, ATSR still able to switch the route quickly once encounter the malicious attack. But this time ATSR is more likely to encounter another malicious node after the switch, then ATSR quick switch back to the previous malicious node that it creates more packet loss. In such case, ATSR algorithm requires a higher value in trust weight factor, so the trust weight factor has higher priority to detour to a further but a secured route. In DTEGR algorithm, it might take a while to have trust metric collect enough bad feedbacks to let the malicious nodes to lose their position in the safe forwarding list, so the algorithm can avoid them. This can cause higher packet loss compare the ATSR algorithm. But once these malicious nodes lose their position in the safe forwarding list, the algorithm will not select them again until the safe forwarding list trend to empty, and this can prevent the packet loss reoccurred on the same malicious nodes. This situation is more likely happen for ATSR in the network which is under heavy attack. In such case, ATSR normally have better performance in low attack level network scenarios while the trust weight factor is at 0.5, and DTEGR is likely to have a better performance under heavy attacks. The network under heavy attack requires ATSR algorithm assign a higher trust weight factor to have better performance in packet loss. In conclusion, , DTEGR resolve the trust weight factor selection problem in ATSR algorithm, and it can maintain the good performance level upon different network scenarios without any

parameter adjustment, where ATSR algorithm requires extra process to find optimal factors for each scenario to achieve such performance.

4.4.5 Energy consumption

In this scenario, we introduced the energy-aware functionality to DTEGR algorithm. The DTEGR algorithm has 2 steps, the first step is to use trust metric to generate a safe forwarding list for next hop selection, and second step will find out the neighbour in the list with shortest distance to the destination. DTEGR algorithm resolved the weight factor selection problem between trust and distance metric in ATSR. But here we have to again put the weight factor between energy and distance metric so as to emerge the energy-aware functionality into the algorithm. Namely, the energy-aware DTEGR algorithm is at the second step using weight factor technique to combine it with distance metric. We set the traffic sessions as 900, and focus on node 55's battery life, so as to see whether energy metric can extend node 55's battery life time. We were using the energy weight factor value sample set {0, 0.2, 0.4, 0.6} respectively in this scenario, and the results are shown in below.

Table 4.18 - DTEGR with energy metrics

Energy weight factor	*0.0*	*0.2*	*0.4*	*0.6*
Life time on node 55 (sec)	671.59	682.21	692.41	646.48
Mean Packet Latency (ms)	11.51	11	12.63	23.48
Packet loss	17	5	0	192

As can be seen in the table 4.18, as the increment on energy weight factor, the life time of node 55 was increasing accordingly until reach 0.6. This is because after 0.6, the energy weight factor was too high to have the distance metric look for the direction to destination for the algorithm. In such case, the algorithm was more likely to find the next hop by the energy metric rather than the distance and this make the algorithm select a very long route to the destination so as to consume more energy (i.e., from overall network point of view) to achieve same task. This can be seen in the packet latency metric which was very high. Moreover, the high packet loss number and packet latency when energy weight factor was at 0.6, this is due to the nodes in the center of the network have the batteries exhausted before the 900 traffic sessions were completed one by one. The high priority in energy metric that it makes the DTEGR algorithm select a long distance route to avoid those low batteries nodes, and at the end it even

can't find the way to the destination that cause the high packet lost. In such case, there was no valid route to the destination that causes the packet loss. While the energy weight factor was at 0.4, the battery life on node 55 was extended by about 20 seconds, but at the same time, the packet latency was increased due to the energy load balance. The energy metric in DTEGR algorithm try to select alternative routes to destination rather than a single route, so as to achieve energy load balance. From the packet loss point of view, the packet loss number was decreasing accordingly due to the energy metric affect the algorithm to avoid the low battery nodes before they die. As can be seen that, while energy metric was turned off, it cause 17 packet loss due to flatten batteries. While the energy metric was at 0.4, there was no packet loss at all. The results are indicating the energy metric can help algorithm predict empty batteries node so as to avoid them before they exhaust to save the packet loss.

In the last scenario for the DTEGR algorithm, we deploy 26 malicious nodes into the network to perform grey-hole attacks. The attacks will be performed at the beginning of the simulation all at the same time. The network setup is same as figure 4.2 in the TIGER simulation section. There are 900 traffic sessions will be proceeded, and each traffic session is 1 second interval. We use the energy weight factor sample value sets {0, 0.1, 0.2, 0.3, 0.4}. The result are shown in the below.

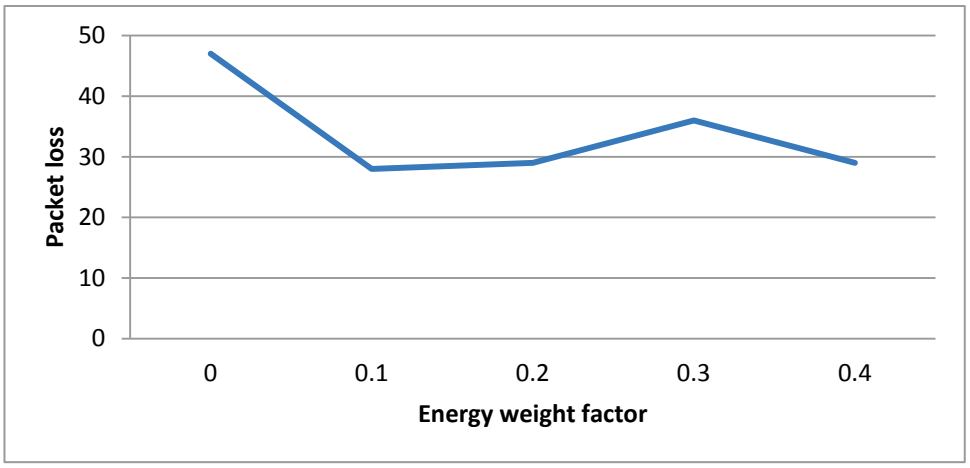

Figure 4.21 - Energy metric cost in TIGER packet loss

Table 4.19 - Mean packet latency for DTEGR

Means packet latency (ms)	*Energy weight factor*				
	0.0	**0.1**	**0.2**	**0.3**	**0.4**
DTEGR	13.5	15.5	13.32	14.45	17.01

From the figure 4.21, we can see that, once the energy metric was turn on, the packet loss number were similar between different energy weight factors. When the

energy metric was turn off, the high packet loss result was caused by the flat battery nodes, as almost half of packet loss were due to this reason. The J-Sim Tools has showing every process of the traffic session in the simulation. It is easy to find out what cause the packet loss. In such case, when the energy metric was turn off, excluded the packet loss number which caused by the exhausted battery, the packet lost number were similar to when energy metric was turn on. These results have confirmed the energy metric did not affect the performance of trust metric in DTEGR algorithm. As DTEGR is a two steps algorithm, it first has the trust metric to filer the malicious nodes out of the safe forwarding list, then use the energy and distance to select the neighbour as next hop to forward the packet. In such case, the energy metric cannot affect the trust metric. The packet latency results in table 4.19 have also confirmed the sacrifice on the distance to achieve energy load balance. After the energy metric was turn on, the packet latency results were increased except while energy weight factor was at 0.2. The reason for low packet latency result when energy weight factor was at 0.2 is because of the energy load balance feature in the algorithm. In the network, node 21 had the default choice for the next hop is node 33 in DTEGR algorithm while energy metric is turn off as node 33 is closer to the destination. But when the energy is turn on in the DTEGR algorithm, as the energy weight factor increase, the algorithm will more prefer node 42 as next hop. This is because node 42 is closer to the edge of the network compare to node 33. In such case, the energy consumption per second is less than node 33. In this network scenario, route through node 42 actually is a closer secured route to the destination rather than through node 33. This is the reason why the packet latency is lower while energy weight factor at 0.2.

4.4.6 Summary

In this section of simulation studies, we first confirmed that the DTEGR algorithm able to resolve the weight factor selection problem between trust metric and distance metric in ATSR algorithm by introducing the two stages strategy. The first stage it uses trust metric with trust threshold to generate a safe forwarding list, then the second stage is to use the distance metric to select the next hop from that safe forwarding list. We also setup more scenarios with different malicious attack levels to verify the stability of DTEGR algorithm. We found that DTEGR is able to maintain the performance level

through different network scenarios, and better performance under heavy malicious attack compare to ATSR algorithm. ATSR algorithm requires adjusting different trust weight factor so as to perform well in different attack level scenarios, where DTEGR does not need to have any adjustment on any parameter. Moreover, the energy-aware functionality in DTEGR requires a weight factor to combine it with distance metric. Through the extensive simulation studies on the energy-aware functionality in DTEGR, it confirmed the energy metric able to help DTEGR achieve the energy load balance, so as to extend the batteries life for the nodes in the network. But at the same time, this energy-aware functionality sacrificed the distance metric performance to achieve the energy load balance. In such case, the selection of optimal weight factor between energy and distance metric is another problem waiting to be resolved.

4.5 Case study 3: FEATGR vs. TIGER and DTEGR

In previous two case studies, we have identified that the weight factors of different routing metrics have significant impact on the algorithms' performance. Though DTEGR algorithm resolved the weight factor selection problem between the trust and distance metrics, it still has the weight factor selection problem remain for the energy metric and distance metric, also the direct trust and indirect trust metrics. In this section, we propose the Fuzzy-based Energy Aware Trust Geo Routing (FEATGR) algorithm to use a new approach, fuzzy logic to resolve this hard problem on how to select the optimal weight factors. The FEATGER algorithm is proposed to overcome this problem and make the algorithm scalable to add new metrics if need. To verify this, we first look at the packet loss ratio and mean packet latency performance through different network scenarios to investigate whether the FEATGR algorithm is able to perform well as DTEGR. Then we look at how convenient to add energy aware metric in the FEATGR algorithm using fuzzy logic approach.

