FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Trust Management in Wireless Ad Hoc Networks

by

Yelena Trofimova

A dissertation thesis submitted to
the Faculty of Information Technology, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

Dissertation degree study programme: Informatics
Department of Computer Systems

Prague, February 2022

Supervisor:
prof. Ing. Pavel Tvrdík, CSc.
Department of Computer Systems
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Co-Supervisor:
Ing. Alexandru Mihnea Moucha, Ph.D.
Department of Computer Systems
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Abstract and contributions

This dissertation deals with trust management in wireless ad hoc networks. These networks are vulnerable due to the absence of central management and fixed infrastructure. Cooperation and communication challenges are frequently addressed by deploying a trust management scheme.

This dissertation first generalizes the notion of trust and then describes trust management schemes. Next, we developed a new method for managing trust in wireless ad hoc networks.

The method is based on neural networks, and the packet delivery ratio is used to determine the trust of nodes. We demonstrated that neural networks are capable of trust estimation and evaluation in ad hoc networks. The second part of the dissertation describes a method integration into ad hoc routing protocols.

In particular, the thesis has two main contributions:

1. Design of a trust management scheme for wireless ad hoc networks using neural networks (Chapter 5).

2. Enhancing reactive as hoc routing protocols with trust (Chapter 6).

Keywords:
wireless ad hoc network, packet delivery ratio, trust, neural network, reactive routing, route discovery.

Abstrakt a přínosy

Tato disertační práce se zabývá problematikou důvěryhodnosti v bezdrátových ad hoc sítích. Takovéto sítě jsou zranitelné zejména kvůli absenci centrální správy identit uzlů a pevné infrastruktury. Důvěra v oblasti bezdrátové komunikace se standardně řeší nasazením mechanismů vhodných pro stanovení důvěryhodnosti jednotlivých uzlů.

Hlavními úkoly disertační práce jsou zobecnění pojmu důvěryhodnosti a popis schémat týkajících se možností její správy. Na základě stanovených požadavků byla navržena nová metoda pro stanovení důvěryhodnosti.

Tato metoda využívá matematické modely, které jsou založeny na neuronových sítích. Stanovení důvěryhodnosti uzlů vychází z měření hodnoty poměru odeslaných a doručených paketů. Bylo prokázáno, že neuronové sítě jsou pro odhadování důvěryhodnosti uzlů v ad hoc sítích vhodné. Druhá část disertační práce je věnována integraci konceptu důvěryhodnosti do obecných ad hoc směrovacích protokolů.

Hlavní přínosy disertační práce jsou následující:

- Návrh schématu správy důvěryhodnosti pomocí neuronových sítí (Kapitola 5).
- Integrace konceptu důvěryhodnosti do reaktivních ad hoc směrovacích protokolů (Kapitola 6).

Keywords:
bezdrátová ad hoc sít, poměr doručení packetů, důvěryhodnost, neuronová sít, reaktivní směrovaní, objevování cesty.

Acknowledgements

This dissertation thesis would not have been possible without my family: Murzagali, Tanya, Viktor, Petra and Radek. Thank you for your love, support, patience, encouragement and inspiration.

I would like to express my gratitude to the thesis supervisor, prof. Ing. Pavel Tvrdík, CSc. for the guidance and, most importantly, critical feedback that made my thesis a lot better than it would have been.

I would like to thank my thesis co-supervisor, Ing. Alexandru Mihnea Moucha, Ph.D. for quick responses and proofreading.

Finally, a much deserved special thanks to Ing. Jan Fesl, Ph.D, for the valuable insight and motivation.

My research has been partially supported by the Grant Agency of the Czech Technical University in Prague, grants No. SGS16/123/OHK3/1T/18 and SGS17/214/OHK3/3T/18.

Dedication

In memory of my mother.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

AODV	Ad-hoc On demand Distance Vector protocol
BATMAN	Better Approach to Mobile Ad-hoc Networking protocol
CS	Candidate Set
CSMA/CA	Carrier-Sense Multiple Access with Collision Avoidance
DFF	Destination only Flag False
DFT	Destination only Flag True
DSDV	Destination Sequenced Distance Vector routing
DSR	Dynamic Source Routing
DYMO	DYnamic Manet On-demand
ExNT	Expected Number of Transmissions
FANN	Fast Artificial Neural Network
IDE	Integrated Development Environment
MAC	Medium Access Control
MANET	Mobile Ad-hoc NETwork
MD	Maximum Delay value
MLP	Multi-Layer Perceptron
NN	Neural Network
NED	NEtwork Description
NeNTEA	Neural Network Trust Evaluation in Ad hoc networks

NIC	Network Interface Card
NTT	Node Trust Table
NTV	Node Trust Value
OGM	OriGinator Message
OLSR	Optimized Link State Routing
OMNeT++	Objective Modular Network Testbed in C++
OR	Opportunistic Routing
P2P	Peer-to-Peer
PB	Packet Buffer
PDR	Packet Delivery Ratio
PL	Path Length
RBF	Radial Basis Function
RFC	Request For Comments
Rprop	Resilient backpropagation
RERR	Route ERRor
RREP	Route REPly
RREQ	Route REQuest
RTS	Request To Send
TARA	Trust-Aware Reactive Ad hoc routing
THR	THReshold value
TMS	Trust Management Scheme
VANET	Vehicular Ad-hoc NETwork
WANET	Wireless Ad-hoc NETwork

Chapter **1**

Introduction

Computer networks have become an ubiquitous part of our lives. Wireless ad hoc networks (WANETs) are used in cases where it is not feasible or efficient to build a network infrastructure. The examples are disaster relief operations, surveillance, sensors for environmental monitoring [37]. While such networks can be deployed fast and are easily scalable, they pose challenges for creation of network protocol and maintaining security. Generally, wireless networks are more vulnerable than wired networks due to the shared nature of the transmission medium. Furthermore, ad hoc communication implies, that data is routed by nodes in the network, resulting in additional vulnerability. The network area is often much larger than radio range of a single node, therefore, nodes use other nodes as relays. This type of communication is called a multi-hop routing [54]. Multi-hop transmission assumes that devices are fully cooperative and honestly provide routing and forwarding functions [26]. Although cryptographic methods improve the network's robustness, they cannot address a variety of vulnerabilities caused by nodes behavior, such as selfish forwarding or selective dropping of data. Methods to monitor nodes behavior and penalize deviations from the expected cooperative protocol are requires in this case. In such methods the behavior of nodes is typically quantified as a trust metric.

In this chapter, we introduce the topic of the thesis, explain the motivation behind it, mention related work, and formulate the goals of the research.

The field of wireless rapid deployable networks has received a lot of attention in the past two decades because of the exponential growth and evolution of wireless communication. With the world becoming more mobile and dynamic each year, the applications of WANETs are more and more common.

WANETs are a broad category that includes a variety of network types depending on their intended use. We can have wireless networks of mobile devices, vehicles, sensors, or IoT devices.

We will define a *wireless ad hoc network (WANET)* as a set of battery-powered, wireless nodes connected on-the-fly. Every node plays the roles of both a router and an end station at the same time. Nodes located inside the radio range can communicate directly; otherwise they need to use intermediate nodes for data delivery. The lack of infrastructure or central management is an advantage from the point of view of scalability and resilience, but it poses security risks and requires cooperation between nodes for the network to function correctly. Therefore, mechanisms to enforce the collaboration are required. Constraints such as the deficiency of computational power or poor energy resources prevent the use of classical key-establishment schemes and central certification authorities. To reduce potential threats, the concept of trust was introduced.

One of the major causes of WANET performance degradation is the dropping of data packets. The nodes have to make a trade-off between saving the battery (behaving selfishly) and forwarding packets of other nodes to maintain the network functionality. A part of this trade-off may be the decision of a node to drop some of the data packets in transit. Thus, dropped data packets may be a sign of selfish or even malicious behavior, resulting in performance degradation of the network. One of the security criteria for a WANET is *availability*. Availability is the ability of a node to deliver the entire set of network-specific services irrespective of its security stature [15]. Let us assume that a packet is dropped. From the viewpoint of the sender, it is generally hard to find out which node is responsible and what was the reason (selfishness in saving the battery, interference, malicious behavior, etc.). Even if we have information from other nodes, it is difficult to decide whether a given node behaves as expected, i.e., whether it properly functions as a router forwarding the packets of others and is therefore trusted or not.

Trust in this scope means a measure of confidence that a node will behave according to expectations. The trust concept in this case provides a simple and feasible way to enhance the security of the network. The security schemes that govern trust among communicating entities are collectively known as *trust management schemes (TMSs)* [45]. TMSs are used for different purposes, such as authentication, access control, intrusion detection, or secure routing[16]. Moreover, trustis important in clouds, the edge computing, IoT, automotive computing, blockchain, and AI [13].

Ad hoc routing protocols by design rely on the fact that nodes cooperate in route discovery. For example, the AODV RFC says that the protocol is intended for use in networks where all nodes can trust each other [20]. However, this is not always the case. The reasons why nodes do not cooperate can vary. However, the result is the same. Due to unexpected node behavior (i.e., the existence of untrusted nodes), the network performance degrades or security is compromised.

Our goal is to improve the functionality of ad hoc routing protocols so that they can work more reliably in WANETs with untrusted nodes.

Incorporating trust in reactive ad hoc routing protocols can be solved by changing the protocol, i.e by changing the protocol messages and structures.

The problem we are going to tackle is to enable trust-aware ad hoc routing without changing the routing protocol. In this work, we propose one possible solution to the stated problem.

The constraints of WANETs restrain the deployment of traditional security mechanisms of wired networks, such as authentication protocols, digital signature, and encryption. Traditional security mechanisms still play role in achieving security goals in WANETs. However, these mechanisms are not sufficient by themselves [78]. Routing protocols in wired networks can be assumed to be executed by trusted entities, namely the routers [5]. WANETs by nature do not have this property. Thus, for ad hoc routing protocols finding a route with specific trust levels is more relevant than finding the shortest route between the two end points [45].

Various methods to assess trust in WANETs were proposed. In most of them, certain information from the network through interaction between nodes or eavesdropping is collected. Then, the computation of some metrics takes place, and (in most cases) recommendations from other nodes are collected [44, 66, 11]. Based on these pieces of information, the value of trust for a particular node is calculated. The main disadvantage of eavesdropping is the fact that it may exhaust the computational and battery resources of the nodes. The aim of this research is to avoid such resource intensive approach to trust management in WANETs.

For more details on related work, see Chapter 3.

1.1 Structure of the Dissertation Thesis

The thesis is organized into 7 chapters as follows:

1. *Introduction*: Introduces the topic of the dissertation thesis and the motivation behind our work.

2. *Trust Management in Ad Hoc Networks, Background*: Describes the necessary theoretical background. Focuses on the properties of WANETs, notion of trust, outlines neural networks (NNs), and routing protocol.

3. *Previous Results and Related work*: Surveys the previous results and related work.

4. *Goals of the Dissertation Thesis*: Includes a detailed statement of the goals to solve.

5. *The 1 st Contribution: the NeNTEA Method*: Explains the details of the suggested solution. The assumptions and ideas behind the approach are specified. Covers the environment and experiment setup. Defines metrics for measuring quality of the approach, describes the experiments, and reports the obtained results. Demonstrates that artificial NNs can be used for evaluation of trust in WANETs, namely, for detection of untrusted nodes and for estimation of the trust values. States the benefits of the proposed NN approach – Neural Network Trust Evaluation in Ad hoc networks (NeNTEA).

6. *The 2 nd Contribution: the TARA Method*: Further, we describe our proposed method (Trust-Aware Reactive Ad hoc routing (TARA) of enhancing reactive ad

hoc routing protocols with trust, present the simulator, describe the experiments and their results, and discuss the considerations regarding the implementation of the proposed method.

7. *Conclusions*: Summarizes the results of our research, suggests possible topics for further research, and concludes the thesis.

Chapter 2

Trust Management in Ad Hoc Networks, Background

This chapter is dedicated to the theory related to the dissertation thesis topic. Section 2.1 introduces structure and properties of WANETs. The trust concept and TMSs are described in Section 2.2. Routing protocols and NNs are described in Sections 2.3 and 2.4, respectively.

2.1 Ad Hoc Networks

An ad hoc network consists of *nodes*; see Figure 2.1. These are small devices, consisting of an antenna and radio transceiver, processing unit, memory, and battery. From the point of view of this research, a detailed description of processing unit and memory is not important. Energy consumption is also a topic of another research. In the further text, we need just the description of antenna and radio transceiver.

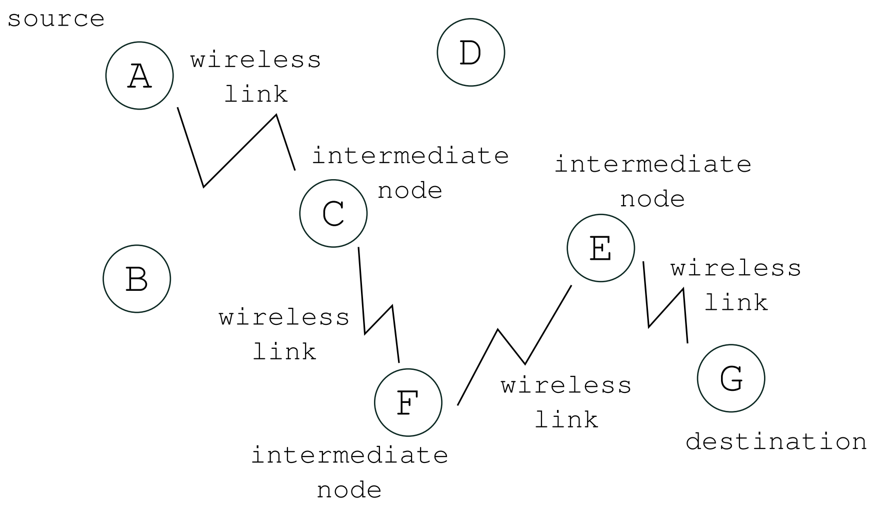


Figure 2.1: An ad hoc network.