4.5.1 Simulation scenario setup

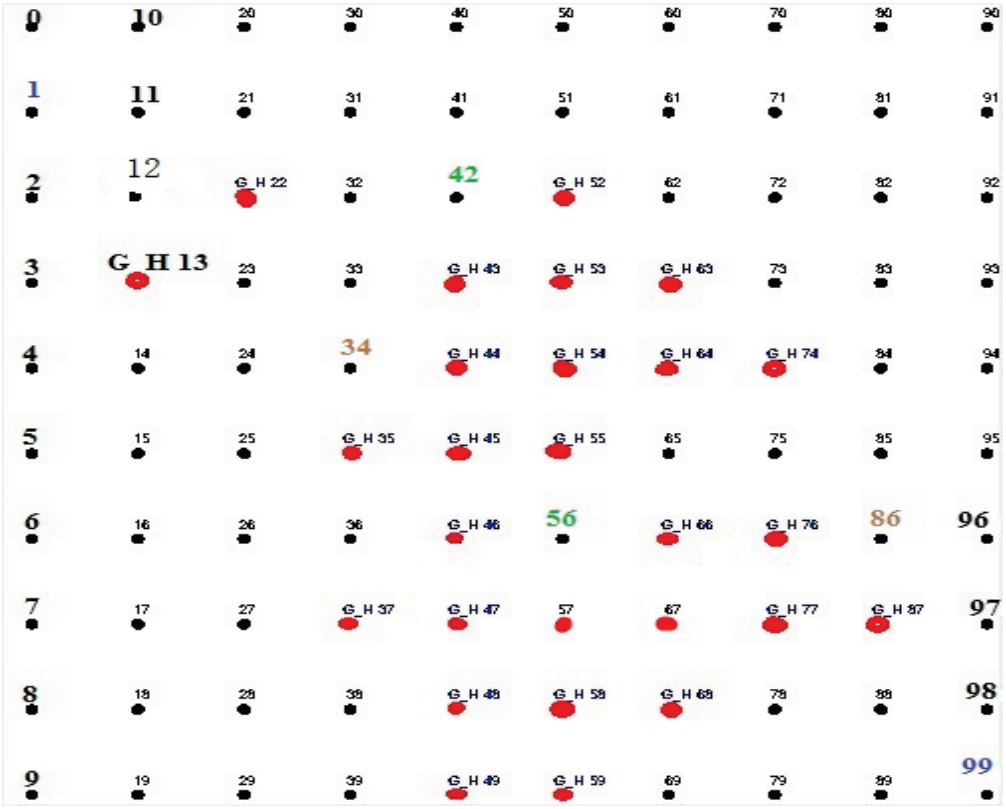

Figure 4.22 - Simulation topology for FEATGR

The network topology for the simulation studies is a 10 x 10 grid network i.e., 100 nodes are setup in the network shown in Figure 4.22 above. In addition, there are 900 UDP packets with 31 kilobytes for each packet will be transmitted in each scenario. The interval for the transmission is one second, and the time to live (TTL) is 128 milliseconds. We conduct simulation to compare the new FEATRG with previous TIGER and DTEGR algorithms.

We have introduced the warm up mechanism which is the equation 3.10 for trust evaluation in FEATGR algorithm and presented in chapter 3. The warm up mechanism is to make sure the algorithm collect enough information to make a judgement on the neighbours whether they are malicious. In such case, this can prevent the misjudgement of the legitimate nodes as malicious. The equation 3.10 also can be seen in the below.

$$b_{good} + b_{malicious} < 10 \Rightarrow t_{direct} = 1$$

$$10 <= b_{good} + b_{malicious} <= 30 \Rightarrow t_{direct} = \frac{b_{good}}{b_{good} + b_{malicious}}$$

Also, we apply the constant window (CW) timestamp technique from TIGER algorithm into FEATGR and DTEGR as it has been confirm in the previous section that CW has better performance under random attack. As the warm up mechanism can cause more packet loss at the beginning. In such case, for the benchmark algorithms TIGER and DTEGR, we apply the warm up mechanism and CW onto them so all these three algorithms can be at the same line. The warm up experiences is 10, and CW size is 30 for all the simulation in this section. Moreover, we set energy weight factor at 0.4 in DTEGR algorithm as FEATGR algorithm has already had the energy metric considered all the time. The reason to select 0.4 as energy weight factor for DTEGR algorithm as it is the optimal weight factor in the algorithm which we have found out from the last section.

For the attacks, we assume that there are 27 malicious nodes deployed in the network, which is intending to block the route between node 1 and 99 except leaving only one trustable route through these malicious nodes. These 27 malicious nodes will conduct grey-hole attacks after 30 sessions proceeded by the malicious nodes and the attacks stop after 60 sessions by the malicious nodes. Then grey-hole attacks start again after 100 sessions by the malicious nodes, and stop again after 130 sessions are preceded. Finally, the grey-hole attacks start again after 200 sessions. The figure 4.3 is able to present the attack pattern in the previous section in this chapter. Here the grey-hole attacks are initiated at different time is to validate the detection sensitivity of these three algorithms. The positions of these malicious nodes are highlighted by red colour, as shown in Figure 4.22 above.

4.5.2 The comparison study for FEATGR, TIGER and DTEGR

Firstly, we investigate these three algorithms upon three traffic flow scenarios with different source - destination pairs, so as to compare the adaptability of these three algorithms to the network changes. The first traffic flow is from node 1 to node 99 where the packet travels from left top corner of the network to the other end of the corner. In second scenario, the source is node 34 and destination is node 86, in which the source node is surrounding by the malicious nodes on the direction to the destination. The third traffic flow is from node 56 to node 42, where the path between source and destination is blocked by the malicious nodes. In these three traffic flow scenarios, we

compare the FEATGR with DTEGR algorithm and also the TIGER algorithm with different trust weight factors. The results are shown as follows.

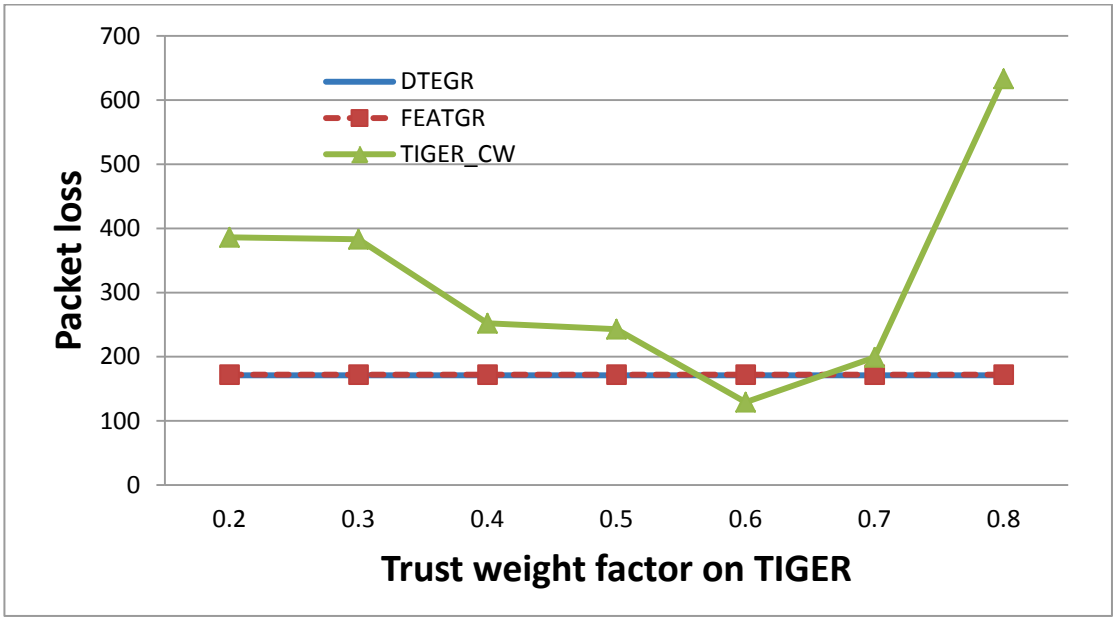

Figure 4.23 – Scenario 1: Packet loss of TIGER vs. FEATGR and DTEGR

Table 4.20 - Scenario 1: TIGER, FEATGR, and DTEGR in packet latency

TIGER	***TIGER Trust Weight Factor***							***FEATGR***	***DTEGR***
	0.2	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**		
Latency (ms)	11.1	12.48	14.75	15.01	17.03	16.97	48.38	16.2	16.54

In the 1st scenario, the traffic flow is transmitting from node 1 to node 99. We are targeting to investigate the path finding behaviour of three algorithms under heavy cyber-attacks environment. It can be seen that, in figure 4.23, the TIGER can achieve the best result in packet loss when trust weight factor is at 0.6. While the FEATGR and DTEGR does not have such problem as it is not sensitive on the weight factors between distance and trust metrics. They can achieve lower packet latency than TIGER and similar performance in packet loss when the. In this scenario, FEATGR and DTEGR almost have the same performance in packet loss where FEATGR has 172 packets lost and DTEGR has 171 packets lost in the simulation. As the packet loss results between these two algorithms are so close that in the figure 4.23 the curve for DTEGR algorithm is hiding behind the FEATGR curve. Moreover for the latency in table 4.20, as the trust weight factor increase in TIGER, the packet latency was increasing accordingly as well. This is because when the trust weight factor increased, the trust metric in the algorithm will have higher priority to affect the algorithm to avoid the malicious nodes, so as to take longer but secured route to the destination. FEATGR and DTEGR algorithms both

have similar packet latency results. But compare to TIGER, they have higher packet latency than TIGER while the trust weight factor is less than 0.6 in TIGER. As mentioned before, TIGER sacrificed the transmission packets to achieve such low packet latency.

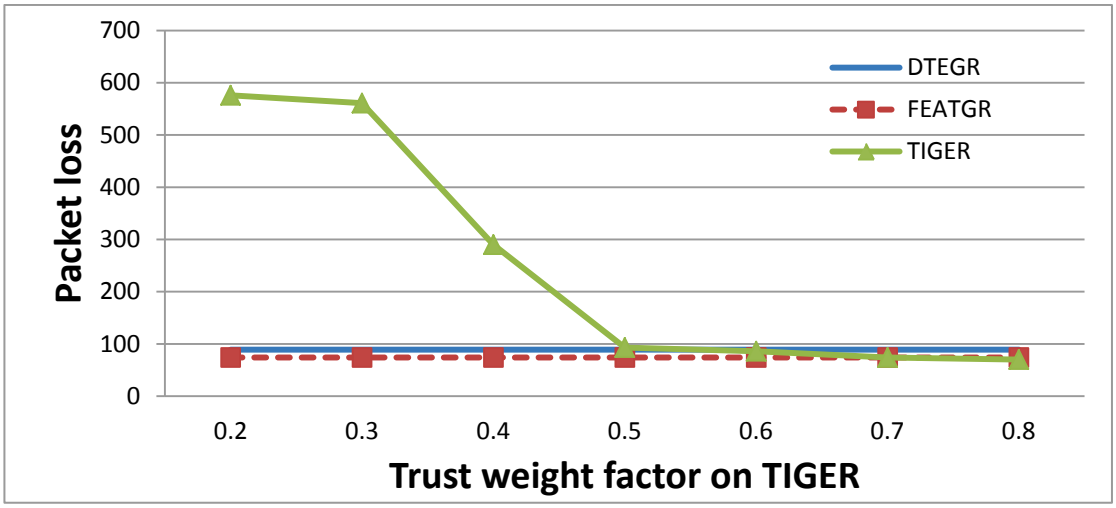

Figure 4.24 - Scenario 2: Packet loss of TIGER vs. FEATGR and DTEGR

Table 4.21 - Scenario 2: TIGER, FEATGR, and DTEGR in packet latency

TIGER	***TIGER Trust Weight Factor***							***FEATGR***	***DTEGR***
	0.2	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**		
Latency (ms)	4.57	4.5	4.56	5.98	6.01	6.03	6.05	6.06	6.05

In the second scenario, the packets are sending from nodes 34 to node 86, so as to study whether the new FEATGR algorithm will be adaptable enough to select the trustable path to destination by sacrificing the distance. The TIGER is achieving the least packet loss when the trust weight factor is change to 0.8 for this scenario. When the trust weight factor is at previous 0.6, the packet loss is higher than the FEATGR's results and similar to DTEGR's results. This is because the source node is surrounding by the malicious nodes, and it requires higher priority on trust metric to enable the TIGER algorithm select a longer but more trustable neighbour as next hop to forward the packet. This is why the trust weight factor has to be reconfigured at 0.8. In this scenario, FEATGER achieved a better packet loss result compare to DTEGR, this is due to DTEGR algorithm has encountered more malicious nodes that cause additional packet loss. The node 56 is one of the hops to forward the packet from node 34 to 86. The node 56 has two choices to forward the packet which are through node 75 or node 77 with the same distance to the destination. DTEGR select node 77 first that cause the additional packet loss, while FEATGR select node 75. As node 75 and 77 has the same

distance to the destination node 86, node 75 and 77 whoever come in the algorithm first will be selected as next forwarding hop. Then node 77 comes into DTEGR algorithm first, and node 75 comes into FEATGR algorithm first. For the packet latency result in table 4.21, the packet latency was increasing while the trust weight factor is increasing in TIGER, and FEATGR and DTEGR has higher latency. The reasons for these are same as the first scenario.

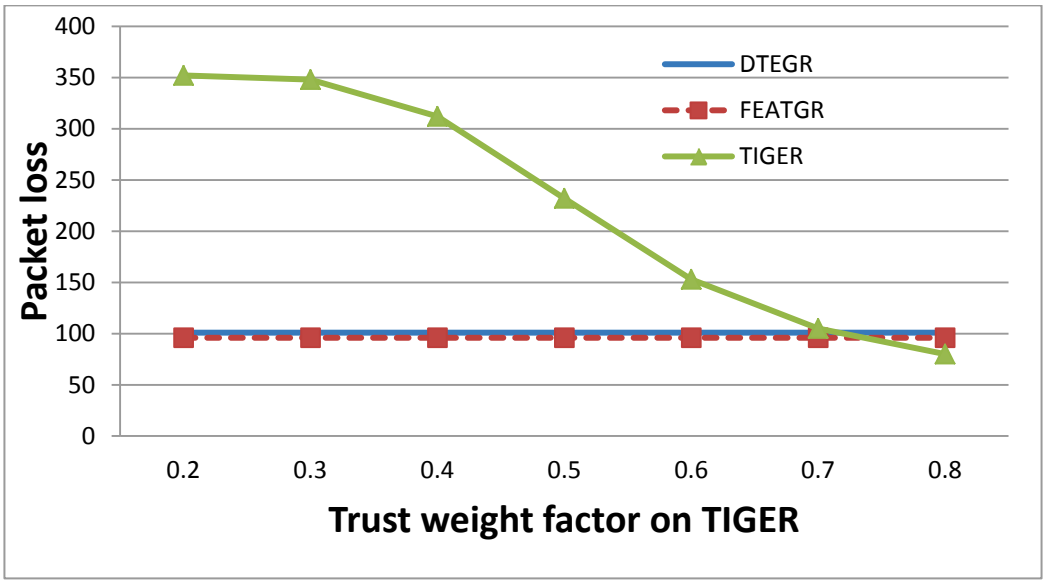

Figure 4.25- Scenario 3: Packet loss of TIGER vs. FEATGR and DTEGR

Table 4.22 - Scenarios 3: TIGER, FEATGR, and DTEGR in packet latency

TIGER	***TIGER Trust Weight Factor***							***FEATGR***	***DTEGR***
	0.2	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**		
Latency (ms)	2.85	2.83	4.56	3.88	5.19	5.55	5.78	5.72	5.53

In the third scenario, the traffic flow is between node 56 and node 42, which the results are sort of similar to the scenario 2. FEATGR and DTEGR algorithms achieved a low packet loss performance compare to TIGER in overall. The TIGER can only perform well as the weight factor is configured into a higher value of 0.8.

It can be seen that, all these 3 scenarios are in the same network deployment, but with different source - destination node pairs. FEATGR can handle all these different scenarios with acceptable network performances by using the dynamic fuzzy logic approach. If we use TIGER algorithm to conduct these three tasks, a fixed set of trust weight factors is not suitable for all the cases, it should manually find the proper weight factors to have the best performance. Moreover, FEATGER algorithm is able to achieve similar or even better results than DTEGR algorithm.

4.5.3 FEATGR vs. DTEGR under different attack levels

In addition, we have conducted more simulation studies to compare the FEATGR and DTEGR algorithms upon different attack levels. First of all, we setup the attack level at 30% (i.e., 30 malicious nodes) of the network, and there are ten scenarios with random deployment of these malicious nodes perform grey-hole attacks with the same attack pattern in figure 4.3 as previous scenarios did in this section. Each scenario performs 900 UDP packets traffic with 1 second interval. The simulation results of these ten scenarios are shown in below.

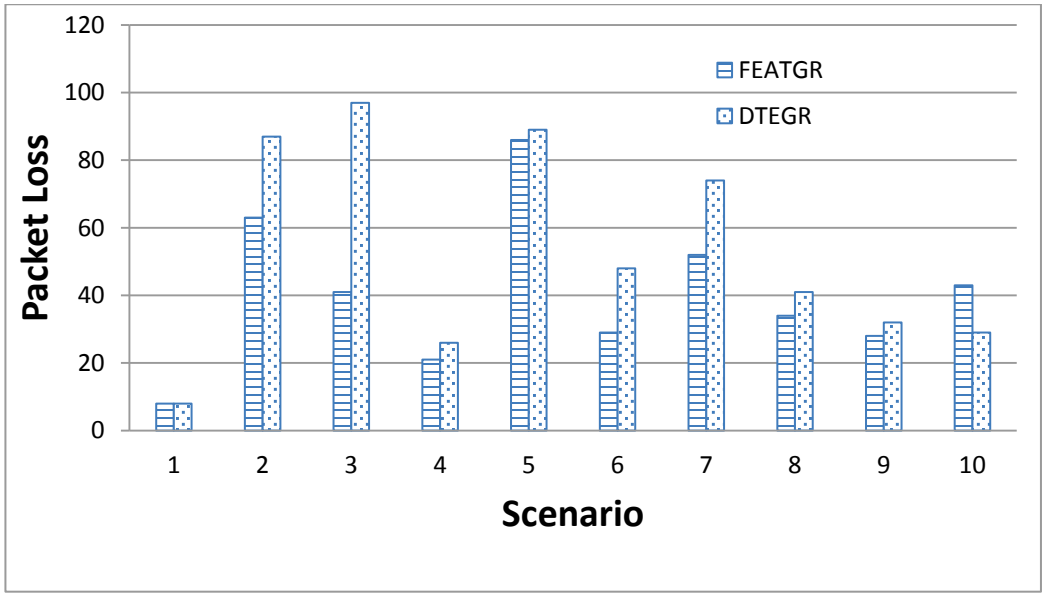

Figure 4.26 - FEATGR vs. DTEGR under 30% attack level

Table 4.23 - DTEGR vs. FEATGR in packet latency under 30% attack

30% attacks	*Network Scenario*									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
FEATGR packet latency (ms)	11.05	12.3	11.04	11.01	12.93	12.27	10.99	12.79	11.01	11.45
DTEGR packet latency (ms)	10.92	11.43	12.19	10.94	11.86	11.08	10.96	12.5	10.9	10.93

As shown in Figure 4.26, the FEATGR algorithm achieved less packet loss than DTEGR in most of scenarios. In scenario 3, The DTEGR's packet loss is in double of FEATGR's packet loss. This is because the energy metric in the DTEGR algorithm affects the algorithm to achieve energy load balance, which makes the DTEGR algorithm select the alternative path with more opportunity to encounter malicious

nodes to cause packet loss. Although the FEATGR can also achieve energy load balance by its energy metric, it will wait for the battery level drop below the threshold then switch the route, rather than like DTEGR is very sensitive to the battery level. The energy metric in DTEGR can result in switching route very frequently comparing with the FEATGR algorithm. In the table 4.23, it has shown the FEATGR result in slightly higher latency compare to DTEGR algorithm. It means the distance metric in DTEGR algorithm has higher priority, so as to find the shortest path to the destination.