For wireless communication, the antenna is probably the most critical element. An *antenna* is usually a metal device that is capable of sending and receiving data. In other words, the antenna is the transitional structure between free-space and a guiding device. The guiding device or transmission line may take the form of a coaxial line or a hollow pipe (waveguide), and it is used to transport electromagnetic energy from the transmitting source to the antenna, or from the antenna to the receiver [7]. Radio transceivers operate in half-duplex mode, since transmitting and receiving data at the same time is not possible.

The fundamental parameter of the antenna is the *radiation pattern*. It is defined as "a mathematical function or a graphical representation of the radiation properties of the antenna as a function of space coordinates" [7]. Radiation properties can include power density, field strength, directivity, polarization, or phase. The most interesting is the space distribution of related energy, so the range of node can be determined. A graph of the spatial distribution of the power density is called a *power pattern* and usually is plotted in decibels.

The radiation pattern can be isotropic, directional, and omnidirectional. The *isotropic* pattern means the equal radiation in all directions. It is not reachable in practice, but serves as a reference for expressing directional properties for real antennas. A *directional* antenna sends/receives more effectively in a particular direction. An antenna whose radiation pattern is nondirectional in one plane and directional in the orthogonal plane, is called *omnidirectional* and it is represented in Figure 2.2.

The *range* of antenna is the distance within which the node can communicate. It grows with the square root of power: to double the communication distance requires to increase the power four times. The communication range implies which nodes will be able to communicate.

Choosing transmission power level has an important impact on energy efficiency and network lifetime [40]. We assume, that some distributed algorithm for transmission power control is used.

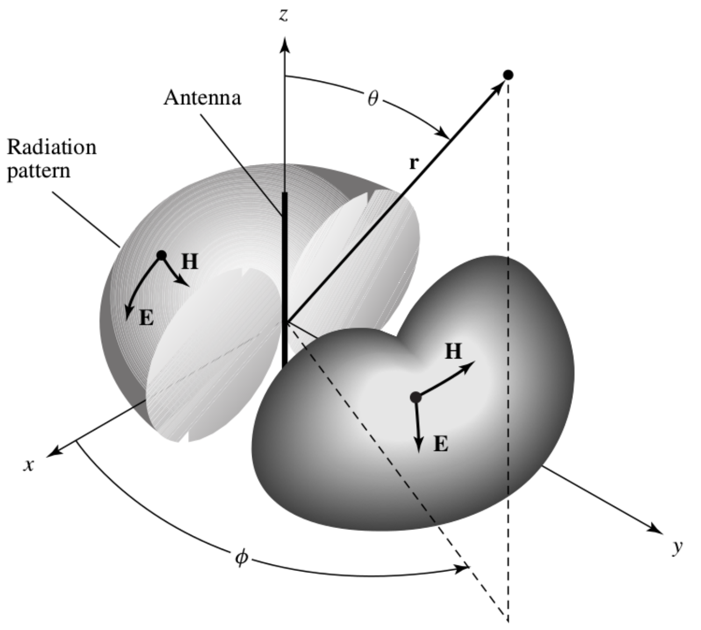


Figure 2.2: Omnidirectional antenna pattern. (SOURCE: C. A. Balanis, *Antenna Theory: Analysis and Design*, Wiley-Interscience. Copyright © 2005).

An ad hoc network can be viewed as a graph $G = (V, E)$, where V is the set of its nodes and E is a set of links between nodes. The graph G is also called the *topology* of the network. A *link* between two nodes A and B is formed when A and B are able to radio communicate and we suppose the link to be symmetrical (if A can successfully transmit radio signals to B, then B can successfully transmit radio signals back to A). A node B is a *neighbor* of node A, if it is in the maximum transmission range. The nodes adjust their transmission power to be high enough to reach the intended neighboring destination while causing minimal interference at other nodes.

Communication between nodes that are not neighbors can be performed using *intermediate* nodes as routers/relays.

Definition 2.1.1. A *path* between two nodes is an ordered sequence of intermediate nodes that need to be passed to get the message from the *source* node to the *destination* node (see Figure 2.1).

Definition 2.1.2. Let $P = \langle N_1, N_2, \ldots, N_l \rangle$ be a path from source N_1 to destination N_l, then *path length* of path P, $PL(P)$, is defined as the number of hops from the source node to the destination node:

$$PL(P) = l - 1, \tag{2.1}$$

where $l = |P|$ is the number of nodes in the path.

Paths are constructed by routing protocols.

Definition 2.1.3. We say that the source *uses* intermediate nodes to deliver the data.

Definition 2.1.4. We assume that every node in a WANET can compute routing paths (sequences of intermediate nodes) to all other nodes. For simplicity and further reference, the collection of this information about paths from every node is called a *network communication arrangement.*

An essential property of each node is its ability to forward packets.

Definition 2.1.5. The *packet delivery ratio (PDR) of a node* N_i is defined as follows:

$$PDR(N_i) = \frac{\phi(N_i)}{\sigma(N_i)}, \tag{2.2}$$

where $\phi(N_i)$ is the number of data packets correctly forwarded by node N_i and $\sigma(N_i)$ is the total number of data packets sent to node N_i that were supposed to be forwarded.

Definition 2.1.6. Let $P = \langle N_1, N_2, \dots, N_l \rangle$ be a path from source N_1 to destination N_l. Then the *PDR of path P* from the viewpoint of the source node N_1 towards a destination node N_l, is defined as follows:

$$PDR(P) = \frac{\phi(P)}{\sigma(P)}, \tag{2.3}$$

where $\phi(P)$ is the number of data packets correctly forwarded to the destination N_l through the path P and $\sigma(P)$ is the total number of data packets sent from N_1 to N_l using P.

Theorem 2.1.7. The PDR of path P is computed as a product of PDRs of intermediate nodes in P:

$$PDR(P) = \prod_{i=2}^{l-1} PDR(N_i) \tag{2.4}$$

Here source node N_1 and destination node N_l are not considered since they are not intermediate nodes.

Proof. Suppose that a WANET is running for a sufficiently long time for the collected information about PDRs of the paths to be statistically significant. The PDR of a given path can be viewed as a probability of delivering data packets through that path. The events of successful forwarding of the data packet by N_2, N_3, ..., N_{l-1} are independent. Using the multiplication rule of probability [67] that states that in case of n independent events $A_1, A_2, \dots, A_n$ probability P that all of them occur is:

$$P(A_1 \cap A_2 \cap \cdots \cap A_n) = P(A_1) \times P(A_2) \times \cdots \times P(A_n), \tag{2.5}$$

where $P(A_i)$ is the probability of event A_i. □

2.2 Notion of Trust

The notion of trust originally comes from social disciplines and is defined as a measure of subjective belief regarding the behavior of a certain entity [19]. Within several last years a theory of trust in relation to WANETs was gradually developed. The theory discusses the concept and properties of trust and is generally accepted among researchers. Definition of trust according to [13]:

Definition 2.2.1. *Trust is the assurance that one entity holds that another will perform particular actions according to a specific expectation.*

Fundamental properties of trust are dynamicity, subjectivity, incomplete transitivity, asymmetry, and context-dependency [16].

Dynamicity means that a WANET infrastructure is not static. Nodes can join and leave time to time, some nodes can fail, thus network status can change, and trust values should reflect such changes. Nodes can have incomplete and partially local information about the situation in the whole network.

Subjectivity comes from social disciplines and indicates biased nature of trust evaluations based on different experience. Trust in psychology makes accent on the cognitive process, implying that humans acquire trust values from their experiences.

Nontransitivity or *incomplete transitivity* means that if node A trusts node B and node B trusts node C, that does not imply that node A trusts node C. Although, using recommendations from other nodes about a particular node trust results in partial transitivity.

Trust has property of *asymmetry*. The fact that a node believes in another's node trustworthiness does not induce its trustworthiness in return.

The last but not least property of trust is *context-dependency*. Li and Singhal [41] define trust as the belief that an entity is capable of performing reliably, dependably, and securely in a particular case; hence, different levels of trust exist in different contexts. For example, one can trust his doctor's advice on health issues, but does not trust the doctor if he makes advice on finance management.

One more important aspect of trust is its diminishing in time. Many studies use an aging mechanism embedded in trust evolution calculations.

Jøsang & Pope define a notation for describing and reasoning about trust [35]. They use the term *trust purpose* to represent the semantic content of a particular trust instance. For example, node A can trust node B for the purpose of forwarding packets. There are five elements to form trust: trustor (trust originator), trustee (trust target), trust purpose, trust measure, and time. If trust purposes are the same, trust can be transitive.

2.2.1 Metrics for Measuring Trust

Trust management protocols use various metrics to compute trust. Metrics should justly reflect the situation in a WANET with respect to trust calculations. To determine which of them to use is not a simple task.

Taking into account the economic basis of trust, the selfishness of nodes should be considered, as trust in economics is based on the assumption that humans are maximizers of their own interest [16]. Selfishness can take form of dropping data packets or ignoring control messages of a routing protocol.

Trust also implies willingness to take a risk of losing data, so trust is often linked to the level of reliability. Reliability in terms of networking means a guarantee of data delivery. So, the metric is the packet delivery ratio. Josang et al. [36] and Solhaug et al. [64] conclude that trust is generally neither proportional nor inversely proportional to risk. Risk is closely connected to stake, which is value of the outcome of a risky operation, how bad the loss of particular data is. Even when the trust is strong, if the stake is high, the risk will be also high.

On account of trust to be context-dependent, trust metrics should return adequate references for diverse situations, depending on how sensitive the information is.

In our research, trust reflects the confidence that a given node will forward data accurately (correctly). We define *trust* in this scope as a measure of confidence that a node will cooperate – by properly delivering data in transit, sourced by, or destined for other nodes. The previously defined *PDR* is used as a metric of trust value, therefore we use the notions *"PDR"* and *"trust value"* interchangeably in this dissertation. *PDR* as a metric of trust is often used [44, 21, 3, 29, 8], since the most critical feature of networks is their ability to deliver data. PDR plays a decisive role in determining a node's behavior as described in [15], chapter 5, section *The Trust Model*, trust calculations depend on the PDR information.

Definition 2.2.2. The *trust of a node* N_i is its estimated $PDR(N_i)$ and it is a real number between 0 and 1: $PDR(N_i) \in \mathbb{R}, 0 \leq PDR(N_i) \leq 1$.

Authors in [35] model trust of a path as a multiplication of nodes trust, which corresponds to our model of calculation path *PDR* (Equation 2.4).

Definition 2.2.3. A node N_i is considered *trusted* if $PDR(N_i) > \tau$, where $0 \leq \tau \leq 1$ is the given *threshold*. If $PDR(N_i) \leq \tau$, then the node is *untrusted*.

In the context of this dissertation, a node whose trust value is to be estimated is called an *investigated node*.

2.2.2 Trust Management Schemes

Some considerations underlying the construction of trust-based methods exist. The main aspect is the necessity of their distributed deployment based on cooperative evaluation of trust. As TMSs mostly rely on some routing protocol, they should be flexible and should not cause disruptions of computations on nodes or resource exhaustion. TMSs must count with that that nodes by default cannot guarantee cooperation. Also, for a TMS to meet the dynamic nature of WANETs, it should be capable of self-reconfiguring. Each TMS is a trade-off between WANET performance and security.

Trust basically can be of two types: *direct*, obtained by individual observations of a node, and *indirect*, learned from others.

Trust management consists of three stages: initiation, evaluation, and amendment.

Initiation: when a node connects to a WANET, it is typically assigned a default trust value. In this case, other nodes have to take a risk when starting a communication with an unknown node.

Trust *evaluation* means a method of calculation of trust values. These calculations can be based on various factors:

- previous direct interactions,
- observation of the activity of the node,
- recommendations received from others.

It is also necessary to carry out periodic updates of trust values. It is done during the *amendment* stage.

When the trust value is available, a node will then compare it with the specified threshold and make a decision if the particular node is trusted. Implementation of trust calculations and modifications in different TMSs vary.

2.3 Ad Hoc Routing Protocols

Generally, routing protocols for WANETs can be proactive or reactive. The goal of *proactive* protocols is to maintain consistent, up-to-date routing information, which requires periodic exchange of control messages between the nodes. Hence proactive protocols can waste resources unnecessarily [14]. In contrast to this, *reactive* protocols establish a route between a source and a destination only when the source wants to send something to the destination. Because of this, reactive protocols are also known as *on-demand* protocols.

Proactive protocols maintain routing information by distributing routing tables throughout the network, thus causing higher overhead. Due to this proactive protocols (OLSR, Destination-Sequenced Distance-Vector Routing (DSDV)) are not commonly used for trust-based schemes. Instead, reactive protocols are preferable here because only when a route to some destination is needed, the route discovery process is started. This way, communication overhead is lower and node batteries are saved compared to proactive routing protocols [43].

The most prevalent reactive protocols are AODV and DSR. Their disadvantages are higher latency and flooding the network during route discovery.

The DSR protocol is similar to AODV in terms that it is on-demand routing, but it does not rely on routing tables at the intermediate nodes, since the source node adds the whole path to the data packet it sends.

We need to mention that neither AODV nor DSR choose a route based on the state of links but based on the shortest path to the destination leading through possibly untrusted nodes.

2.3.1 The AODV Protocol

The AODV protocol [20] is a reactive routing protocol, which implies that a route discovery process is initiated when a route to a specific destination is needed. Route discovery is accomplished by flooding the network with RREQ messages. The routing table on each node contains information about the next hop to the destination, the number of hops, and the expiration time of the routing table entry. Each time the route is used for forwarding data, its expiration time is updated. Thus, in the case of periodic communication, the routing table entry may never expires.