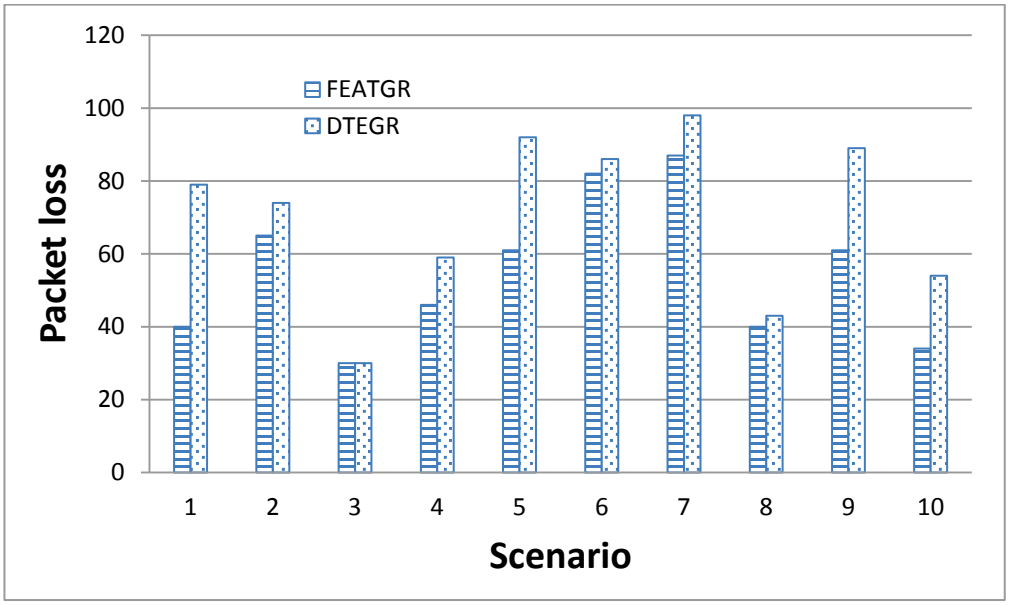

Figure 4.27 - FEATGR vs. DTEGR under 40% attack level

Table 4.24 - DTEGR vs. FEATGR in packet latency under 40% attack

40% attacks	*Network Scenario*									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
FEATGR packet latency (ms)	11.05	12.3	11.41	11.01	12.93	12.27	10.99	12.79	11.01	11.45
DTEGR packet latency (ms)	10.92	11.43	11.41	10.94	11.86	11.08	10.96	12.5	10.9	10.93

Next, we increase the malicious attacks level from 30% to 40% of the network nodes, and the rest of the setups were same as last scenario. The simulation results are shown in above in figure 4.27 and table 4.24. In the figure 4.27, it can be seen that FEATGR's performance in packet loss in ten scenarios achieved all better results than DTEGR algorithm. For DTEGR algorithm, just like under 30% attack level scenarios, its energy metric enforces the energy load balancing which make the selected path have more chances to be attacked by malicious nodes.

Moreover, we increase the attack level again from 40% to 50% to see how this affects FEATGR and DTEGR algorithms' performance under such heavy attack. We kept the rest of network setting same as last scenario. The simulation results are shown in below.

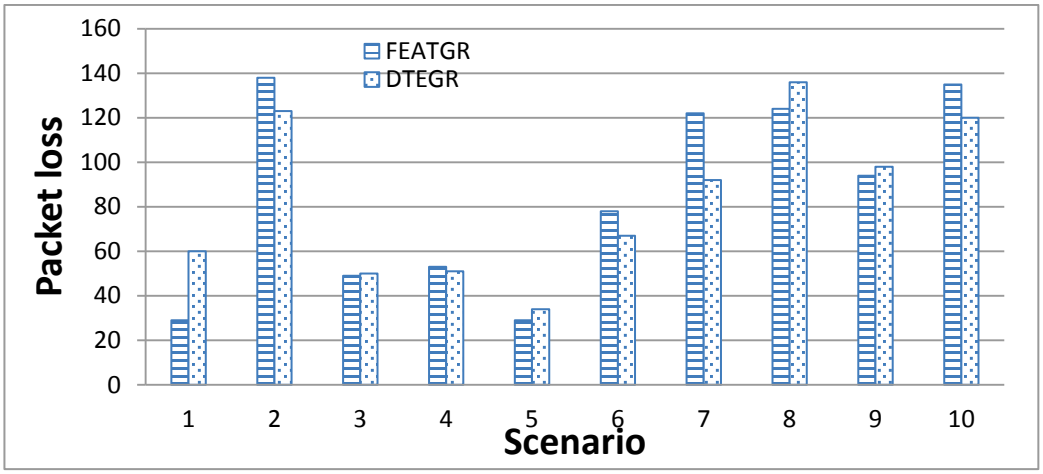

Figure 4.28 - FEATGR vs. DTEGR under 50% attack level

Table 4.25 - DTEGR vs. FEATGR in packet latency under 50% attack

50% attacks	*Network Scenario*									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
FEATGR packet latency (ms)	12.91	12.84	12.84	11.87	11.03	12.69	12.74	13.78	12.64	13.37
DTEGR packet latency (ms)	12.58	12.4	12.6	11.14	11.06	11.05	12.5	12.6	12.51	12.44

Upon 50% of the network heavy malicious attack, FEATGR and DTEGR's performance results were similar, but DTEGR was performing slightly better than FEATGR algorithm in overall in packet loss from the results in figure 4.28. There are similar results between these two algorithms' performance except scenario 1 and 7. In scenario 7, FEATGR selected a different route from DTEGR's to the destination where this route encountered more malicious nodes to cause more packet loss compare to what DTEGR achieved. As there are two nodes have the same distance to the destination with similar trust metric value, FEATGR algorithm will obtain the exact trust metric value and combine with other metrics such distance into an exact final score. Obviously the one with highest final score is select as next hop. But for the DTEGR algorithm, it uses the trust threshold value to generate the safe forwarding list, and other metrics such as distance will select it from the list only. This allows the nodes in the safe forwarding have the same distance to the destination have the same score. In such case, DTEGR

will select the node which first comes in the algorithm as next hop rather than the one with higher trust metric value. This makes FEATGR and DTEGR algorithm select different route to the destination. As DTEGR keep switching the route to achieve energy load balance as always, the positions of the malicious nodes in scenario 1 that cause this behaviour of the algorithm encountered more malicious nodes than other scenarios to cause additional packet loss. Moreover, from the table 4.25, it showed slightly higher packet latency in FEATGR algorithm in overall, just like previous scenarios, DTEGR algorithm performed better in finding a shorter path. As explained above, FEATGR is combining the trust with distance metric by the fuzzy logic to determine the next hop, but DTEGR has the trust and distance metric separated which make DTEGR algorithm's distance metric has more priority compare to FEATGR to select a shorter path.

The table 4.26 shows the standard deviation and average of packet loss for all previous simulation scenarios. It can be seen that FEATGR is able to maintain the performance under 30% and 40% of network malicious attacks, both standard deviation and average were low compare to DTEGR algorithm. But when under 50% of network heavy malicious attack, FEATGR has similar performance with DTEGR, but DTEGR was performing slightly better. This is because DTEGR algorithm is switching the route very often to achieve the energy load balance. In such case, when under light or medium level of malicious attack, it will have more chance to encounter the malicious nodes compare to FEATGR's performance behaviour as DTEGR algorithm try more nodes in the network to find more alternative routes, so as to achieve energy load balance. But when it under the heavy level which was 50% of the network for malicious attack, no matter how the algorithm select the route or switch between alternative routes, the chances of encounter malicious nodes by FEATGR and DTEGR were equalled by the amount of malicious nodes in the network. In overall, upon these three different levels of malicious attack, FEATGR algorithm is performing better.

Table 4.26 - FEATGR vs. DTEGR on packet loss

FEATGR/DTEGR	***Attack level***		
	30%	**40%**	**50%**
DTEGR standard deviation	31.21	27.21	35.56
FEATGR standard deviation	22.32	19.78	43.46
DTEGR average	53.1	68.4	83.1
FEATGR average	40.5	54.6	85

4.5.4 FEATGR vs. DTEGR on energy metric and scalability

In this section, we will first study the energy consumption of FEATGR algorithm and compare its performance to DTEGR algorithm. The first scenario has 100 nodes were deployed in 10x10 grid without any malicious attack. In this scenario, there are 300 traffic packets are transmitting between node 1 to node 99, so as to have all the nodes in the network had enough batteries to the end of the simulation. We test the batteries in the simulation, the nodes start exhaust their batteries from 500 seconds. Each UDP packet size is 31 bytes and the generating interval is per second. While the beacon messages exchange interval is 0.5 seconds for keeping contact with neighbour nodes and also signalling. In addition, we assume that each node has total 1000 units of energy restored, where the transmitting consumes 50 units per second and receiving consumes 30 units per second.

Table 4.27 - FEATGR vs. DTEGR on energy consumption

Nodes	*1*	*22*	*34*	*55*	*67*	*88*	*99*
DTEGR (per sec) Beacon exchange	0.43	0.43	0.96	0.96	0.59	0.59	0.19
FEATGR (per sec) Beacon exchange	0.43	0.43	0.96	0.96	0.58	0.58	0.2
DTEGR (per sec) Packets & Beacon	0.79	0.79	1.49	1.5	0.97	1.03	0.39
FEATGR (per sec) Packets & Beacon	0.77	0.77	1.48	1.49	0.97	0.98	0.38

As shown in table 4.27, the FEATGR has similar energy consumption as the DTEGR algorithm. There were slightly different in node 1, 22, 67, and 88. On node 1, 22, and 88, the energy consumption by FEATGR algorithm was slightly less than DTEGR's consumption which is almost no different between. In general speaking, the energy consumption between these two algorithms is the same.

As previous sections for the evaluation on the TIGER and DTEGR energy consumption, we use node 55's battery life time as a benchmark. Here the network under FEATGR algorithm node 55's battery life time is 684.16 seconds. In TIGER and DTEGR algorithm, when the energy metric was turned off, the live time for node 55 was 671.59 seconds. In the simulation with FEATGR algorithm, the energy metric make the algorithm select node 55 as one of the forwarding hops for only 261 seconds rather than 909 seconds which were the complete simulation time. The energy consumption different between forwarding packets and stand-by was about 0.5 energy unit per second according to table 4.27. In such case, theoretically the battery life time for node 55 should around 898 seconds rather than only 684.16 seconds after the energy load balance was activated. The actual life time for node 55 only last 684.16 seconds

was because the indirect trust mechanism in the FEATGR algorithm consumed additional energy while node 55 on stand-by status. While FEATGR algorithm was avoiding node 55 to achieve energy load balance, the algorithm was still asking node 55 for indirect trust value for its neighbours. In such case, it consumed additional energy units per second for receiving request and replying back with indirect trust value on node 55. This confirms the FEATGR does extend the batteries life of the nodes' in the network. The TIGER algorithm want to achieve such result will be very difficult base on the results it achieved in table 4.9. DTEGR was able to achieve it or even better by setting up the energy weight factor to 0.4, but at the same time, this will sacrifice the distance metric performance. In FEATGR algorithm, it can emphasis energy metric by alternating the fuzzy rules table, and if there are any other new metric comes in, it only requires to alternate the fuzzy rules table rather than like the discrete model which using the weight factors to share another proportion of the algorithm performance, namely scalable.