In the AODV protocol, all nodes work collectively to discover a route path from a source to a destination. An actual data transfer takes place only after the route is established. There are three types of control messages: RREP (Route Reply), RREQ (Route Request), and RERR (Route Error). To find a path, the source broadcasts RREQ message to the network. On receipt of a RREQ message, a node sends a RREP message, if it is the destination or if it has a fresh enough route to the destination. Otherwise it just forwards the RREQ message to its neighbors. On receipt of a RREQ message, the node increases the hop count field in the message by one, and on receipt of a RREP message, intermediate nodes update their route entry with the new data (see Figure 2.3). Whenever a node sends a new RREQ, RREP, or RERR message, it increases its own sequence number. The higher the sequence number, the more that information is considered. A path with the smallest hop count is chosen [52].

Simplified algorithm of AODV protocol operation is represented in Figure 2.3.

There are two ways of creating routes in the AODV protocol. Protocol's behavior depends on the flag called *Destination only.* When the flag is set (Destination only Flag True (DFT)), only the destination node can reply to a RREQ message. This implies that every time a new route is requested, a new original route from the source to the destination is created, without any influence of already existing routes. On the other hand, when the flag is not set (Destination only Flag False (DFF)), any node that has a route to the destination can reply to the RREQ message. As a result, the routes that already exist influence the searching process. The existing routes are then used by many nodes and even though a better route exists, it is not discovered. The difference in the created routes can be seen in Figure 2.4. Nodes are depicted as circles with their IDs written up left from them. The node with $ID = 0$ is the sink. Grey links, although they exist, are not used in any route. The thicker the link is, the more routes are going through it. Topology in Figure 2.4a was created with DFF. Compared to 2.4b, routes in 2.4a were attached to the existing ones, and thus the topology has fewer "branches".

AODV scales well to large networks, as shown in [52]. It is a widely accepted choice for reactive routing [58].

2.3.2 The DSR Protocol

The Dynamic Source Routing protocol (DSR) [34] is also a reactive protocol. Its route discovery process is similar to AODV [53]. To discover a path the source node broadcasts a

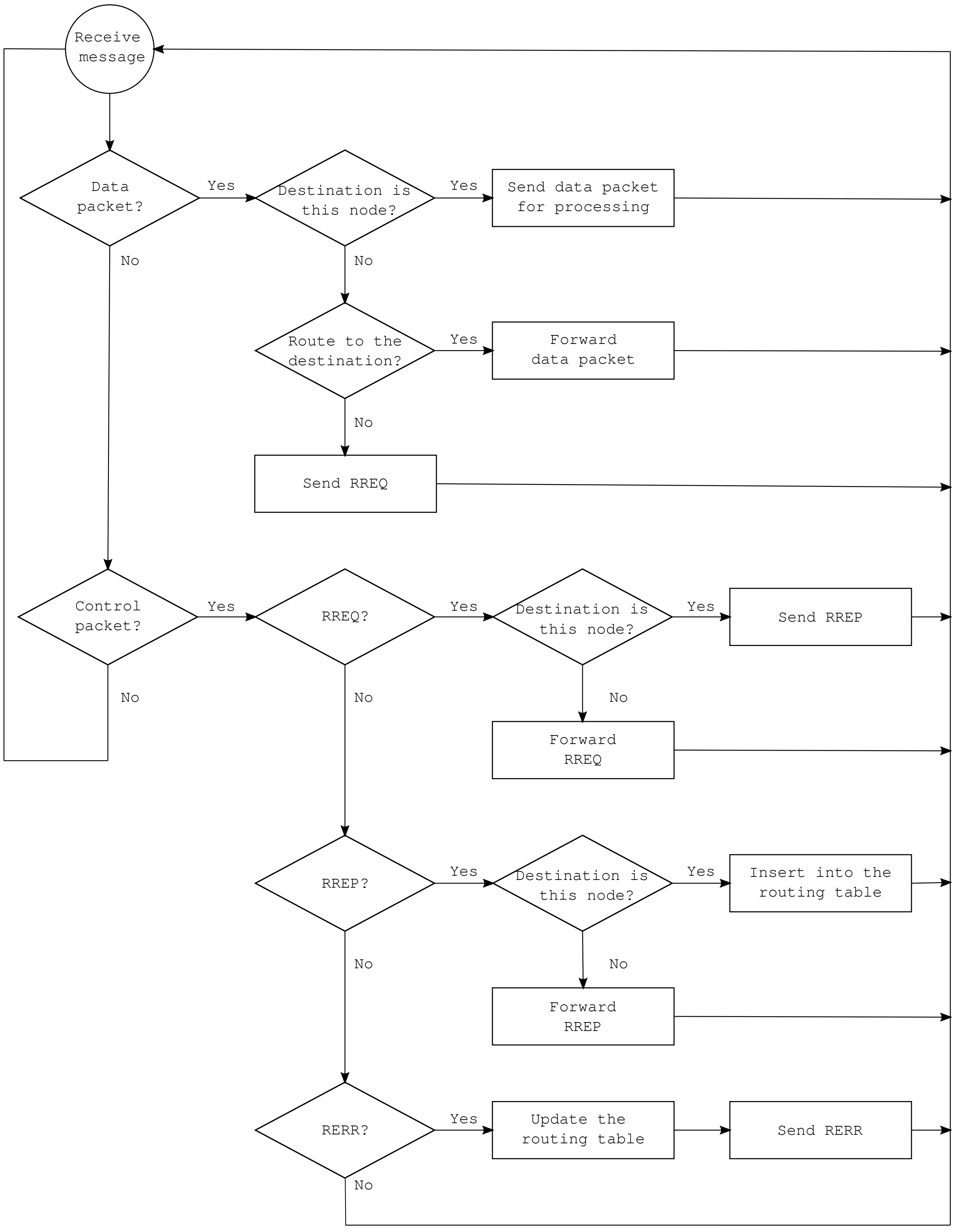
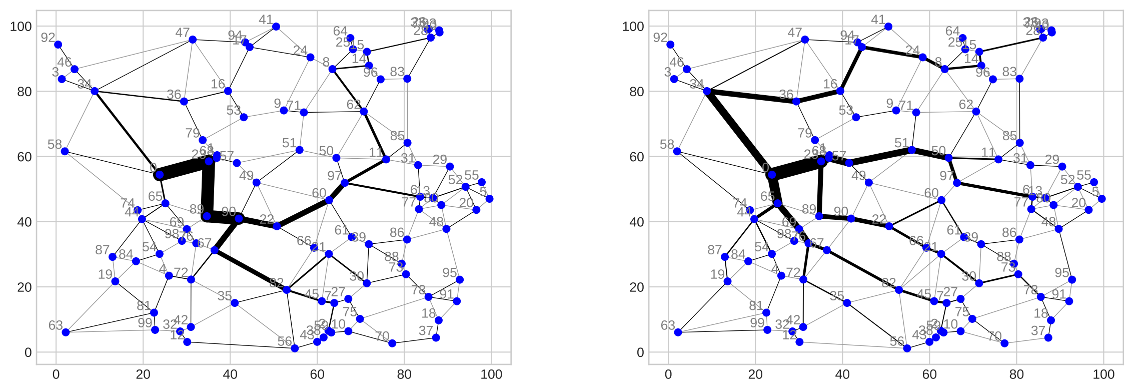


Figure 2.3: The AODV algorithm.



(a) *Destination only* Flag is set to False (DFF) (b) *Destination only* Flag is set to True (DFT)

Figure 2.4: Topologies created by the AODV protocol.

RREQ message that is forwarded by the intermediate nodes. However, DSR uses so called *source routing*. The intermediate nodes append their IDs to the RREQ message. This sequence of intermediate nodes is returned in the RREP message and saved by the source node. When a data packet is sent, it contains a list of intermediate nodes it will traverse on its way to the destination.

Compared to AODV, the DSR protocol keeps the information about multiple routes to the destination. According to its RFC, DSR is designed for networks of up to 200 nodes and of small diameter up to 10 hops [34]. Another difference is that there is no expiration time for routes.

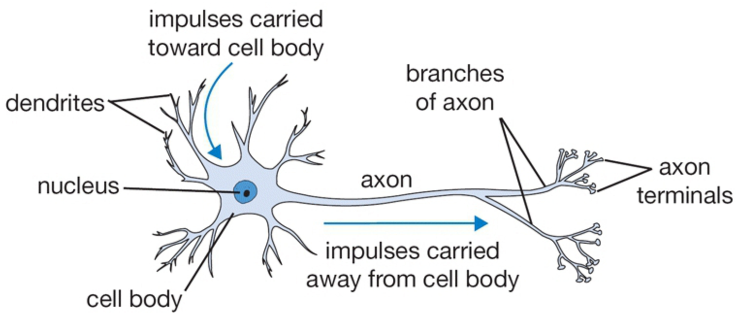
To store routing information, the DSR protocol implements *Route Cache* data structure, that contains discovered routes. It can also implement *Link Cache* data structure, where each individual link (edge of the topology) learned from the control messages is stored. Hence, the Link Cache keeps dynamically the current list of edges of the discovered part of the topology. In case of Link Cache, to search for a route, the source node performs Dijkstra's shortest path algorithm to find the best path to the destination [34].

2.4 Neural Networks

2.4.1 Model of Neuron

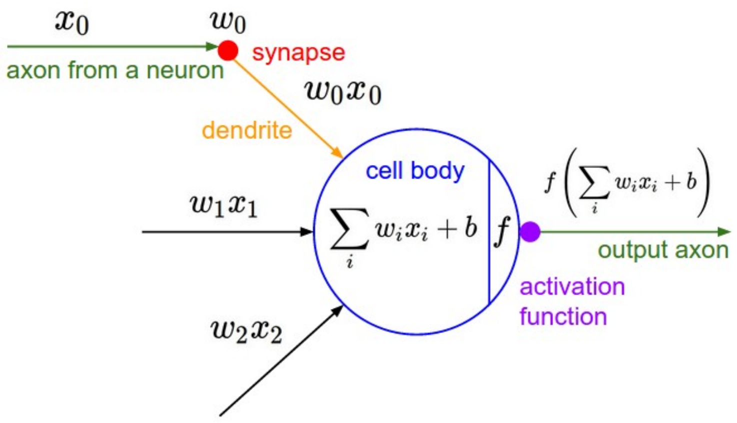
The biological neuron is the prototype and inspiration for the mathematical model of artificial NNs (Figure 2.5). It has several dendrites (inputs) and one axon (output). Signals from dendrites traverse through the neuron body, which in mathematical model is implemented with an *activation* function.

The axon connects to other dendrites via synapses, which have different strength



impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body

(a) Biological neuron



(b) Mathematical model

Figure 2.5: Models of a neuron. (SOURCE: Karpathy, Andrej. *Neural Networks Part 1: Setting Up the Architecture.* Notes for CS231n Convolutional Neural Networks for Visual Recognition, Stanford University, 2015 [38]).

(weights: $w_0, w_1, \ldots$) and thus can be excitatory or inhibitory. First, the weighted sum of inputs is calculated and then passed as an argument to the activation function. Usually bias b is used to shift the activation function to the left or right. Generally, if the value of activation function is greater than some threshold, a signal is fired to the output.

The most common activation function is *sigmoid* (Equation 2.6, Figure 2.7), which is used in multi-layer perceptron networks.

$$S(t) = \frac{1}{1+e^{-t}} \tag{2.6}$$

2.4.2 Architecture of Neural Networks

As stated in [71] a Neural Network (hereafter NN) is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection weights, and the processing performed at computing elements or nodes.

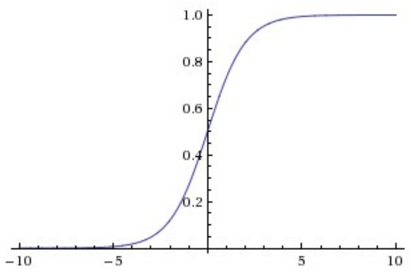


Figure 2.6: Sigmoid activation function. (SOURCE: Karpathy, Andrej. *Neural Networks Part 1: Setting Up the Architecture.* Notes for CS231n Convolutional Neural Networks for Visual Recognition, Stanford University, 2015 [38]).

In essence NN is a structure of connected nodes with some number of inputs and outputs, embedded activation function and each connection has a weight assigned to it. A NN should be considered when input data is high-dimensional or possibly noisy and the transformation function is unknown.

A lot of different types of NN were created: multi-layer perceptron (MLP), radial basis function (RBF), Kohonen features maps, Hopfield networks, Boltzmann machine and other.

Each NN is characterized by:

- model of neurons (their inherent properties and activation functions),
- topology of the NN,
- learning method.

Several NN topologies are distinguished according to signal flow:

- single-layer feed-forward,
- multi-layer feed-forward,
- recurrent.

In feed-forward NNs information always flows in one direction. Recurrent topologies allow cycles. There is at least one feed-back connection.

Single-layer NNs consist of one input layer and one output layer of processing units. Multi-layer NNs have one or more hidden layers of processing units.

Recurrent NNs may or may not have hidden layers. There are also further variations of topology, like short-cut connections, partial connectivity or time-delayed connections.

A multi-layer feed-forward NN with perceptron neurons with sigmoid activation function was used in this work and is schematically represented in Figure 2.7.

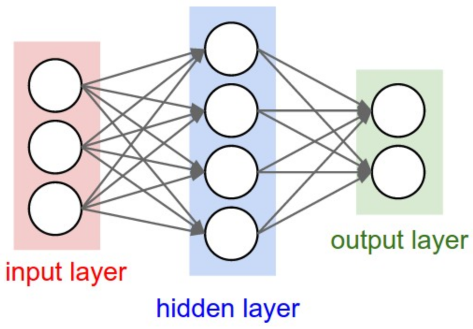


Figure 2.7: A multi-layer feed-forward NN. (SOURCE: Karpathy, Andrej. *Neural Networks Part 1: Setting Up the Architecture.* Notes for CS231n Convolutional Neural Networks for Visual Recognition, Stanford University, 2015 [38]).

2.4.3 Learning Methods

A NN needs to be configured for a particular problem. This is usually achieved by process called *learning*. Learning consists in adjusting connection weights to get the desired output. For learning a set of known input/outputs is needed. After feeding the system with input and getting the result, the connection weights are modified to obtain a more fitting output. So node connection weights are used to store the learned knowledge.