There are 900 traffic sessions will be generated, and the session interval is 1 second. The malicious attack pattern is as figure 4.3 shown in the previous section in this chapter. The grey-hole attack will start after 30 packets forwarded by the malicious nodes, then stop at 60th packet, and start again at 100th packets and stop at 130th packets, and finally start again at 200 packets. The results have come back as below in table 4.28. The FEATGR had 172 packets lost in the simulation and with 14.07 milliseconds mean packet latency, where DTEGR algorithm had 218 packet lost and 15.93 milliseconds in mean packet latency.

Table 4.28 - FEATGR vs. DTEGR on packet loss

FEATGR/DTEGR	***Algorithms***	
	FEATGR	**DTEGR**
Packet Loss	172	218
Means Packet Latency (ms)	14.07	15.93

It confirmed that FEATGR algorithm able to maintain the performance which is to find the secure and shorter path to the destination compare to DTEGR algorithm. While FEATGR algorithm was detecting and avoiding the malicious nodes, it also detecting and avoiding the low batteries nodes in the simulation with no sacrifice of distance metric performance, this reflect on the shorter packet latency in the table 4.28. The DTEGR algorithm start to loss the direction to the destination as huge amount of the packet loss were occurred at the end of the simulation while nodes in the centre of

network start black out, this is caused by energy weight factor was too high to make the algorithm ignore the distance metric. Namely, there is a trade-off between energy and distance metric in DTEGR algorithm, while FEATGR do not have such problem which confirm that FEATGR algorithm is more flexible and scalable compare to DTEGR algorithm.

4.5.5 Summary

In this section, we have extensive simulation studies for the FEATGR algorithm, and these studies have confirmed FEATGR algorithm can adjust itself to different network scenarios to maintain the performance. It was able to achieve better performance than DTEGR algorithm under light and medium level of malicious attack scenarios. Moreover, FEATGR is able to achieve similar performance with DTEGR in packet loss under heavy level of malicious attacks. At the end of this section, we also have simulation studies on energy consumption in FEATGR algorithm, and it indicated the energy metric in FEATGR algorithm is able to achieve energy load balance as it is able to achieve the extension of node battery life time. Moreover, we test the energy metric under malicious attacks and compare with DTEGR algorithm, it is indicating that FEATGR is able to achieve energy load balance with minimum sacrifice of other metrics' performance. This is because FEATGR is able to maintain the performance level in the network with huge amount of low battery nodes using fuzzy logic approach, while DTEGR in such environment had lost the direction to the destination to cause large amount packet loss. This is because DTEGR algorithm has to sacrifice distance metric performance for energy metric to achieve energy load balancing.

4.6 Summary

The TIGER algorithm uses the logic of ATSR algorithm which uses trust to detect and avoid the malicious node, and distance metric to find the direction to the destination, and finally use weight factor combines these metric together. Most importantly, TIGER algorithm proposed the timestamp mechanisms to maintain the efficiency of detecting malicious nodes over time. DTEGR algorithm has use trust to evaluate neighbours, and use the trust threshold to determine the malicious nodes, so as to generate the safe

forwarding list. Finally, it uses the distance metric to select the next forwarding hop from the safe forwarding list. In such way, DTEGR resolve the weight factor selection issue between trust and distance metrics in ATSR algorithm. But as the discrete model based trust-based routing protocols, ATSR, TIGER, and DTEGR algorithms are using the linear functions like weight factor to combine all different metrics together and have to sacrifice other metrics to improve particular metric's performance. To advance, we proposed the FEATGR algorithm uses the fuzzy logic approach to combine different metrics together into a final score with fuzzy rules table and defuzzification mechanism to evaluate the neighbours. The advantage of this fuzzy approach is to emulate the process of human decision making so as to make the algorithm or system more intelligent and scalable.

In this chapter, we have first run through different network scenarios with TIGER algorithms and ATSR to confirm TIGER algorithm able to resolve the time-insensitive problem on historical records of ATSR by using timestamp mechanisms. Moreover, we found that the AW timestamp mechanism is performing better when encounter focus attack such as black-hole attack, and CW is performing better when encounter random attack such as grey-hole attack. Namely TIGER algorithm is able to maintain the efficiency of detecting malicious nodes over time. Then next we have verified the DTEGR algorithm able to maintain the efficiency of detecting malicious nodes in different network scenarios, while ATSR requires different trust weight factor to achieve this. In other words, DTEGR algorithm resolved the weight factors selection issue between trust and distance metrics in ATSR algorithm. Finally, we compare the FEATGR algorithm with TIGER and DTEGR algorithm through different network scenarios as well. We confirmed the FEATGR algorithm able to maintain the performance in different network scenarios and achieve better performance in packet loss in overall compare to DTEGR algorithm. In additional, through the energy metric study on FEATGR and DTEGR, it confirmed the FEATGR algorithm is more flexible and scalable.

Chapter 5

Conclusion and Future work

The purpose of this thesis was to provide an in-depth understanding of the fundamentals of the network availability problem in wireless mesh network for Smart Grid environment. We were striving to contribute new knowledge and solutions secure routing in Smart Grid communication networks by using trust-based geographical routing approaches to detect and avoid various malicious attacks.

5.1 Contributions

Inspired from the existing trust-based algorithm Ambient Trust Sensor Routing (ATSR) algorithm, we have identified its weaknesses, and try to tackle these problems by proposing three new trust-based geographical routing algorithms.

The first algorithm we have proposed is Trust-base Intelligent Geo Elective Routing (TIGER) which overcome the time-insensitive problem on historical records in ATSR by introducing timestamp mechanism. There are two different timestamp mechanisms introduced, which are attenuation window and constant window. The attenuation window (AW) is to emphasize the most recent experiences and fade away the out of date experiences. The constant window (CW) is only considering the certain amount of up to date experiences in the trust evaluation. In both cases, the malicious nodes can hardly accumulate a large amount of good experience so as to cheat the algorithm to ignore their malicious behaviours later on. The extensive simulation studies have confirmed that TIGER algorithm is able to maintain its sensitivity on detecting and avoiding the malicious attacks at any time include short term and long term, which is better than ATSR. Moreover, it also shows that TIGER_CW has better and more stable performance than TIGER_AW when the network was under random malicious attack scenarios such as grey-hole attacks. On the other hand, the TIGER_AW has better performance than TIGER_CW when the network was under targeted attacks such as black-hole attacks.

The second contribution we have proposed is Dynamic Trust Elective Geo Routing (DTEGR) which is to resolve the inflexible weight factor selection problem in ATSR and also TIGER algorithm. It splits the routing selection into two steps. The first step is trust evaluation and filtering out the un-trustable neighbours. The second step is to select the next hop based on the geographical information. The algorithm first evaluates the neighbours and decides whether they are legitimate or malicious nodes by setting a trust threshold value. Then a safe forwarding list can be generated with only legitimate nodes listed. In the second step, the distance metric will be used to select the next forwarding hop from this safe forwarding list only. The extensive simulation studies have proven the DTEGR algorithm is able to adopt itself to different network scenarios so as to maintain good performance without adjusting any weight factors like ATSR and TIGER algorithm did. Moreover, we run the simulation studies to study the energy metric in DTEGR algorithm and find that the energy metric in DTEGR algorithm is more effective comparing with ATSR and TIGER.

For the third contribution, we have proposed Fuzzy-based Energy Aware Trust Geo Routing (FEATGR) algorithm. Although DTEGR resolved the inflexible weight factor selection problem, but it is still using weight factors to synthesize the direct trust and indirect trust, energy metric and distance metric. It means that it still has weight factors selection problem between these four metrics. While the FEATGR algorithm uses the fuzzy logic mechanism to combine all these metrics using fuzzy rules table and central of gravity defuzzification method. In such way, the inflexible weight factors selection problem can be resolved. Through the extensive simulation studies, we confirmed that FEATGR is able to maintain its performance level upon different network scenarios. For the energy consumption aspect, FEATGR has been proven to have more stable performance even under the network condition with lots of low battery nodes, while DTEGR already lost the direction to the destination node.

5.2 Future work

Considering the work covered in this thesis within a constraint in the limited time period and the development of the future network, it would be useful to highlight some future areas to be further investigated.

The traffic demand scenarios in this thesis are only assumed as single traffic session, while the multi traffic demands scenarios should be used and validated in the future work.

The distance metric used in the proposed routing algorithms is calculated using GPSR approach, but it only inherits the first approach of greedy forwarding. In such case, the routing loop problem needs to be addressed in the future work.

The direct trust and indirect trust metrics are synthesized using weight factor or the fuzzy logic approach in our current algorithms. A better way to select the optimal weight factor between direct trust and indirect trust metrics or optimal configuration on direct trust and indirect metrics in the fuzzy rules table are also need to be further investigated. Moreover, an efficient solution to against the bad mouthing attacks is necessary as this attack can make the indirect trust mechanism become inefficient in the trust management system.

Reference

[1] T. Baumeister, “Literature Review on Smart Grid Cyber Security,” Tech Report, http://csdl.ics.hawaii.edu/techreports/10-11/10-11.pdf, 2010.

[2] Y. Yan, Y. Qian, and D. Tipper, “A Survey on Cyber Security for Smart Grid Communications,” IEEE Communications Surveys & Tutorials, vol. pp, issue 99, 2012, pp. 1-13.

[3] Y. Yan, Y. Qian, H. Sharif, and D. Tipper, “A Survey on Smart Grid Communication Infrastructures: Motivations, Requirements and Challenges,” IEEE Communications Surveys & Tutorials, vol. pp, issue 99, 2012, pp. 1-16.

[4] Y. Zhang, W. Sun, L. Wang, H. Wang, R.C. Green, and M. lam, “A Multi-Level Communication Architecture of Smart Grid Based on Congestion Aware Wireless Mesh Network,” North American Power Symposium (NAPS), 2011, pp. 1-6.

[5] Z. Lu, X. Lu, and W. Wang, “Review and Evaluation of Security Threats on the Communication Networks in the Smart Grid,” Military Communications Conference, 2010, pp. 1830-1835.

[6] Smart Grid Conceptual Model, http://smartgrid.ieee.org/ieee-smart-grid/smart-grid-conceptual-model.

[7] D. Geelen, G.V. Kempen, F.V. Hoogstraten, and A. Liotta, “A wireless mesh communication protocol for smart-metering,” Computing, Networking and Communications (ICNC) International Conference, 2012, pp. 343-349.

[8] J. Jung, K. Lim, J. Kim, Y. ko, Y. Kim, and S. Lee, “Improving IEEE 802.11s Wireless Mesh Networks for Reliable Routing in the Smart Grid Infrastructure,” Communications Workshops (ICC), 2011 IEEE International Conference, 2011, pp. 1-5.

[9] A.P. Athreya and P. Tague, “Survivable smart grid communication: Smart-meters meshes to the rescue,” Computing, Networking and Communications (ICNC) International Conference, 2012, pp. 104-110.

[10] H. Gharavi, and B. Hu, “Multigate mesh routing for Smart Grid last mile communications,” Wireless Communications and networking Conference (WCNC), 2011, pp. 275-280.

[11] T. Zahariadis, H.C. Leligou, S. Voliotis, S. Maniatis, P. Trakadas, and P. Karkazis, “An Energy and Trust-aware Routing Protocol for Large Wireless Snesor Networks,” Proceedings of the 9th WSWAS International Conference on Applied Informatics and Communications (AIC ‘09), 2009, pp. 216-226.K. Elissa, “Title of paper if known,” unpublished.

[12] R. Yu, Y. Zhang, S. Gjessing, C. Yuen, S. Xie, and M. Guizani, “Congitive radio based hierachical communications infrastructure for smart grid,” IEEE network, vol. 25, no. 5, pp. 6-14, September-October 2011.

[13] G. Vasilakis, G. Perantinos, I. Askoxylakis, N. Mechin, V. Spitadakis, A. Traganitis, “Business opportunities and considerations on wireless mesh networks,” IEEE WoWMoM, 2009.

[14] I.F. Akyildiz, and X. Wang, “A survey on wireless mesh networks”, IEEE Communication Magzine, vol. 43, issue 9, pp. S23-S30, Sep 2005.

[15] Report to NIST on Smart Grid Interoperability Standards Roadmap EPRI, Jun. 17, 2009 [Online]. Available: http://www.nist.gov/smartgrid/InterimSmartGridRoadmapNISTRestructure.pdf.

[16] B. Kannhavong, H. Nakayama, Y. Nemoto, N. Kato, A. Jamalipour, “A survey of routing attacks in mobile ad hoc networks,” Wireless communications, IEEE, vol. 14, issue: 5, pp. 85-91, October 2007.

[17] C. Karlof, and D. Wagner, “Secure routing in wireless sensor networks: attacks in moible ad hoc networks,” IEEE International Workshop on Sensor Network Protocols and Applications, pp. 113-127, May 2003.

[18] TC. Chiang, and YM. Huang, “Group keys and mulitcast security in ad hoc networks”, Parallel Processing Workshops (ICPPW), 2003, pp. 385.

[19] V. C. Giruka, M. Singhal, J. Royalty, S. Varanasi, “Security in wireless sensor networks”, Wireless Communications & mobile Computing Vol. 8, 2008, pp.1-24.

[20] CP. Pfleeger, and SL. Pfleeger, “Security in Comupting”, 3rd Ed. Prentice Hall, 2003, pp. 35.

[21] National Bureau of Standards, Data Encryption Standard, FIPS-Pub.46. National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

[22] National Institute of Standards and Technology, "Announcing the Advanced Encryption Standard (AES)," in Federal Information Processing Standards Publication, Nov. 26, 2001.

[23] R. L. RIVEST, A. SHAMIR, and L. ADLEMAN, A method for obtaining digital signatures and public key cryptosystems, Commun. ACM 21, pp. 120-126, 1978.

[24] Y. Sun, W. Trappe, and K. J. R. Liu, "A scalable multicast key management scheme for heterogeneous wireless networks", IEEE/ACM Transactions on Networking, vol. 12, issue 4, pp. 653-666, Aug 2004.

[25] Chokhani, S., Ford, W. "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework". RFC 2527. March 1999.

[26] D. Chadwick, "Role-based access control with X.509 attribute certificates", Internet Computing, IEEE, vol. 7, issue 2, pp. 62-69, Mar/Apr 2003.

[27] B. Rehax, "Increasing user privacy in online transactions with X.509 v3 certificate private extensions and smartcards", Seventh IEEE International Conference on E-Commerce Technology, pp. 293-300, Jul 2005.

[28] Z. Zhao, "NeoMAN: Negotiation Management Method for IKE Protocol Based on X.509", Advanced Language Processing and Web Information Technology `08 International Conference, pp. 335-340, Jul 2008.

[29] A.S. Wazan, "Validating X.509 Certificates Based on their Quality", the 9th International Conference for Young Computer Scientists, pp. 2055-2060, Nov 2008.

[30] V. Dehalwar, "Multi-agent based public key infrastructure for smart grid", 7th International Conference on Computer Science and Education, pp. 415-418, Jul 2012.

[31] J. Xia, and Y. Wang, "Secure Key Distribution for the Smart Grid", IEEE Transactions on Smart Grid, vol. 3, issue 3, pp. 1437 -1443, Sep 2012.

[32] T. Baumeister, "Adapting PKI for the smart grid", IEEE Internation Conference on Smart Grid Communications, pp. 249-254, Oct 2011.

[33] F. Zhao, Y. Hanatani, Y. Komano, B. Smyth, S. Ito, T. Kambayashi, "Secure authenticated key exchange with revocation for smart grid", Innovative Smart Grid Technologies, pp. 1-8, Jan 2012.

[34] Q. Gao, "Biometric authentication in Smart Grid", Energy and Sustainability Internationl Conference, pp. 1-5, Mar 2012.

[35] H. Khurana, R. Bobba, T. Yardley, P. Agarwal, and E. Heine, "Design Principles for Power Grid Cyber-Infrastructure Authentication Protocols", in Hawaii International Conference on System Sciences, 2010. http://water.iti.illinois.edu/papers/IllinoisGridAuthenticationPrinciples.pdf

[36] Perrig A, Szewczyk R, Wen V, Culler D, "Tygar JD. SPINS: security protocols for sensor networks",Wireless Networks 2002, **pp.** 521–534.

[37] Perrig A, Canetti R, Tygar JD, Song D. "The tesla broadcast authentication protocol", RSA CryptoBytes 2002, pp. 2–13.

[38] Q. Li, "Multicast Authentication in the Smart Grid With One-Time Signature", IEEE Transactions on Smart Grid, vol. 2, issue 4, pp. 686-696, Dec 2011.

[39] E. Ayday, "Secure, intuitive and low-cost device authentication for Smart Grid networks", IEEE Consumer Communications and Networking Conference, pp. 1161-1165, Jan 2011.

[40] Q. Bai, and Y. Zheng, "Study on the access control model", Cross Strait Quad-Regional Radio Science and Wireless Technology, vol. 1, pp. 840-834, Jul 2011.

[41] H. Cheung, A. Hamlyn, T. Mander, C. Yang, R. Cheung, "Role-based model security access control for smart power-grids computer networks", Power and Energy Society General Meeting – Conversion and Delivery of Electrical Eenergy in 21st Century, pp. 1-7, Jul 2008.

[42] D. Bowen, "Tech View: Laying the Groundwork for the Coming Smart Grid", [Online], available: http://www.research.att.com/articles/featured_stories/2010_01/201002_techview_smartgrid.html?fbid=2v8TG_br_6U, Feb 2010.

[43] "Open, Standard Smart Grid Communication Network", [Online], available: http://www.trilliantinc.com/mobile-home.

[44] H. Frey, "Scalable Geographic Routing Algorithms for Wireless Ad Hoc Networks," IEEE Network Magazine, pp. 18–22, July/Aug. 2004.

[45] B. Karp and H.T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," MobiCom '00 Proceedings of the 6th annual international conference on Mobile computing and networking, 2000, pp.243-254.

[46] G.T. Toussaint, "The relative neighbourhood graph of a finite planar set", Pattern Recongnition, vol. 12, issue 4, 1980, pp. 261-268.

[47] K.R. Gabriel, and R.R Sokal, "A New Statistical Approach to Geographic Variation Analysis", Systematic Zoology, vol. 18, issue 3, 1969, pp. 259-270.

[48] DB. Johson, and DA. Maltz, "dynamic source routing in ad hoc wireless network", Mobiling Computing, 1996, pp. 153-181.

[49] C.E. Perkins, and E.M. Royer, "Ad-hoc on-demand distance vector routing", Proceedings 2nd IEEE Workshop on Mobile Computing Systems and Applications, pp. 90-100, Feb 1999.

[50] C. Perkins, and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", SIGCOMM '94 : Computer Communications Review, vol. 24, issue: 4, pp. 234, Oct 1994.

[51] H. Redwan, and K.H. Kim, "Survey of Security Requirements, Attacks and Network Integration in Wireless Mesh Networks", Japan-China Joint Workshop on Frontier of Computer Science and Technology, pp. 3-9, Dec 2008.

[52] T. Baumeister, "Literature review on Smart Grid Cyber Security", Technical Report, http://www.public.asu.edu/~xfang5/survey_smartgrid_2011.pdf, 2010.

[53] L. Zhou, Z.J. Haas, "Securing ad hoc networks", IEEE Network, vol. 13, issue: 6, pp. 24-30, Nov/Dec 1999.

[54] S. Marti, TJ. Giuli, K. Lai, M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks", Proceedings of International Conference on Mobile, 2000.

[55] A.S. Hornby, "Oxford Advanced Learner's Dictionary of Current English", Oxford University Press, Oxford, UK, 1988.

[56] R. Hardin, "The Street-Level Epistemology of Trust", Politics and Society, vol. 21, issue 4., 1993, pp. 505-529.

[57] J.B. Rotter, “Interpersonal Trustworthiness, and Gullibility”, American Psychologist, vol. 35, issue 1, pp. 1-7, Jan1980.

[58] F.D. Schoorman, R.C. Mayer, J.H. Davis, “An Integrative Model of Organizational Trust: Past, Present, and Future”, Academy of Management Review, vol. 31, issue 2, 2007, pp. 344-354.