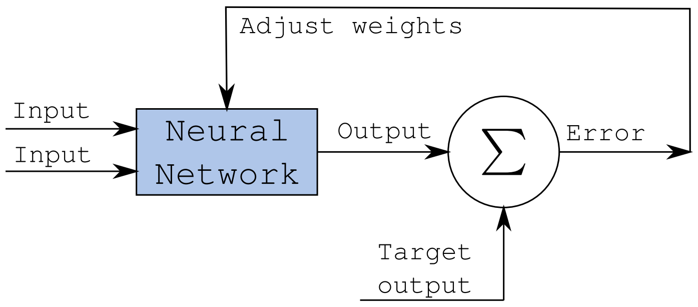


Figure 2.8: Training of NNs.

There are two types of learning: *supervised* and *unsupervised* learning. The former assumes that data for training contains values of inputs and their corresponding outputs.

At the beginning all weights are initialized randomly. In each training cycle the NN is fed up with inputs and after receiving the response from the network, the response is compared to the correct output and values of weights are adjusted. The Backpropagation algorithm calculates gradient of cost function with respect to weights. Gradient shows how fast the cost changes when the weights change. In order to optimize the cost function, the gradient is passed to the optimization method, which uses it to update weights [24].

Various methods of adjusting weights exist. One of most common is Backpropagation algorithm. It calculates a gradient of the cost function with respect to weights. Gradient is the vector whose components are the partial derivatives of weights. Gradient shows how fast the cost changes when the weights change. The gradient is input to the optimization method which uses it to update the weights, in an attempt to minimize the cost function [24].

The adaptation-rule works as follows: every time the partial derivative of the corresponding weight changes its sign, which indicates that the last update was too big and the algorithm has jumped over a local minimum, the update-value is decreased [57] by the factor η^-, where $\eta^- < 1$. If the derivative retains its sign, the update-value is slightly increased by a factor of η^+, where $\eta^+ > 1$. The update values are calculated for each weight in the above manner, and finally each weight is changed by its own update value, in the opposite direction of that weight's partial derivative, so as to minimize the total error function.

Important parameters of this learning method are *momentum* and *learning rate*. Momentum helps avoid local minimum. It simply adds a fraction of the previous weight update to the current one. The learning rate determines how much of the respective adjustment is applied to the old weight [49].

It may happen that after being learned NN may tend to overfitting that results in poor generalization. It means that the NN is overlearned - it predicts perfectly the results of training instances but cannot deal with new data.

Overlearning may occur due to excessive iterations, namely too big number of learning epochs. An *epoch* is a time during which all training instances are used once to update the weights.

The number of hidden layers is also important. In general, if the problem is simple, one or two hidden layers will suffice; but, as problems become more complex, more layers are required [48].

A *classification problem* requires data to be classified in two or more categories. The number of training instances for each category in a classification problem should be exactly the same. Training instance should cover the problem set uniformly.

Results of training depends on random initialization, which is some cases may not lead to sufficient training results.

2.4.4 Motivation to Use NN for TMS

Core of the TMS is calculation of trust metrics according to some function. The efficient expression of this function is not an easy task, but NNs are able to learn this dependencies based on the provided input/output pairs.

Moreover, NNs are able to cope with some degree of uncertainty. This is an advantage when constructing TMS in a WANET, since not all information can be available and moreover, some of the data can be wrong (e.g., provided by a malicious node).

Chapter **3**

Previous Results and Related Work

Security problems of WANETs have recently received a lot of attention [43, 18, 15]. Numerous methods for improving security in WANETs using the trust concept were proposed. The first part of this section describes TMSs, including the ones based on NNs. The second part is dedicated to applying TMSs for secure routing.

3.1 Trust Management Schemes

A considerable amount of methods to establish and maintain trust relationships in WANETs were proposed [4, 11, 25, 32, 42, 44, 51, 69, 73, 63, 12, 56, 27, 74, 75, 77]. Let us described some of them.

Zhang et al. in [77] proposed distributed and adaptive trust metrics. Direct trust is calculated using the communication trust and energy trust. Communication trust calculations are using the number of successful and unsuccessful data packets combined with predicted trust values using regression model. The energy trust calculation uses the residual energy of nodes. The recommendation trust is calculated using the recommendation reliability and the recommendation familiarity, that together with propagation distance gives indirect trust. Recommendation reliability is calculated as the difference from average recommendation trust. Recommendation familiarity is calculated from the number of successful data packets and the number of common neighbors.

Alnumay et al. [4] adopted the clustering framework. A node monitors traffic of its neighbors to calculate direct trust. Nodes that are heads of clusters initiate trust calculations and are responsible for trust propagation. Direct trust is calculated as a ratio of the number of good behaviors to the sum of numbers of good and bad behaviors. The resulting trust is calculated by the cluster head using ARMA/GARCH forecasting model to predict the behavior of the node based on its past behavior.

Yang et al. [75] proposed a decentralized TMS based on blockchain techniques. Roadside units collect ratings from vehicles, calculate trust of vehicles and pack them into blocks, which are added to the trust blockchain.

Guleng et al. [25] uses a fuzzy logic for direct trust calculations and reinforcement learning for indirect trust. They use asymmetric cryptography to authenticate the ID of the node that sent the packet.

The neighbor observing model based on watchdog mechanism is a basis for the trust management scheme proposed by Yang et al. [74]. Each node watches its neighbor node and marks its behaviors as α, β, and γ: the number of benevolent, malicious, and suspicious behavior, respectively, in a certain time period. The neighbor recommendation model accompanied with indirect trust value is used. Dempster-Shafer evidence theory is implemented to combine indirect trust values from different neighbors.

Biswas et al. [11] proposed that each node in a network is assigned three parameters: rank (measure of reliability of that node), remaining battery power, and stability. Trust value is calculated as their product. If the destination node has received data packets, it will send an acknowledgment to the source node, and then the ranks of the intermediate nodes will be incremented. If no acknowledgment is received within the timer period, the ranks of the intermediate nodes will be decremented. The stability factor is based on pause time and velocity: relative velocity with respect to the source node. The rank tables are periodically exchanged among the participating nodes.

Trust model presented by Xia et al. is based on fuzzy theory [73]. After finishing an interaction between two neighbour nodes, both sides will make a satisfactory evaluation of this interaction depending on the packet forwarding ratio. The protocol uses time stamps for computing the impact of packet forwarding ratio on trust evaluation. Direct trust of the particular time slot is an average of the satisfactory evaluations. The trust value consists of a weighted sum of direct trust, recommendation trust, some punishment of malicious or uncooperative entities and the level of activity of an entity in a network (based on the cumulative number of entities interacted with the evaluated node).

The method proposed by Jhaveri et al. in [32] is based on the AODV protocol complemented with an attack pattern discovery mechanism and trust model with packet dropping ratio as a metric. They have analyzed the effect of a threshold, trust weights, and update interval on PDR and routing overhead.

3.2 TMS with NNs

Most of the TMSs are based on calculations of trust metrics from experience of nodes, eavesdropping, or collecting information from the WANET. This information can be partial or it is challenging to determine which part of the calculation should get more weight. Therefore, some of the researchers adopt NNs for the evaluation of trust.

Several methods for TMS in Vehicular ad hoc networks (VANETs) based on NNs were proposed. Namely, deep reinforcement learning (Zhang et al. [76]) or Bayesian NNs (Eziama et al. [23]) were used for the computation of trust. In [76], information about each vehicle's location and forwarding ratio is collected by the centralized controller; that information serves as an input for a deep NN. In [23], authors combined deep learning with probabilistic modeling of trust computations.

In Kavitha et al. [39], a NN classifier with sigmoid activation function was used to identify an intruder node in mobile ad hoc networks (MANETs). Input for the classifier is provided by features extracted from neighboring nodes and based on the node's probability metric, reflecting the node's forwarding behavior.

Azmi et al. [6] suggested to use RBF-NNs in peer-to-peer (P2P) networks for evaluating recommendations received from neighbors to determine the trust level of a particular node. After the communication phase, NN weights are adjusted using the obtained result and the NN is trained and can be used for making decisions. Unlike in our approach, trust is not calculated using metrics but it is just a consequence of estimations from other nodes. Also, the learning is performed in the running network, which means that nodes have to use battery and computational resources to actually find untrusted nodes. In our method the learning process for the NN is performed on synthetically generated data while in [6] it is performed on data from a P2P network.

In Imana et al. [31] RBF-NNs are used to predict the node trust in MANETs, i.e., to discover a node that is about to start malicious activity in advance. Authors assume the existence of nodal attributes (time, battery power, type of encryption, the number of neighboring nodes, etc.) that define states of node: benign, vulnerable, compromised, malicious, and forgiveness state. The state defines the reputation score of a node (trust value). Node behavior is modeled heuristically. The main difference to our approach is that in [31] the estimation of trust of a node is done without observing the actual activity of the node.

3.3 Enhancing Routing with Trust

Methods to enhance ad hoc routing protocols with trust can be divided into two groups. The first one represents approaches that apply trust during the route discovery phase. Either the control messages are rejected based on the trust or they contain a special field in the header that collects the trust value of the path during the dissemination of the RREQ message. Examples of such approaches are Eissa et al.[22], Marchang and Datta [44], Venkanna et al. [69], Simaremare et al. [63], Hatzivasilis et al. [27]. The second group represents approaches that select routes according to their trust values stored in routing tables. Examples are Li et al. [42], Wang et al. [70], Pathan et al.[51]. Our approach falls into the first group.

The approach presented by Pathan et al. [51] aims to combine trusted routing with ensuring quality of service. To achieve this goal the proposed scheme selects forwarding node by considering channel quality, link quality, and residual energy. A node trust is calculated based on these QoS parameters and node forwarding behavior. During route discovery RREP messages from untrusted nodes are discarded.

The method of Beheshtiasl and Ghaffari [9] was designed for wireless sensor networks. The method is based on distances and makes use of geometry, the Dijkstra algorithm is used for finding shortest distance, and location of nodes are found by executing multidimensional scaling algorithm. Then fuzzy logic is used to calculate trust.

Hu et al. proposed a secure routing protocol based on deep learning [29]. Direct trust is calculated using forwarding behavior and then combined with recommendations. Trust of a path is calculated as a product of the trust values of the nodes in the path. Packet flows and their demands are an input of a NN. The output of the NN is the set of links satisfying the demands.

3.3.1 Trust-Based AODV Protocol Approaches

Aggarwal et al. [2] proposed a modified AODV routing protocol to implement trust model by introducing node trust table and packet buffer. A table is used to store information for neighbors and malicious nodes. Each node stores the node ID of its neighbor and calculates the trust value for that node based on packet observation. Authors have a slightly different approach to incorporate trust. Each node has its trust table, and when receiving a packet, it checks the trust value of the source. If it is untrusted, then the packet is dropped. The trust of a node gets updated on successful and failed transmission and the minimum value is set as 0 whereas the maximum trust value is set as 1. The threshold value is set to 0.5 and if any node has trust value less than 0.5, then the packet coming from that node is simply discarded.

For evaluation of trust Bar et al. in [8] count the number of packets received/sent at each node. Also the number of RREQ messages received and the number of RREP messages sent by each node is counted. Then the packet forwarding ability (the number of packets sent/the number of packets received) and weight factor, which is defined as the ratio of the number of RREP messages sent to the number of RREQ messages received by the node, are calculated. Trust value is calculated as a product of packet forwarding ability and weight factor. After this calculation, trust value is inserted into the routing table. The rest of the scheme is similar to the traditional AODV routing protocol, except that the most trusted path is selected.

Hazra and Setua proposed an context-aware trusted AODV protocol [28]. Trust evaluation is based on context-aware time dependent direct and indirect trusts. The direct trust is computed directly by a trustor (by sending control messages to a trustee, evaluating response time and then increasing or decreasing the trust value). Time is measured for acknowledgment of control packets or timeout is applied. For each acknowledgment the Trust manager stores direct trust equal to 0.9 and for the absence of acknowledgment it stores 0.1. On the other hand, indirect trust is the collaborative effect of recommendations and notifications. Indirect trust is computed as a weighted sum of them. Final trust is then a weighed sum of direct and indirect trust. Trusted On-demand Routing model is implemented on each node [28].

Thachil et al. proposed collaborative AODV [66]. When any node wishes to send messages to a distant node, it sends the RREQ message to all the neighboring nodes. A RREP message obtained from its neighbor is sorted by trust ratings. Every node monitors neighboring nodes and calculates trust values on its neighboring nodes dynamically as a ratio of the number of packets dropped to the number of packets to be forwarded by that neighboring node. If the trust value of a monitored node goes below a predefined threshold, then the monitoring node assumes it is malicious and removes that node from the route path. At the beginning, all neighboring nodes are given a range value of 0.5 by each node. The trust value is calculated based on the most recent set of packets transmitted to neighboring nodes. A node keeps a range value on all of its neighboring nodes. If the trust value is less than the threshold (0.3 used in this work), the range value is decremented according to the trust value until it reaches 0.0. If the trust value is above the threshold, the range value is incremented until it reaches 1.0. When the range value of a node goes below the threshold, it is considered to be malicious. Even if a node is not malicious, it may drop packets due to a broken link. In that case the node will send a RERR message to its previous node and precursors. That RERR message has been modified to include the unique identifier of the data packet dropped due to the broken link so that by other nodes this data packet will not be counted as dropped packet. This avoids the false positives. RERR messages from non-trusted nodes are discarded.

Distributed trust protocol called TERP is used in the work of Ahmed et al. [3] to detect and isolate misbehaving nodes. Trust, residual-energy, and hop count are used for making routing decisions. Trust is calculated as a weighted sum of direct and indirect trust. Direct trust is the ratio of correctly forwarded packets to the total number of received packets. Indirect trust is derived from the direct trusts of other nodes. The weight of the particular recommendation is determined by the direct trust of the recommending node. TERP extends AODV with its trust and energy information. TERP uses a composite routing function metric that is a weighted sum of trust, energy and hop count.

Li et al. in [42] proposed that routing tables of nodes contain trust value for each path. So, if there are more entries for some destination with the smallest hop count, the more trusted path is chosen. The trust of a path is calculated as the product of the node trusts along the path.

Eissa et al. [22] proposed a trust-based scheme for AODV which uses a friendship mechanism. Nodes keep lists of friends and their friendship values, calculated over some features. RREQ and RREP messages are rejected if friendship values of previous and next hops are below the friendship threshold. Also, a friendship of a route is evaluated, and the route is registered if it is more friendly than the existing one.

The method proposed in [44] by Marchang et al. changes the original neighbor table entries and routing table entries. Neighbor trust value is used to handle RREQ and RREP messages from that neighbor. Route trust is used to select the route. The reliability (trustworthiness) of a route depends on the reliability of all its nodes. The most trusted route is chosen. When two or more routes have the same trust value, the shortest path is chosen. A node uses packet forwarding behavior of a neighbor to evaluate its trust level. The trust value is calculated using only local information. The scheme uses time windows to ensure the trust information is up-to-date. While at some time window node may be untrusted, it may change the behavior and became trusted in the later time window.

Venkanna et al. [69] suggested the path trust value is computed during the spread of RREQ and RREP messages. Each node adds the trust value of the next hop while the control message is spread over the network. The destination node computes the average and that is defined as the trust of the path.

In Simaremare et al. [63], when a node receives a RREQ message, it checks the trust of the source. If the source is untrusted, the RREQ message is ignored. Otherwise, the trust of the neighbors is calculated and RREQ message is forwarded only to trusted ones. RREP messages are only forwarded.

3.3.2 Trust-Based DSR Protocol Approaches

Hammi et al. [26] assume that shared secrets are exchanged between nodes during an initialization phase. Trust is calculated as a weighted average of recommendations and observations. An observation is calculated using the number of successful or failed operations. Trust of a path is computed as a product of path edges trust. The protocol extends fields of the DSR RREQ message. Control messages are signed and their integrity is verified upon receiving. When transmitting a packet, the source node inserts the structure of trust multigraph into it. Trusted multigraph from source node S to destination node D is composed of two levels. First level is composed of nodes trusted by S. The second level is composed of untrusted or unauthenticated nodes. A forwarding node selects the outgoing edge based on source reliability requirement. When the requirement allows, the source diminishes some edge's weight to force forwarding nodes to utilize less forwarding paths [26]. When multigraph trust is below requirements, the protocol triggers a new route discovery.

The on-demand secure routing protocol called Ariadne presented in [30] by Hu et al. relies on symmetric cryptography and is based on DSR. Ariadne relies of the setting up of $n(n + 1)/2$ shared secret keys before the protocol can be used in network with n nodes. Then each node verifies the origin and integrity of routing data.

Buchegger and Le Boudec in [12], a trust scheme is presented to extend the DSR routing with detection and isolation of misbehaving nodes. When non-cooperative behavior is detected, the misbehaving node is excluded from routing.

Wang et al. [70] evaluate trust using the similarity of nodes. The similarity is a weighted sum of different node attributes, such as velocity or moving direction. The decision to forward the data is based on the similarity degree.

Pirzada et al. [56] suggested the trust mechanism be integrated into the route discovery process. Trust is a cumulative sum of the normalized values for different categories of behavior. In the DSR protocol, nodes add their IP addresses in RREQ messages. During the propagation of a RREP message, each node adds the trust of the preceding node.

SCOTRES system [27] was created by Hatzivasilis et al. for WANETs deployed for monitoring environmental parameters. The routing protocol uses three metrics to evaluate the node trust. Based on the information from the routing table, a topological metric of a node is evaluated, and the rating system will tolerate failures of significant nodes. Energy and channel-health are the other two metrics. Paths that contain malicious nodes are excluded. The method was integrated into the DSR protocol.

3.4 Overview of Existing Simulators

For verification of ideas and validation of proposed algorithms, a testbed is required. Since a real testbed is expensive and not always real hardware is required to verify ideas, many researchers use different network simulators. Let us analyze the most common network simulators.

The NS-2 simulator is an open-source and popular simulator, which brings such a benefit as a continuous introduction of new extensions. These allow support of additional functionality. However, this also increases complexity. The simulator is written as a solid-state application, without possibility to divide it into modules. Thus it supports the whole batch of functions, introduced in official releases and is backwards compatible. However, this gives the simulator redundant functionality. The simulator is designed for general purposes network simulation, it is not optimized for any particular type of network. The official package does not include statistical collection instruments, which makes it harder to collect and analyze the data.

The NS-3 simulator is not backwards compatible with NS-2. The existing simulation scenarios have to be rewritten, which lead many researches to use NS-2.

Other software such as QualNet and OPNET are commercial, which is the main drawback. Although QualNet was improved in comparison with its open-source predecessor (GloMoSim), it still has some drawbacks such as inability for a user to implement new custom modules. The drawbacks of OPNET are similar to that of the NS family, as it requires a user to be experienced with finite state automata and the Proto-C language.

The NEtwork Description (NED) is a special high-level programming language designed specifically for creation of simulation scenarios in the OMNeT++ simulator. It is a discrete event simulator. Although it takes time and effort to get familiar with the programming concept, once learned it becomes a powerful tool. We chose it for experiments as we are familiar with C++ used to write modules of simulation.

WANETs have limited support in the modern network simulators. They are mostly implemented with the help of additional packages, typically developed by enthusiastic open-source community members. These external packages are often insufficient, have little or no documentation, and are sometimes abandoned by the original authors. Also, there are no network simulators that have sufficient support for the concept of trust.

Chapter 4

Goals of the Dissertation Thesis

It is a matter of context when it comes to trust. For a WANET it is vital to detect nodes with low delivery ratio, as they degrade performance of the network. The problems addressed in this dissertation can be divided into two parts, study of trust concept in WANETs, and utilization of trust for routing.

Considering the review of the related work, the goal of this dissertation is to create a novel TMS that does not consume valuable resources of WANET nodes and is able to manage trust on-the-fly. The second goal is to integrate the proposed TMS into routing in a WANET without changing the routing protocol.

NNs are mathematical models of biological nervous systems used to estimate or approximate functions that depend on a large number of inputs that are generally unknown [61]. As NNs are suitable for solving problems with hidden dependencies based on the performed learning, we wanted to investigate the suitability of NNs for the detection of untrusted nodes in WANETs.

Information collected from a WANET can be incomplete or incorrect. In this case, trained NNs should be still able to estimate/predict trust values till some degree of correctness.

To start using NN, data instances for training are needed. The goal of this research is to create a way to minimize time needed to collect data for training and the related delay to minimum.

4.1 Novel Trust Management Scheme

Develop a NN-based TMS for trust estimation/detection of untrusted nodes in WANETs under the following assumptions:

- ○ The network is connected.
- ○ The physical layer assumptions are that transmission of frames between neighboring nodes is guaranteed. Links between nodes are bidirectional, which is a necessary requirement for many wireless Medium Access Control (MAC) protocols as they need to exchange of link-layer frames for collision avoidance and reliability [1].
- ○ WANETs are homogeneous, i.e. all nodes are identical.
- ○ WANETs have fixed topology, they are running for sufficiently long time. Therefore, we can construct the training data for the NNs synthetically.
- ○ If a WANET changes its topology, NNs will be retrained. So we can see the "before change" and "after change" WANET as two different networks.
- ○ Size of a WANET is expected to be up to 50 nodes.

The stages of development will be as follows:

1. Design appropriate NN architecture for detection of untrusted nodes/estimation of the trust value.
2. Generate synthetic training and testing data.
3. Train the NNs and adjust the training options.
4. Propose the performance metrics for the NN-based TMS.
5. Perform the experiments and analyze the NN-based TMS performance.
6. Evaluate the NN-based TMS properties.

In the rest of the dissertation we call our NN-based TMS shortly NeNTEA(Neural Network Trust Evaluation in Ad hoc networks).

4.2 Integration of Trust into a Standard Reactive Ad Hoc Routing Protocol

Standardized routing protocols for WANETs were created with the assumption that network nodes cooperate and can be trusted. There are various mechanism to enforce the cooperation in WANETs.

The first part of the dissertation is dedicated to the problem of node trust prediction or estimation in a certain class of WANETs.

The 2nd main contribution of the dissertation is the integration of the NeNTEA into the WANET routing protocol. The design goal of this integration is that the ad hoc routing protocol cannot change.

The method of enhancing reactive ad hoc routing protocol with trust should be sufficiently generic to be applied to a variety of standard reactive ad hoc routing protocols[1].

The stages of the integration wil be as follows:

- Design the method of NeNTEA integration into the ad hoc routing protocol.
- Propose the performance metrics.
- Perform experiments and analyze the method performance.
- Analyze the method parameters and properties.

In the rest of the dissertation we call the method of NeNTEA integration into a routing protocol shortly TARA (Trust-Aaware Reactive Ad hoc routing).

4.3 Use Case and Communication Model

The use case of the proposed solutions is a WANET of sensor nodes monitoring, measuring, and collecting data about environmental parameters. The communication model assumes that all nodes keep sending data periodically to a single node, called a *sink*. This is a typical case for wireless sensor networks that represent a special class of WANETs [18]. The real-world applications of such networks include monitoring of air pollution, agricultural monitoring, and more [37]. The common property of all these use cases is that there is always one dedicated node collecting data from other nodes that act as sensors. These sensors cannot communicate directly with the sink, they need to use intermediate nodes to deliver their messages to the sink. Each route from a node to the sink is traversed periodically so often that it does not expire.

There could be one sink, two sinks or we may generalize the communication pattern so that each node plays role of the sink. The latter communication pattern was used for the designing and performing experiments for NeNTEA.

This communication model was used to perform an experimental evaluation of our solution, but the solution is not limited to this model.

[1]It is worth noting that the majority of the related work are designed to certain protocols rather than being generic.

Chapter 5

The 1st Contribution: the NeNTEA Method

5.1 Design Considerations

Detection of untrusted nodes in a given WANET is a challenge. Untrusted nodes should not be used as intermediate nodes, because they are not reliable. Thus information about untrusted nodes should be incorporated in the routing protocol.

We recall that we defined trust of a node as the confidence that the node will forward data correctly. Thus its PDR is used as a metric of its trust.

There could be several reasons for a node to make errors and not to deliver data. The first one is traffic congestion. If a node is a part of many routes and traffic is intensive, it can be overloaded and not able to forward data. This problem can be solved by a better topology: create alternative routes to decrease the load from the most frequently used nodes. Several topology control algorithms exist to accomplish this task.

Another reason can be the node itself. Regardless of the traffic, a node can behave improperly due to faulty hardware or software, or the node can be malicious. This situation is hard to solve. This particular node should be detected and excluded from the network communication.

Each node can attempt to measure and share information about other nodes delivery ratio. As follows from the Chapter 3, approaches to measure packet delivery ratio include:

- eavesdropping,
- measurement by communication.

5.1.1 Eavesdropping

Due to the fact that wireless communication uses shared medium, each node can observe packets in its communication range, even if these packets are not destined to it. Thus, when a node sends a packet to its neighbor, it can eavesdrop if the neighbor forwards the packet further. Figure 5.1 shows that node N_1 and node N_3 cannot communicate directly, therefore they forward their data through node N_2.

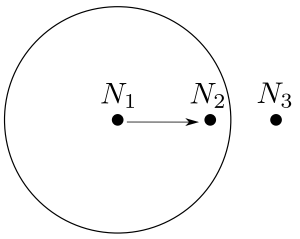


Figure 5.1: Communication range of node N_1.

Node N_2 is in the communication range of node N_1. Node N_1, after sending the packet, turns its network card to *promiscuous mode*. Promiscuous mode implies receiving all traffic even if it is not destined to this node, so node N_1 eavesdrops if node N_2 forwards the data (Figure 5.2). This implies extra energy and resource consumption.

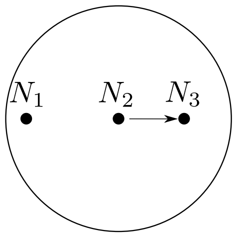


Figure 5.2: Situation 1: eavesdropping succeeds.

This seems to work, but let us show another example in Figure 5.3. Nodes N_1 and N_3 are again not in the direct communication range of each other and use node N_2 for relaying their data. But now by eavesdropping node N_1 will not obtain relevant information, as node N_2 adjusted its transmission power to be sufficient to send data to N_3, so node N_1 is not receiving this transmission. From node N_1's point of view, node N_2 did not forward this packet.

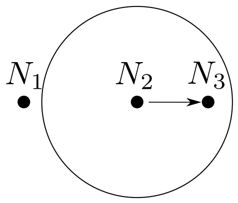


Figure 5.3: Situation 2: eavesdropping fails.

This is the first disadvantage of eavesdropping. Another one is the fact that being in the promiscuous mode, a node is not able to send any data (as radio transceivers operate in half-duplex). Thus promiscuous mode collides with methods responsible for regulating access to the shared medium (for example CSMA/CA), because such a node will not reply to Request to Send (RTS) messages.

5.1.2 Measurement of Communication

Another way to measure PDR of a node N_i is by measuring its communication (see Figure 5.4). The assumption is the existence of:

- alternative path aroung node N_i,
- extra protocol for negotiation of measurements.

In the situation in Figure 5.4, node N_1 and node N_2 want to perform measurement of node N_i. They need to agree about the $PDR(N_i)$ measurement and have to use some alternative route for that (for example through node N_3), which may not always exist. N_1 and N_2 should be running an extra protocol to negotiate parameters of measurement, control sending data for measurement and evaluation of results. Even if this is achieved, there is another difficulty: to ensure an appropriate environment for measuring. In particular, there should not be any other traffic in this part of the network during measurement process. This is hard, if not impossible, to guarantee.

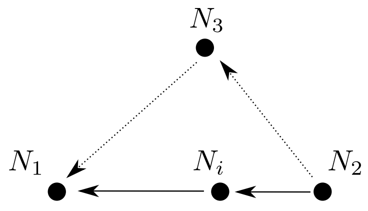


Figure 5.4: Measuring communication.