[59] S. Marsh, “Formalising Trust as a Computational Concept”, PhD Thesis, University of Stirling, UK, 1994

[60] D. Gambetta, “Can we Trust Trust?”, Trust: Making and Breaking Cooperative Relations, Basil Blackwell, Oxford, 1990, pp. 213-237.

[61] W.J. Adams, and N.J. Davis, “Towards a Descentralized Trust-Based Access Control System for Dynamic Collaborations”, Proceedings 6th Annual IEEE SMC Information Assurance Workshop, pp. 317-324, Jun 2005.

[62] R.C. Mayer, J.H. Davis, and F.D. Schoorman, “An Integrative Model of Organizational Trust,” Proceedings of the 6th annual ACM/IEEE International Conference on Mobile Computing and Networking, vol. 20, no. 3, 1995, pp. 709-734.

[63] A. Josang, “The Right Type of Trust for Distributed Systems,” Proceedings of the ACM New Security Paradigms Workshop, 1996, pp. 119-131.

[64] D. Denning, “A new Paradigm for Trust Systems,” Proceedings of the ACM New Security Paradigms Workshop, 1993, pp. 36-41.

[65] Y. Wenzhong, H. Cuanhe, W. Bo, W. Tong, Z. Zhenyu, "A General Trust Model Based on Trust Algebra," International Conference in Multimedia Information Networking and Security, vol. 1, pp.125-129, Nov 2009.

[66] M.C. Fernandez-Gago, R. Roman, J. Lopez, "A Survey on the Applicability of Trust Management Systems for Wireless Sensor Networks," Third International Workshop in Security, Privacy and Trust in Pervasive and Ubiquitous Computing, pp.25-30, July 2007.

[67] G. Yajun, and W. Yulin, "Establishing Trust Relationship in Mobile Ad-Hoc Network," International Conference on Wireless Communications, Networking and Mobile Computing, pp.1562-1564, Sep 2007.

[68] B. Mu, and S. Yuan, "A method for evaluating initial trust value of direct trust and recommender trust," International Conference on Computer Design and Applications (ICCDA), vol.2, pp. V2-185-V2-190, Jun 2010.

[69] A. Ukil, "Secure Trust Management in Distributed Computing Systems," Electronic Design, Test and Application (DELTA), 2011.

[70] J. Cho, A. Swami, I. Chen, "A Survey on Trust Management for Mobile Ad Hoc Networks," Communications Surveys & Tutorials, IEEE, vol. 13, issue 4, 2011, pp. 562-583.

[71] J.M. Gonzalez, M. Anwar, J.B.D. Joshi, "Trust-based Approach to Solve Routing Issues in Ad-Hoc Wireless Networks: A Survey", IEEE 10th International Conference on Security and Privacy in Computing and Communications, pp. 556-563, Nov 2011.

[72] D. Gambetta, "Trust: Making and Breaking Cooperative Relations", electronic edition, Department of Sociology, University of Oxford, 2000, available: http://www.sociology.ox.ac.uk/papers/trustbook.html.

[73] L. Eschenauer, V. D. Gligor, and J. Baras, "On Trust Establishment in Mobile Ad Hoc Networks", Proc. 10th Int'l Security Protocols Workshop, Cambridge, U.K., vol. 2845, pp. 47-66, Apr 2002.

[74] M. Brinklov, R. Sharp, "Incremental Trust in Grid Computing," Cluster Computing and the Grid, pp. 135-144, May 2007.

[75] P. Lamsal, "Understanding trust and security", Department of Computer Science Technical Report, University of Helsiki, Finland, 2001, available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.7843&rep=rep1&type=pdf.

[76] E. Ayday, and F. Fekri, "Iterative Trust and Reputation Management Using Belief Propagation", IEEE Transactions on Dependable and Secure Computing, vol. 9, issue 3, pp. 375-386, May-Jun 2012.

[77] G. F. Marias, V. Tsetsos, O. Sekkas, P. Georgiadis, "Performance evaluation of a self- evolving trust building framework," Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks, pp. 132-141, 5-9 Sept 2005.

[78] S. Tanachaiwiwat, P. Dave, R. Bhindwale, and A. Helmy, "Location-centric isolation of misbehavior and trust routing in energy-constrained sensor networks," Performance , Computing, and Communications, IEEE International Conference, 2004, pp. 463-469.

[79] A. Rezgui and M. Eltoweissy "TARP: A Trust- Aware Routing Protocol for Sensor-Actuator Networks" IEEE International Conference on Mobile Ad hoc and Sensor Systems, MASS 2007, Pisa, Italy, October 2007.

[80] D. Ramot, M. Friedman, G. Langholz, A. Kandel, "Complex fuzzy logic", IEEE Transactions on Fuzzy Systems, vol. 11, issue 4, pp. 450-461, Aug 2003.

[81] P. Khatri, S. Tapaswi, U.P. Verma, "Fuzzy based trust management for wireless ad hoc networks", International Conference on Computer and Communication Technology (ICCCT), pp. 168-171, Sep 2010.

[82] J.M.L. Manickam, and S. Shanmugavel, "Fuzzy Based Trusted Ad hoc On-demand Distance Vector Routing Protocol for MANET", International Conference on Advanced Computing and Communications, pp. 414-421, Dec 2007.

[83] H. Dai, Z. Jia, Z. Qin, "Trust Evaluation and Dynamic Routing Decision Based on Fuzzy Theory for MANETs", Journal of Software, vol. 4, issue: 10, pp. 1091, Dec 2009.

[84] G. Ghalavand, A. Dana, A. Ghalavand, M. Rezahosieni, "Reliable routing algorithm based on fuzzy logic for Mobile Ad hoc Network", 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 5, pp. 606-609, Aug 2010.

[85] S. Ganeriwal and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks", Proceedings of ACM Security for Ad-Hoc and Sensor Netw., 2004, pp. 66-67.

[86] J-Sim, https://sites.google.com/site/jsimofficial/.

[87] R. Eberhart, P. Simpson, R. Robbins, "Fuzzy Systems Theory and Paradigms", *Computational Intelligence PC Tools*, United Kingdom ed., UK: Academic Press Ltd., 1996, pp. 250.

Glossary

ACK: Acknowledgement

ACL: Access Control List

AES: Advanced Encryption Standard

AODV: Ad-hoc on Demand Distance Vector

ATSR: Ambient Trust Sensor Routing

AW: Attenuation Window

BAN: Business Area Network

CA: Certification Authorities

CW: Constant Window

DES: Data Encryption Standard

DSDV: Destination-Sequence Distance Vector

DSR: Dynamic Source Routing

DTEGR: Dynamic Trust Elective Geo Routing

FEATGR: Fuzzy-based Energy Aware Trust Geo Routing

GG: Gabriel Graph

GPSR: Greedy Perimeter Stateless Routing

HAN: Home Area Network

IAN: Industry Area Network

ICT: Information and Communication Technology

NAN: Neighbourhood Area Network

NIST: National Institute of Standards and Technology

PKI: Public Key Infrastructure

PLC: Power Line Communication

RNG: Relative Neighbourhood Graph

SCADA: Supervisory Control And Data Acquisition

SSH: Secure Shell

TIGER: Trust-based Intelligent Geo Elective Routing

TTL: Time To Live

VPN: Virtual Private Network

WAN: Wide Area Network

Appendix A: J-Sim tool environment

J-Sim is developed in the Ohio State University and it is an open source, component-based compositional network simulation environment that is developed entirely in Java. The simulator provides many advantages over the "classic" ns-2 approach, namely with the use of the Autonomous Component Architecture (ACA) that enables component independence. The ACA uses a model that resembles integrated circuit assembly, since components have ports to communicate between each other and can be constructed independently. J-Sim uses Java to implement all the cases and Tool Command Language (Tcl) is used as a linking script language that enables construction, configuration and/or network simulation at run-time.

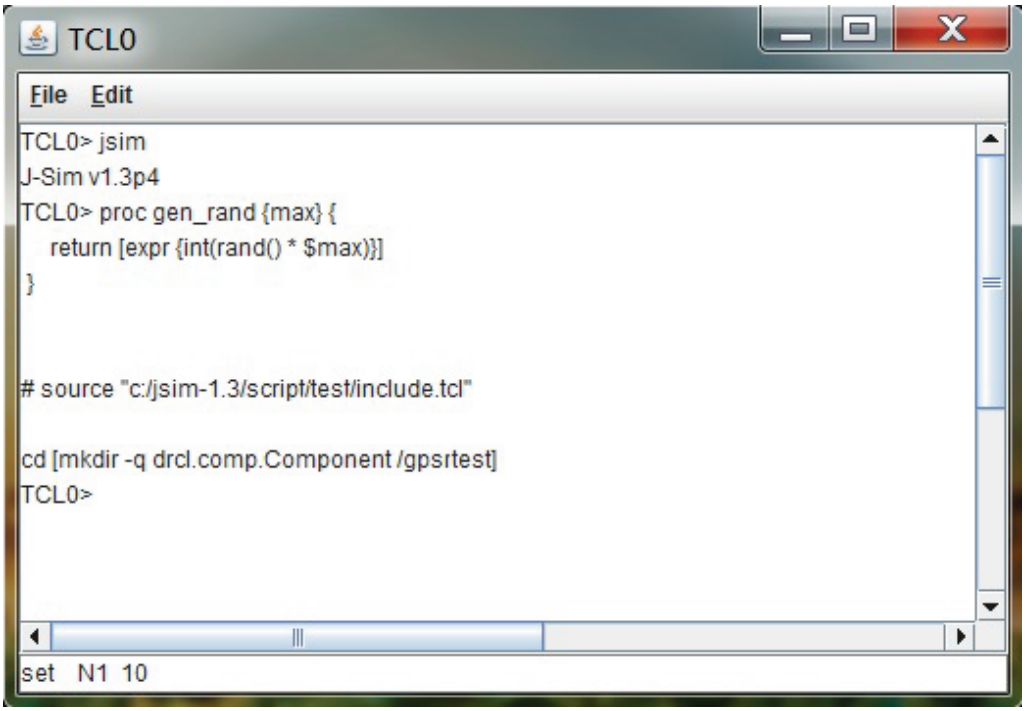

Figure A.1 - J-Sim Command Prompt window

In Figure A.1, it is the screen shot of the command prompt window of J-Sim tool. As J-Sim tool is component-based architecture software, it requires scripts to call and make the components within J-Sim working together for any simulations. TCL is the script language can be used in J-Sim to achieve such jobs.