5.1.3 Our Proposal

Due to the fact that WANET nodes are powered by batteries, the TMS should not increase power consumption on the nodes. We want to achieve energy neutral TMS, thus will not consider eavesdropping or methods with active measurements of communication.

Proposal We propose to determine PDRs of nodes from the PDRs of paths in which the node participates.

Hence, the design criterion of our work was to develop a method for evaluation of node PDRs from the knowledge of paths and their PDRs. PDRs of paths can be obtained with no extra cost just by gathering the information from the traffic on-the-fly. The trust estimation of an investigated node from these information (data) is a nontrivial problem to be solved.

Definition 5.1.1. Problem $\mathcal{T}$: given network communication arrangement (according to Definition 2.1.4) and PDRs of the paths, estimate the PDRs of nodes.

The example of the problem is shown in Figure 5.5. To estimate the PDR of the intermediate node N_5, the PDR path values (these values were measured by path source nodes) are given in Table 5.1.

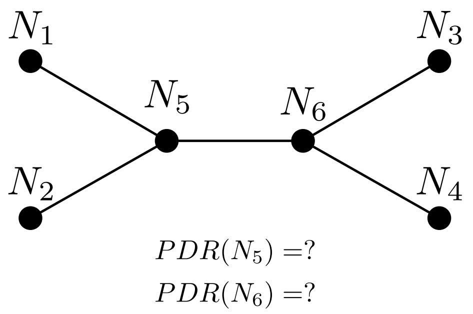


Figure 5.5: An example of a small WANET topology.

Table 5.1: All paths in the example topology with node N_5 as intermediate.

Path P_j	Path nodes	$PDR(P_j)$
P_1	$\langle N_1, N_5, N_2\rangle$	0.4
P_2	$\langle N_1, N_5, N_6\rangle$	0.4
P_3	$\langle N_1, N_5, N_6, N_3\rangle$	0.36
P_4	$\langle N_1, N_5, N_6, N_4\rangle$	0.36
P_5	$\langle N_2, N_5, N_1\rangle$	0.4
P_6	$\langle N_2, N_5, N_6\rangle$	0.4
P_7	$\langle N_2, N_5, N_6, N_3\rangle$	0.36
P_8	$\langle N_2, N_5, N_6, N_4\rangle$	0.36
P_9	$\langle N_3, N_6, N_5, N_1\rangle$	0.36
P_{10}	$\langle N_3, N_6, N_5, N_2\rangle$	0.36
P_{11}	$\langle N_4, N_6, N_5, N_1\rangle$	0.36
P_{12}	$\langle N_4, N_6, N_5, N_2\rangle$	0.36
P_{13}	$\langle N_6, N_5, N_1\rangle$	0.4
P_{14}	$\langle N_6, N_5, N_2\rangle$	0.4

The observations made with the example raise the following questions:

- Does each path PDR have exactly the same impact on a node PDR?
- If not, what will be the weight of individual path?

The problem $\mathcal{T}$ consists of two subproblems:

1. An estimation of a the trust value of a specific node.
2. Detection (making decision) whether a given node is trusted.

If we have method to solve $\mathcal{T}_1$ then $\mathcal{T}_2$ is solved for free by applying a threshold on the estimated trust value.

5.1.4 Definition of the Problem $\mathcal{T}$ of Trust Estimation

Let us consider a WANET with the following parameters:

1. set of nodes $\mathcal{N} = \{N_1, N_2, N_3, ..., N_n\}$, where n is a number of nodes in the network: $n = |\mathcal{N}|$;
2. communication pattern π as a list of ordered pairs (N_s, N_d), where $N_s \in \mathcal{N}$ is a source node and $N_d \in \mathcal{N}$ is a destination node;
3. routing protocol that finds a single path from N_s to N_d.

Routing protocol taking π provides a set of paths $\mathcal{P} = \{P_1, P_2, P_3, \ldots, P_p\}$, where $p = |\mathcal{P}|$ is the number of paths.

Let us describe paths in $\mathcal{P}$ as $P_k = \langle N_{k_1} N_{k_2} ... N_{k_{l_k}} \rangle$, where $k \in \{1, 2, \ldots, p\}$ and $\{N_{k_1}, N_{k_2}, ..., N_{k_{l_k}}\} \subset \mathcal{N}$. Hence P_k has l_k nodes and $PL(P_k) = l_k - 1$ (Definition 2.1.2). Using (2.4) for path P_k we get the equation:

$$PDR(P_k) = \prod_{i=2}^{l_k-1} PDR(N_{k_i}) = PDR(N_{k_2}) \times PDR(N_{k_3}) \times ... \times PDR(N_{k_{l_k-1}}) \tag{5.1}$$

Having information about paths and their PDRs, we get a system of equations:

$$\begin{aligned}
&PDR(P_1) = PDR(N_{1_2}) \times PDR(N_{1_3}) \times ... \times PDR(N_{1_{l_1-1}}) \\
&PDR(P_2) = PDR(N_{2_2}) \times PDR(N_{2_3}) \times ... \times PDR(N_{2_{l_2-1}}) \\
&... \\
&PDR(P_p) = PDR(N_{p_2}) \times PDR(N_{p_3}) \times ... \times PDR(N_{p_{l_p-1}})
\end{aligned} \tag{5.2}$$

Note that a particular node N_{k_i} from path P_k can be present in more paths, just at different positions. However, $\forall N_i \in \mathcal{N} : PDR(N_i)$ is the same no matter which path node N_i belongs to.

The problem is to solve this set of equations with n unknowns, $PDR(N_i)$, given $PDR(P_k), k \in \{1, 2, \dots, p\}$. A standard algorithmic solution of the set of logarithmized equations (5.2) is not possible for several reasons. First, PDR is not a static number. It is changing with time.

Moreover, the data can be incomplete and PDR of some path can be missing, or incorrect (some node may report intentionally incorrect PDRs of paths it is the source of). In such cases, the mathematical equations will not hold, and therefore may have no mathematically exact solution. However, instead of the algebraic solution of (5.2, we may use NNs since their the essential feature is an ability to generalize.

The main idea of our approach is that a NN can assist in searching for dependencies between PDR of paths and PDR of nodes, thus, helping in trust estimation of a particular node.

Backpropagation NNs can naturally solve two types of problems - classification and regression. The former problem is to determine if a given node is trusted or untrusted. The later is the estimation of the PDR value of every node in a given network.

5.2 The NeNTEA Method Design

The input data for the NeNTEA method: $PDR(P_j), P_j \in \mathcal{P}, j \in \{1, 2, \dots, p\}$.

Goal The goal is to design a TMS that satisfies the assumptions stated in Section 4.1 and is able to evaluate trust of nodes according to Definition 2.2.2 using the submitted inputs. Design is based on our proposal stated in Section 5.1.3.

5.2.1 Design

Basic requirement for a supervised NN learning process is a sufficient number of input and target value pairs.

From the proposal 5.1.3 it follows that the inputs of a NN are the PDRs of paths and the output, based on two subproblems, is either answer whether the investigated node is trusted, or an estimated PDR value of the investigated node. In the first case the output is binary, true or false. In the second case the output is a real number between 0 and 1 representing the predicted trust value.

Definition 5.2.1. Given a node $N_i \in \mathcal{N}$. $\mathcal{P}(N_i) \subset \mathcal{P}$ is a subset of paths where N_i is an intermediate node.

Each node N_i in a WANET that forwards data of other nodes participates in a particular set of paths $\mathcal{P}(N_i)$, and its $PDR(N_i)$ contributes to the PDRs of the paths from $\mathcal{P}(N_i)$. Another investigated node N_j participates in another set of paths $\mathcal{P}(N_j)$, and even if

$|\mathcal{P}(N_i)| = |\mathcal{P}(N_j)|$, each path may have different weight on the output. Thus, for each investigated intermediate node $N_i, i \in \{1, 2, \ldots, n\}$, a particular NN_i is constructed and trained.

Definition 5.2.2. Let us denote $X(NN_i)$ the vector of the input values of NN_i. The size of the *input vector* is denoted by $\gamma_i = |X(NN_i)|$. Let us denote $Y(NN_i)$ the vector of the output values of NN_i. The size of $Y(NN_i)$ is 1. $X(NN_i)$ and the corresponding $Y(NN_i)$ create a *training instance*. The *training matrix* $\mathcal{X}(NN_i)$ is a set of training instances.

To perform the learning process of a NN_i $\mathcal{X}(NN_i)$ is needed. We assume the number training instances $m = |\mathcal{X}(NN_i)|$ will be the same for all i. The size of the training matrix $\mathcal{X}(NN_i)$ is therefore $m \times (\gamma_i + 1)$.

Definition 5.2.3. Let $\mathcal{S}(\mathcal{P}(N_i))$ denote the set of source nodes N_s of paths in $\mathcal{P}(N_i)$. Let $\lambda_i = |\mathcal{S}(\mathcal{P}(N_i))|$ denote the number of such source nodes.

Definition 5.2.4. A path where N_s is a source and N_i belongs to that path as an intermediate node is denoted as $P_s(N_i)$. The set of such paths is denoted as $\mathcal{P}_s(N_i)$. For a given $N_s \in \mathcal{S}(\mathcal{P}(N_i))$, let $\mu_{s,i} = |\mathcal{P}_s(N_i)|$.

From the assumptions from in Section 4.1 two approaches for the synthetic construction of training instances follows.

5.2.1.1 Approach 1

Algorithm 5.1 describes a straightforward approach. Given N_i, each input vector is represented by the PDR of paths in $\mathcal{P}_i$. At first, PDRs of all intermediate nodes in the network are randomly generated. Using them and using the knowledge of the network communication arrangement, we calculate PDRs of the paths according to (2.4). These PDRs produce one input vector, the output is the generated $PDR(N_i)$. This pair of input and output vectors crates one training instance.

It follows from [67] that our way of data construction will provide values close to those collected from real traffic over a sufficiently long running time. Thus, the synthetic datasets can model real data traffic.

Let us summarize that in this approach the number of inputs of a NN_i for detection/estimation of $PDR(N_i)$ is equal to the total number of paths where N_i is an intermediate node. In general, $\gamma_i \leq p$ and $p \leq n(n-1)$. This number of inputs may be very large in practice, even if $10 \leq n \leq 50$. Therefore, we have developed another approach, see Algorithm 5.2.

Algorithm 5.1 Straightforward generation of training matrix $\mathcal{X}(NN_i)$ for all NN_i in a particular WANET

```
Input: $\mathcal{P} = \{P_1, P_2, P_3, ..., P_p\}$; $m$
Output: $\mathcal{X}(NN_i), i \in \{1, 2, \ldots, n\}$
  for all $N_i$ do                                        ▷ for each intermediate node
    $\gamma_i = |\mathcal{P}(N_i)|$                       ▷ calculate number of NN inputs
    $x = 1$                                               ▷ initialize index
    for all $P \in \mathcal{P}(N_i)$ do
      $paths[x] = P$; increment $x$                       ▷ create an array of $\gamma_i$ paths containing $N_i$
    end for
    for all $l \in \{1, 2, \ldots, m\}$ do                ▷ for each input vector
      for all $j \in \{1, 2, \ldots, n\}$ do
        randomly initialize $PDR(N_j)$
      end for
      for all $k \in \{1, 2, \ldots, \gamma_i\}$ do       ▷ for each input
        calculate $PDR(paths[k])$
        $\mathcal{X}(NN_i)[l][k] = PDR(paths[k])$         ▷ write element of input vector
      end for
      $\mathcal{X}(NN_i)[l][\gamma_i + 1] = PDR(N_i)$     ▷ write output vector
    end for
    return $\mathcal{X}(NN_i)$                            ▷ return training matrix
  end for
```

Algorithm 5.2 Generation of training matrix $\mathcal{X}(NN_i)$ for all NN_i in a particular WANET

```
Input: $\mathcal{P} = \{P_1, P_2, P_3, ..., P_p\}$; $m$
Output: $\mathcal{X}(NN_i), i \in \{1, 2, \ldots, n\}$
  for all $N_i$ do                                        ▷ for each intermediate node
    $\gamma_i = \lambda_i = |\mathcal{S}(\mathcal{P}(N_i))|$   ▷ calculate number of NN inputs
    for all $l \in \{1, 2, \ldots, m\}$ do                ▷ for each input vector
      for all $j \in \{1, 2, \ldots, n\}$ do
        randomly initialize $PDR(N_j)$
      end for
      for all $s \in \{1, 2, \ldots, \gamma_i\}$ do       ▷ for each source node
        for all $t \in \{1, 2, \ldots, \mu_{s,i}\}$ do    ▷ there are $\mu_{s,i}$ such paths $P_s(N_i)$
          calculate $PDR(P_s(N_i)[t])$
        end for
        $\mathcal{X}(NN_i)[l][s] = \frac{\sum_{t=1}^{t=\mu_{s,i}}(PDR(P_s(N_i)[t]))}{\mu_{s,i}}$   ▷ write element of input vector
      end for
      $\mathcal{X}(NN_i)[l][\gamma_i + 1] = PDR(N_i)$     ▷ write output vector
    end for
    return $\mathcal{X}(NN_i)$                            ▷ return training matrix
  end for
```

5.2.1.2 Approach 2

The second approach proposes to group the inputs by the source nodes of the paths, see Algorithm 5.2. In this case the maximum number of inputs $max(\gamma_i) = \lambda_i \leq n - 1$, see Definition 5.2.3. Each input of a NN_i is calculated as an average of PDRs of paths that start in node N_s and contain node N_i. This approach is based on the assumption of evenly distributed communication load among paths.

5.2.2 Example of Data Instance Calculations

Assume the example in Figure 5.5.

Table 5.2: All paths in the topology from Figure 5.5 with node N_5 as intermediate.

Path	N_s	Path nodes	$PDR(P_i)$	$X(NN_5)[s]$
P_1	N_1	$\langle N_1, N_5, N_2\rangle$	0.4	0.38
P_2	N_1	$\langle N_1, N_5, N_6\rangle$	0.4	
P_3	N_1	$\langle N_1, N_5, N_6, N_3\rangle$	0.4 * 0.9 = 0.36	
P_4	N_1	$\langle N_1, N_5, N_6, N_4\rangle$	0.4 * 0.9 = 0.36	
P_5	N_2	$\langle N_2, N_5, N_1\rangle$	0.4	0.38
P_6	N_2	$\langle N_2, N_5, N_6\rangle$	0.4	
P_7	N_2	$\langle N_2, N_5, N_6, N_3\rangle$	0.4 * 0.9 = 0.36	
P_8	N_2	$\langle N_2, N_5, N_6, N_4\rangle$	0.4 * 0.9 = 0.36	
P_9	N_3	$\langle N_3, N_6, N_5, N_1\rangle$	0.9 * 0.4 = 0.36	0.36
P_{10}	N_3	$\langle N_3, N_6, N_5, N_2\rangle$	0.9 * 0.4 = 0.36	
P_{11}	N_4	$\langle N_4, N_6, N_5, N_1\rangle$	0.9 * 0.4 = 0.36	0.36
P_{12}	N_4	$\langle N_4, N_6, N_5, N_2\rangle$	0.9 * 0.4 = 0.36	
P_{13}	N_6	$\langle N_6, N_5, N_1\rangle$	0.4	0.4
P_{14}	N_6	$\langle N_6, N_5, N_2\rangle$	0.4	

At first PDR values of nodes N_5 and N_6 are randomly generated: $PDR(N_5) = 0.4$ and $PDR(N_6) = 0.9$. Then the PDRs of the paths are calculated using formula (2.4). If we apply the straightforward approach from Algorithm 5.1, we need a NN_5 with 14 inputs to detect if node N_5 is trusted. This would increase the complexity of the learning process. This number of inputs are reduced by averaging the path PDRs containing the investigated node N_5 for each source node N_s (column $X(NN_5)[s]$ in Table 5.2), see Algorithm 5.2. In our example, 14 paths are grouped in 5 groups by source nodes. Input vector is calculated by taking the arithmetic means of the PDRs of constituent paths.

Table 5.3: All paths in the topology from Figure 5.5 with node N_6 as intermediate.

Path	N_s	Path nodes	$PDR(P_i)$	$X(NN_6)[s]$
P_1	N_1	$\langle N_1, N_5, N_6, N_3\rangle$	0.4 * 0.9 = 0.36	0.36
P_2	N_1	$\langle N_1, N_5, N_6, N_4\rangle$	0.4 * 0.9 = 0.36	
P_3	N_2	$\langle N_2, N_5, N_6, N_3\rangle$	0.4 * 0.9 = 0.36	0.36
P_4	N_2	$\langle N_2, N_5, N_6, N_4\rangle$	0.4 * 0.9 = 0.36	
P_5	N_3	$\langle N_3, N_6, N_5\rangle$	0.9	0.63
P_6	N_3	$\langle N_3, N_6, N_4\rangle$	0.9	
P_7	N_3	$\langle N_3, N_6, N_5, N_1\rangle$	0.9 * 0.4 = 0.36	
P_8	N_3	$\langle N_3, N_6, N_5, N_2\rangle$	0.9 * 0.4 = 0.36	
P_9	N_4	$\langle N_4, N_6, N_5\rangle$	0.9	0.63
P_{10}	N_4	$\langle N_4, N_6, N_3\rangle$	0.9	
P_{11}	N_4	$\langle N_4, N_6, N_5, N_1\rangle$	0.9 * 0.4 = 0.36	
P_{12}	N_4	$\langle N_4, N_6, N_5, N_2\rangle$	0.9 * 0.4 = 0.36	
P_{13}	N_5	$\langle N_5, N_6, N_3\rangle$	0.9	0.9
P_{14}	N_5	$\langle N_5, N_6, N_4\rangle$	0.9	

Table 5.4: Input vectors of NN_5 and NN_6 with $PDR(N_5) = 0.4$ and $PDR(N_6) = 0.9$.

Investigated node	Input vector				
N_5	$X(NN_5)[1]$	$X(NN_5)[2]$	$X(NN_5)[3]$	$X(NN_5)[4]$	$X(NN_5)[5]$
	0.38	0.38	0.36	0.36	0.4
N_6	$X(NN_6)[1]$	$X(NN_6)[2]$	$X(NN_6)[3]$	$X(NN_6)[4]$	$X(NN_6)[5]$
	0.36	0.36	0.63	0.63	0.9

The change of the node PDR results in changes in some inputs depending on the underlying topology. The examples of instances created on the same topology with different PDRs can be seen in Tables 5.5 and 5.6. An algorithm that covers all potential topology configurations and PDR values is hard to construct.

Table 5.5: Input vectors of NN_5 and NN_6 with $PDR(N_5) = 0.3$ and $PDR(N_6) = 0.8$.

Investigated node	Input vector				
N_5	$X(NN_5)[1]$	$X(NN_5)[2]$	$X(NN_5)[3]$	$X(NN_5)[4]$	$X(NN_5)[5]$
	0.27	0.27	0.24	0.24	0.3
N_6	$X(NN_6)[1]$	$X(NN_6)[2]$	$X(NN_6)[3]$	$X(NN_6)[4]$	$X(NN_6)[5]$
	0.24	0.24	0.52	0.52	0.8

Table 5.6: Input vectors of NN_5 and NN_6 with $PDR(N_5) = 0.2$ and $PDR(N_6) = 0.7$.

Investigated node	Input vector				
N_5	$X(NN_5)[1]$	$X(NN_5)[2]$	$X(NN_5)[3]$	$X(NN_5)[4]$	$X(NN_5)[5]$
	0.17	0.17	0.14	0.14	0.2
N_6	$X(NN_6)[1]$	$X(NN_6)[2]$	$X(NN_6)[3]$	$X(NN_6)[4]$	$X(NN_6)[5]$
	0.14	0.14	0.42	0.42	0.7

5.2.3 Architecture of NN_i for Detection of Untrusted Nodes/Estimation of Trust Value

The construction corresponds to Algorithm 5.2.

The schema of NN_i is represented in Figures 5.6 and 5.7. $\mathcal{X}(NN_i)[l][1]$ means the first element of the l^{th} input vector. The output for the $\mathcal{T}_1$ is True or False (Figure 5.6). The output for the $\mathcal{T}_1$ is the PDR value of the investigated node (Figure 5.7).

5.2.4 Training

For a given NN_i set of the $X(NN_i)$ from $\mathcal{X}(NN_i)$ is applied to the NN_i inputs. For each $X(NN_i)$ the output is received, compared with the target output $Y(NN_i)$ and values of weights are adjusted.

5.2.5 Trust Evaluation using NeNTEA

Sequence of steps to perform trust evaluation in a given WANET Using NeNTEA:

1. Collect the network communication arrangement, according to Definition 2.1.4;
2. Generate the data for training using Algorithm 5.2;
3. Train NNs as described in Section 5.2.4;
4. Collect the information about path PDRs from the WANET: for given $i, i \in \{1, 2, \dots\}$ the source nodes N_s provide the average of the PDR values of paths in $\mathcal{P}_s(N_i)$;

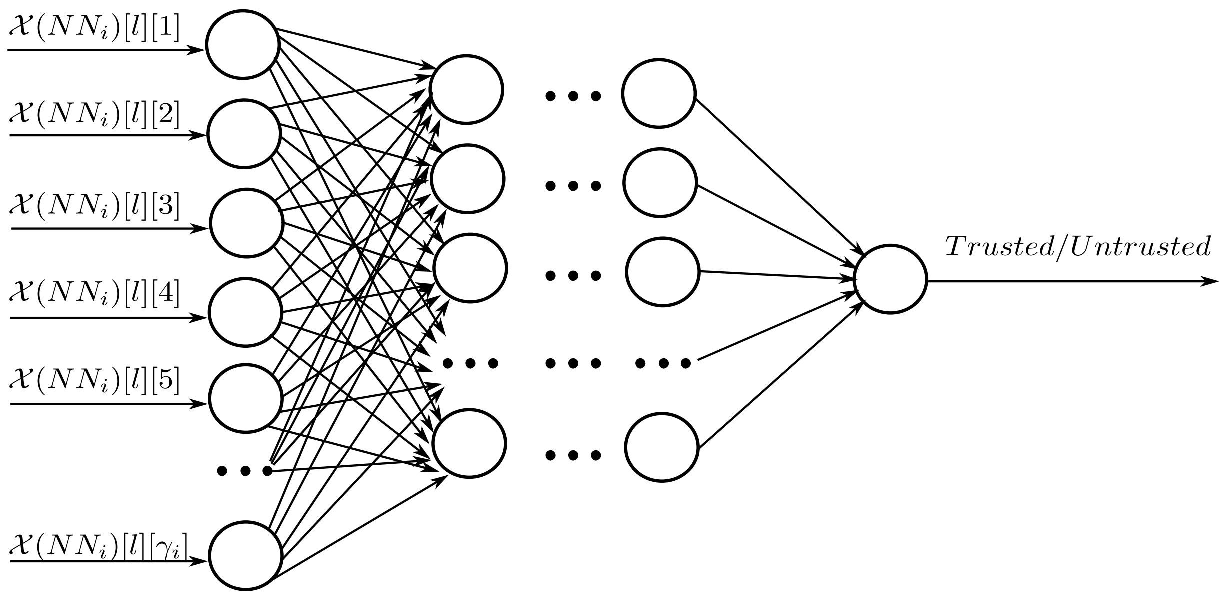

$\mathcal{X}(NN_i)[l][1]$
$\mathcal{X}(NN_i)[l][2]$
$\mathcal{X}(NN_i)[l][3]$
$\mathcal{X}(NN_i)[l][4]$
$\mathcal{X}(NN_i)[l][5]$
$\mathcal{X}(NN_i)[l][\gamma_i]$
$Trusted/Untrusted$

Figure 5.6: Architecture of NN_i for the detection if node N_i is trusted.

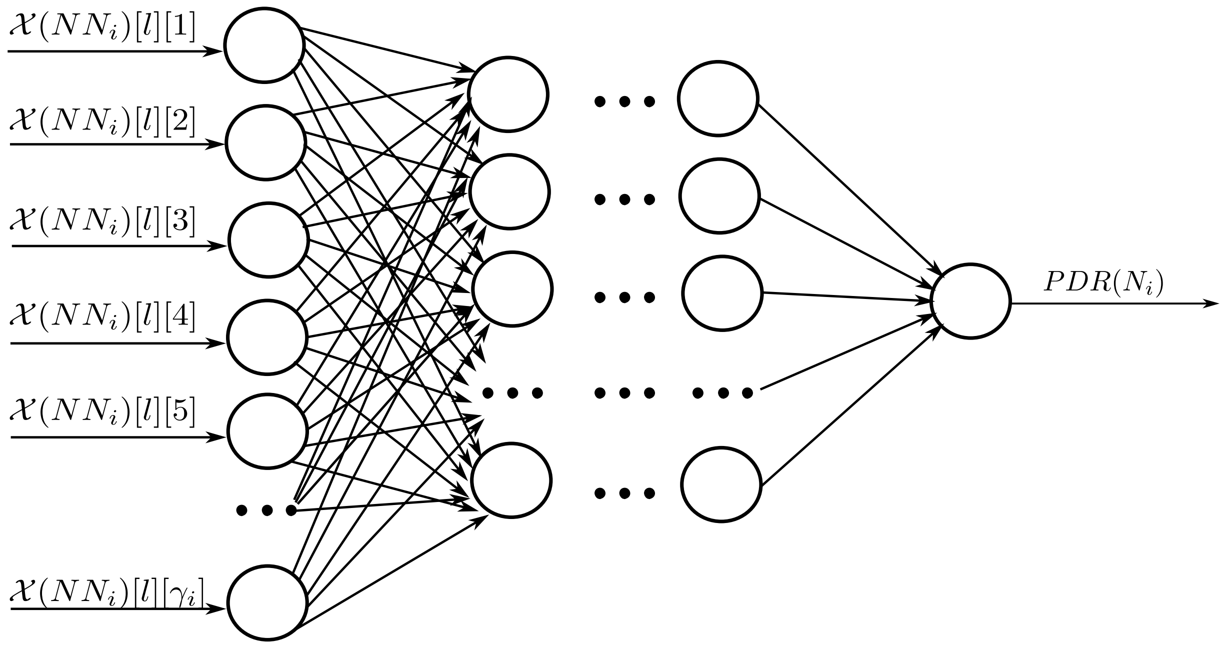

$\mathcal{X}(NN_i)[l][1]$
$\mathcal{X}(NN_i)[l][2]$
$\mathcal{X}(NN_i)[l][3]$
$\mathcal{X}(NN_i)[l][4]$
$\mathcal{X}(NN_i)[l][5]$
$\mathcal{X}(NN_i)[l][\gamma_i]$
$PDR(N_i)$

Figure 5.7: Architecture of NN_i for the estimation of node N_i's trust value.

5. Apply the trained NNs on the collected information to detect/estimate PDRs of nodes of the WANET. The collected average PDRs of paths in $\mathcal{P}_s(N_i)$ are used as input vector to the trained NN_i and the output of NN_i is taken as estimation of $PDR(N_i)$ data to get the result.

5.3 NeNTEA Simulation and Experiments

5.3.1 Simulation environment

Our experiments consist of 3 steps. (1) Simulation of WANETs to generate data for training and testing of NNs. (2) NNs training. (3) An application of the trained NNs on the testing data and calculation of performance metrics. Figure 5.8 depicts this process.