In the Figure A.2 below, it is the screenshot of the J-Sim window during the simulation. On the left hand-side of the window, it contains the information about data

session direction which it in the screenshot example was from node 1 to node 99. Below the data sessions is the session paths section. It records every traffic session path from the source to the destination. From this, we can find out from the history session path records how the algorithm is selecting the paths, such as the algorithm is trying to achieve energy load balance by switching the routes, etc. Moreover, from this we can also find out which nodes are dropping packets. On the bottom of the window, it also provides the statistic of different metrics, such as how many beacon messages are generated, how many network acknowledgement message received, etc.

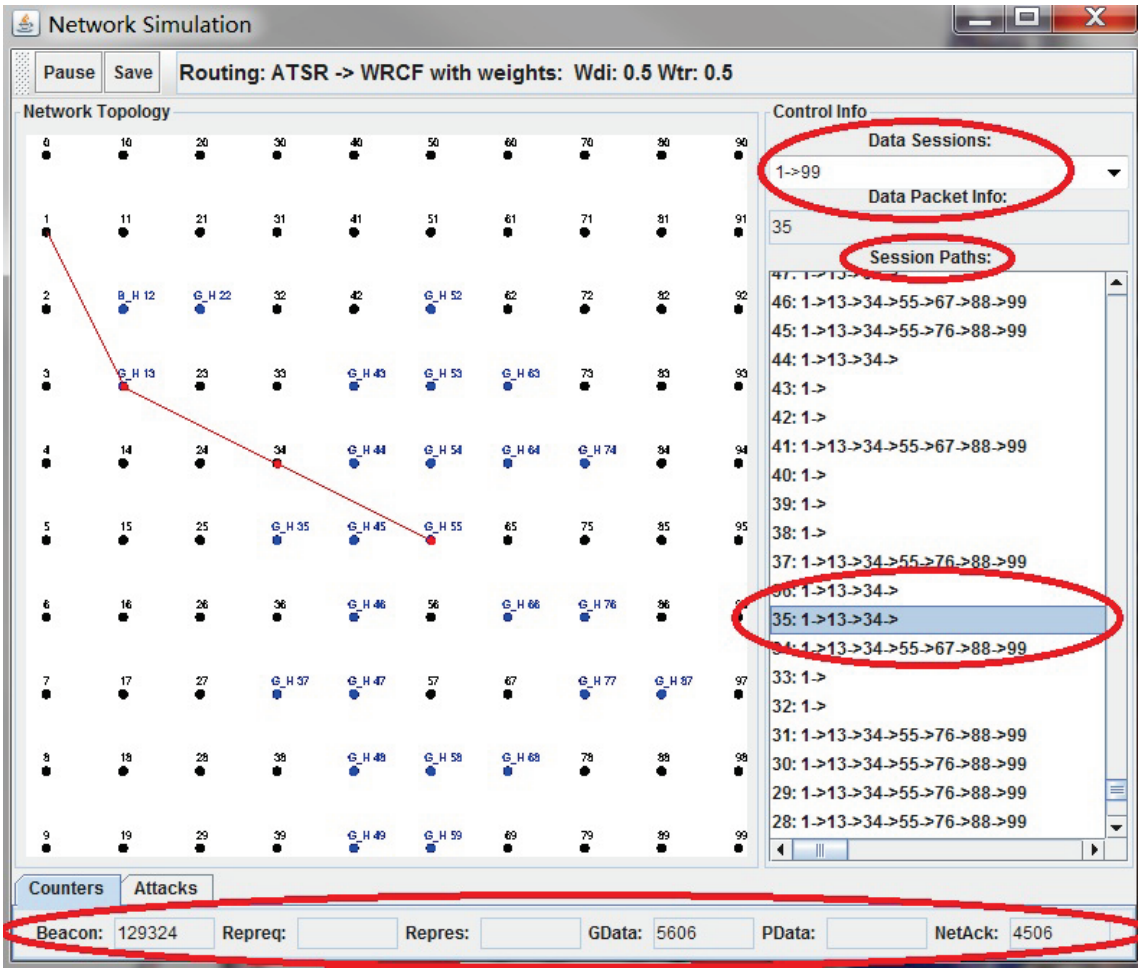

Figure A.2 - Simulation window

Appendix B: Sample codes for proposed algorithms

All the Java scripts for the simulations in this thesis will be deposited in a DVD for final deliverable. Below are some examples of the scripts for the three proposed algorithms in this thesis.

Java scripts for trust metric in TIGER algorithm

Below are the scripts for constant window (CW) timestamp mechanism to collect the trust evaluation records.

```
void set_forward_success(double fwds) {

		if (tsmv.size() < 30) {
			tsmv.add(1);
		} else if (tsmv.size() == 30){
			tsmv.removeElementAt(0);
			tsmv.add(1);
		}

	} // satisfied feedback collection with CW

void set_forward_failure(double fwdf) {

		if (tsmv.size() < 30) {
			tsmv.add(0);
		} else if (tsmv.size() == 30){
			tsmv.removeElementAt(0);
			tsmv.add(0);
		}

	} // dissatisfied feedback collection with CW
```

Below are the scripts for attenuation window (AW) timestamp mechanism and the calculation for the direct trust metric.

```
int totalm = tsmv.size(); // total of evaluation count
int z = 0;

while (z < totalm){ // total success and failure evaluations with timestamp
	if (tsmv.elementAt(z) == 1) {
		double num = totalm - z - 1, num1;
		num1 = java.lang.Math.pow(e, -(num/10));
		c = c + num1;
	}else{
```

```
                double num = totalm - z - 1, num1;
                num1 = java.lang.Math.pow(e, -(num/10));
                d = d + num1;
        }
        z++;
}
forwTrust = c / (c + d); // direct trust metric result
```

Java scripts for DTEGR algorithm

The DTEGR algorithm has two steps to select the next hop neighbour. First it will find out the trust value from the neighbours and compare with the predefine threshold value. The neighbours which have higher trust metric value than the threshold will be selected into the safe forwarding list, then distance metric will select the next hop only from this list. The sample scripts are in the below.

```
for (int i = 0; i < no; i++) {
        GPSR_Nbr nbr = (GPSR_Nbr) gpsr_nbr_list.elementAt(i);
        t = nbr.calc_totalTrust(); // total trust metric
        if (t >= thres) { // nodes selected only trust value greater than
threshold
                coun = coun + 1;
                h = 0.4*nbr.get_nbrEnergy() + 0.6*(dmin/Point2D.distance(nbr.x,
nbr.y,  x, y));
                if (h > shortest) {
                        shortest = h;
                        ne = nbr; // selected next hop
                } // find the short distance neighbour in the safe list as next
hop
        }
}
```

While the safe forwarding list is empty, DTEGR algorithm will adjust the trust threshold value until the list is not empty any more. The scripts for threshold value adjustment are in below.

```
while (coun == 0 && thres >= 0.1) {
        thres = thres - 0.1;
        for (int i = 0; i < no; i++) {
                GPSR_Nbr nbr = (GPSR_Nbr) gpsr_nbr_list.elementAt(i);
                t = nbr.calc_totalTrust(); // total trust metric
                if (t >= thres) {
                        coun = 1;
                        h = 0.4*nbr.get_nbrEnergy() +
0.6*(dmin/Point2D.distance(nbr.x, nbr.y,  x, y));
                        if (h > shortest) {
                                shortest = h;
                                ne = nbr; // selected next hop
                        }
                }
        }
```

```
} // finding the safe list and next hop with lower threshold, TIGER dynamic
weight factor algorithm black list
```

Java scripts for FEATGR algorithm

In below are the sample scripts for the fuzzy membership inputs of direct trust and distance metrics.

```
t = nbr.directTrust;
int tinc1 = 0;
int tinc2 = 0;
double tl1 = 0.0;
double tl2 = 0.0; // trust membership variables
if (t >= 0.7) {
	tinc1 = 3;
	tl1 = 1;
} else if (t >= 0.4 && t <= 0.6) {
	tinc1 = 2;
	tl1 = 1;
} else if (t <= 0.3) {
	tinc1 = 1;
	tl1 = 1;
} else if (t > 0.6 && t < 0.7) {
	tinc1 = 3;
	tl1 = ((t-0.6) * 10);
	tinc2 = 2;
	tl2 = ((0.7 - t) * 10);
} else if (t > 0.3 && t < 0.4) {
	tinc1 = 2;
	tl1 = ((t-0.3) * 10);
	tinc2 = 1;
	tl2 = ((0.4 - t) * 10);
} // assign trust membership and level

dis = 1 - ((Point2D.distance(nbr.x, nbr.y,  x, y) - myp) / 448); // distance
between 0 - 1

int dinc1 = 0;
int dinc2 = 0;
double dl1 = 0.0;
double dl2 = 0.0; // trust memebership variables
	if (dis >= 0.8) {
		dinc1 = 3;
		dl1 = 1;
	} else if (dis >= 0.5 && dis <= 0.7) {
		dinc1 = 2;
		dl1 = 1;
	} else if (dis <= 0.4) {
		dinc1 = 1;
		dl1 = 1;
	} else if (dis > 0.7 && dis < 0.8) {
		dinc1 = 3;
		dl1 = ((dis-0.7) * 10);
		dinc2 = 2;
```

```
            dl2 = ((0.8 - dis) * 10);
      } else if (dis > 0.4 && dis < 0.5) {
            dinc1 = 2;
            dl1 = ((dis-0.4) * 10);
            dinc2 = 1;
            dl2 = ((0.5 - dis) * 10);
      } // assign distance membership and level
```

In below are the scripts for centroid defuzzification for the FEATGR algorithm,

```
double finalscore, fdist; // centroid defuzzification
finalscore = ((i1*f1)+(i2*f2)+(i3*f3)+(i4*f4)+(i5*f5)+(i6*f6)+(i7*f7)+(i8*f8)
+ (i9*f9) + (i10*f10) + (i11*f11) + (i12*f12) + (i13*f13) + (i14*f14) +
(i15*f15) + (i16*f16)) / (i1 + i2 + i3 + i4 + i5 + i6 + i7 + i8 + i9 + i10 +
i11 + i12 + i13 + i14 + i15 + i16);

if (finalscore > fmax) {
      ne = nbr;
      fmax = finalscore;
}
```