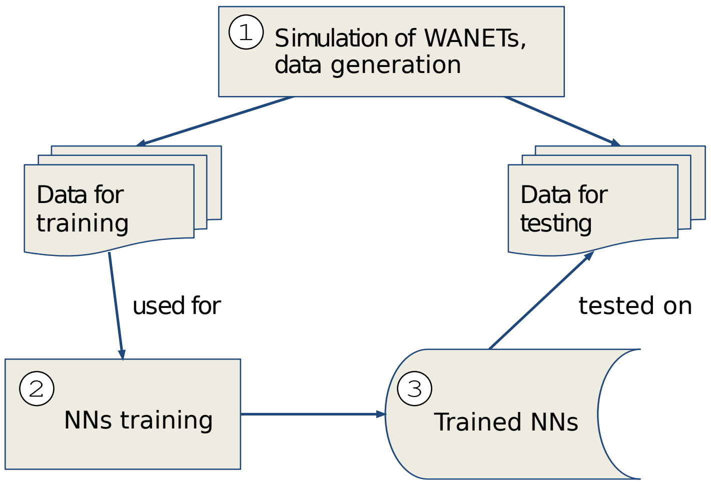


Figure 5.8: Steps in the NeNTEA experiments.

Simulation of WANETs including generation of WANET topologies and generation of training matrices for experiments was done using OMNeT++ 4.6 simulator and its INET framework.

For our purposes the simulation in graphical runtime environment is not needed. The simulator has command-line user interface for simulation execution. It runs on Windows, Linux, Mac OS X, and other Unix-like systems. The NeNTEA experiments were performed in Ubuntu (14.04, 16.04, and 18.04).

After the topologies and paths are generated by OMNeT++, the NNs training follows. The FANN 2.2.0 (Fast Artificial Neural Network) library is a free open source neural network library that implements multilayer artificial neural networks in C [48]. Several different activation functions are implemented in it. The library has bindings for PHP, Python, and Mathematica and the it also became accepted in the Debian Linux distribution [47]. We created an application that allowed to set all important parameters by command line arguments.

5.3.1.1 OMNeT++ simulation

We were considering WANETs of 20 nodes. This size is big enough for creating relevant problems, yet small enough to carry a large number of simulations and to verify the relevance of experimental results.

An example of a WANET created for the simulation is presented in Figure 5.9. Position of nodes were randomly generated within an area of $600m \times 600m$. Each node has an omnidirectional antena with range $250m$.

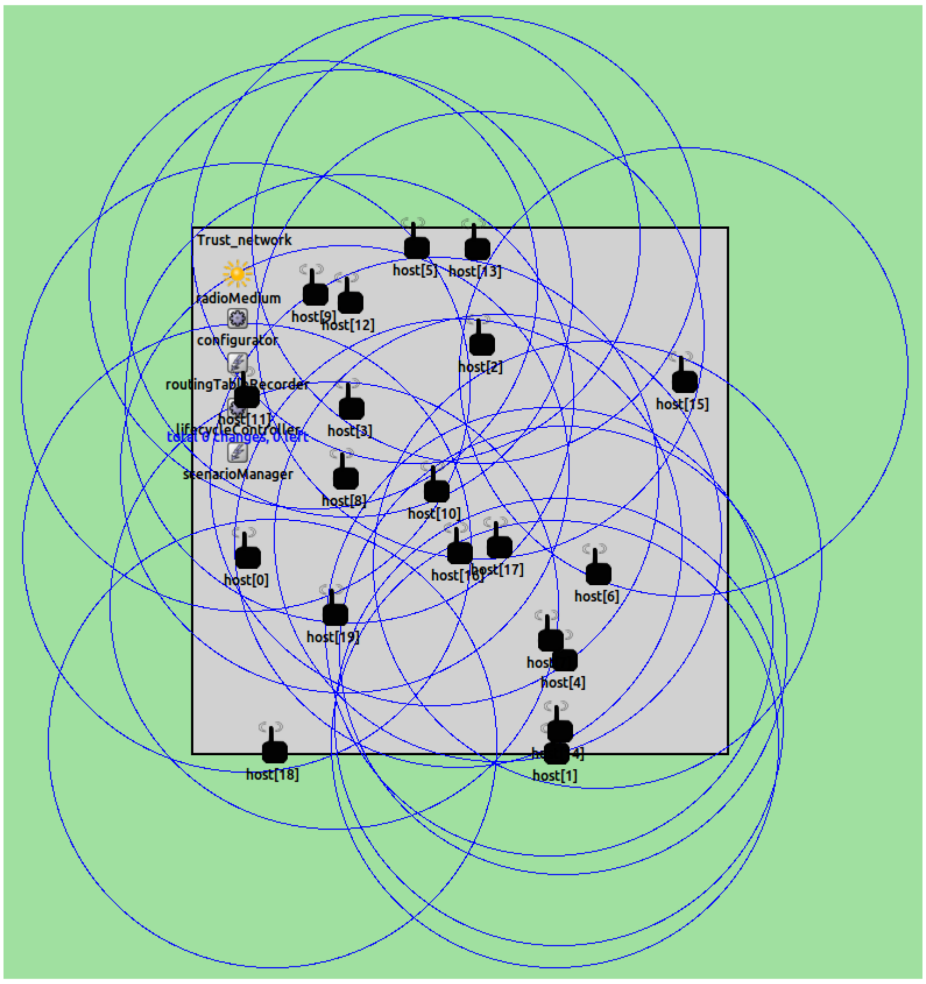


Figure 5.9: Structure of WANET in OMNeT++ simulation.

For the NeNTEA method any routing protocol can be used. In our particular implementation, the AODV algorithm was used for routing, as this is one of the most popular ad hoc routing protocols [59]. Moreover, its implementation in the OMNeT++ simulator is based on RFC 3561.

We assume that simulated WANETs are connected. Communication pattern assumes that each node communicates periodically with all other nodes in the network. The path from A to B is not equal to the path from B to A. Therefore, $p = n(n-1)$.

The structure of a node is presented of Figure 5.10. It has an omnidirectional antenna (see Figure 2.2) and two interfaces: wireless NIC (wlan[0]) and loopback interface(lo0).

The status module keeps information if the node is up. The mobility module takes care about node's position. In our simulation nodes are not moving and initial positions are

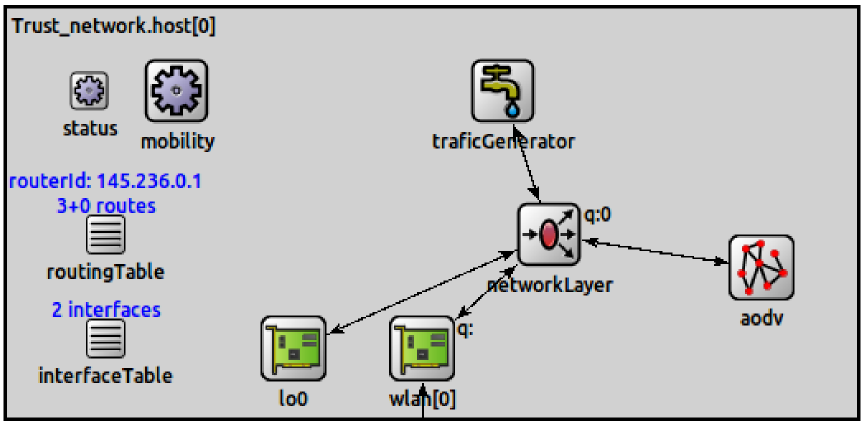


Figure 5.10: Structure of node in OMNeT++ simulation.

randomly generated.

The routingTable stores data about routes: destination, next hop, interface, source (manual, routing protocol), administrative distance, metric and routing protocol specific data.

The networkLayer performs routing with the help of the AODV module. When it receives a packet from the higher layer, it makes a request to the routingTable, and if the route exists, adds control information to the packet and sends it to NIC. If the route to the destination was not found in the routing table, the networkLayer sends a request to AODV to find ta route. When AODV finds a route, the packet can finally be forwarded to the lower layer. The AODV module was adjusted in such a way that after finding a route to the destination, it inserts an entry to the routing table, sets the route expiration time to a number large enough for the route to be valid till the end of the simulation run since fixed WANET topology is assumed.

The main module of a node that implements simulation logic is the trafficGenerator. First it sends probe packets to all other nodes to discover the topology. As a result of this, the routing table of each node should contain routes to all other nodes, specifically addresses of the next hop nodes on the route to the destinations.

At this step there is a check whether the constructed topology is connected, meaning that each node has routes to all other nodes. If it happens that the topology is disconnected, it is discarded and next attempt to create topology follows.

The next step is to collect information about paths, i.e. identify all intermediate nodes. After this step is accomplished, we have collected all information needed to perform data generation.

5.3.1.2 Experiment setup

Five datasets were created, each containing 100 different WANET topologies. Each intermediate node represents one *problem instance* with one *instance file* for which one NN will be constructed and trained. For every problem instance 1000 training data instances and

4000 testing data instances were constructed. In 50% of data instances the investigated node is trusted, and in other 50% of data instances the investigated node is untrusted.

The threshold for trust should be set in the context of the network traffic. In our experiments we set the trust threshold $\tau = 0.5$. Later we performed experiments for analyzing the influence of threshold value on the performance metrics of the NeNTEA method (Section 5.4.1).

As it was mentioned in Section 2.4.3 the set of training instances should uniformly cover the problem set. Data instances of particular problem are created by calculation of node PDRs combinations according to the network communication arrangement. In more details this is described in Section 5.2.1. The sufficient set of training instances can be obtained by random generation of PDR values. The aim is to cover the whole set of possible values. For the PDR values generation uniform and normal distributions were chosen.

As experiments with some datasets provided similar results, we will describe only three datasets here. The rest two datasets and experiments with them are described in Appendix A. All datasets were constructed for the same 100 WANET topologies with 20 nodes each. Datasets differ in node PDRs generation:

Dataset 1, ***Uniform-one:*** represents WANET with one untrusted node. Node PDRs are generated using uniform distribution.

Dataset 2, ***Uniform-all:*** represents WANET where all nodes can be untrusted. Node PDRs are generated using uniform distribution.

Dataset 3, ***Normal-one:*** represents WANET with one untrusted node (similar to Dataset 1), but node PDRs are generated using normal distribution.

Instance files for each dataset are generally of four types:

1. used for training NNs to solve the detection problem;

2. used for training NNs to solve the PDR estimation problem;

3. used for testing of the trained NNs to solve the detection problem;

4. used for testing of the trained NNs to solve the PDR estimation problem.

All of them are text files, created by a script that converts data files constructed by OMNeT++ to the proper format required by training/testing and detection/estimation.

Instance files used for training contain information about the number of data instances, the number of inputs and outputs of NNs. Outputs for detection problem are binary, whereas outputs for estimation are PDR values of the investigated nodes.

Instance files for testing have a simpler format. Each row is one data instance, inputs are followed with the output.

5.3.2 Results evaluation

This sections defines the metrics for the evaluation of the NeNTEA performance.

Let $Y'(NN_i)$ denote a vector of predicted output values by NN_i for a given input vector. During testing the $Y'(NN_i)$ value can be equal to the target value $Y(NN_i)$ or not. For the detection problem, the number of input vectors from the instance file $\mathcal{I}$ where $Y'(NN_i) = Y(NN_i)$ is denoted as $C(\mathcal{I})$.

Definition 5.3.1. The *metric* R_D for measuring the quality of the detection of untrusted nodes is defined as the ratio of the number $C(\mathcal{I})$ of inputs that were correctly predicted to the total number $m(\mathcal{I})$ of data instances in the instance file and is measured in %:

$$R_D = \frac{C(\mathcal{I})}{m(\mathcal{I})} * 100\% \tag{5.3}$$

Definition 5.3.2. Let us define *difference* $J(X(NN_i))$ of the predicted output value from the target output value for a particular input vector $X(NN_i)$ as:

$$J(X(NN_i)) = |Y(NN_i) - Y'(NN_i)| \tag{5.4}$$

The sum of all this differences for all data instances in the instance file of size $m(\mathcal{I})$ is used to measure the NN_i performance:

$$Sum(J_i) = \sum_{k=1}^{m} J(X(NN_i))[k] \tag{5.5}$$

Definition 5.3.3. The *metric* R_E for measuring the quality of the estimation of a node trust value should represent how the obtained results differ from the correct values. It is defined as:

$$R_E = (1 - \frac{Sum(J_i)}{m(\mathcal{I})}) * 100\% \tag{5.6}$$

The values of R_D or R_E are calculated for every instance file.

We have performed 8 experiments on Datasets 1, 2, and 3. In order to see how successful a given experiment was, we have calculated two general values:

- overall results,
- detailed results.

For the *overall results* we take the values R_D or R_E of all instance files from one topology and calculate their average, median, minimum, and maximum. Then these descriptive statistics are averaged by 100 topologies to minimize the effect of the particular topology (see Tables 5.7 and 5.9).

The *detailed results* were calculated to understand how the ability of a NN to detect an untrusted node or to estimate trust of a given intermediate node depends on the structural properties of a given WANET. We have grouped all the problem instances with the same number of source nodes using an investigated node as intermediate. We have defined this number as γ_i (Section 5.2.3). All topologies have 20 nodes and therefore every intermediate node can be used by $\gamma_i \in \{1, .., 19\}$ source nodes (cases of $\gamma_i = 1$ and $\gamma_i = 2$ are ignored) - see Tables 5.8 and 5.10. The value of γ_i is equal to the number of inputs of the corresponding NN_i modeled for those problem instances. We calculate the median, quartiles Q_1 and Q_3, minimum, and maximum of the quality metrics R_D or R_E and draw box-and-whisker plots (Figures 5.11 and 5.12).

5.3.3 Experiments with NN Architecture

First experiments was performed to adjust the following NN parameters:

- the type of NN;
- momentum and learning rate;
- the number of layers;
- the number of learning epochs;
- the number of neurons in the hidden layer.

These parameters are independent and together they create a lot of possible combinations. It would be very hard to investigate all of them. In the following paragraphs the reasons for choosing particular values are given.

We are bound to operate with available implementations of NN, namely OpenNN and FANN. The later was used for the implementation of NNs in our research. As for topology, both libraries are implementations of multilayer feed-forward NNs. In search of suitable parameters for solution, several models of a neuron were tested: perceptron with the sigmoid activation function and RBF neuron with Gaussian activation function. It turned out that on constructed data perceptrons learn faster and have better performance.

The FaNN library uses Rprop backpropagation algorithm as the default learning method. Rprop, short for resilient backpropagation, is a learning heuristic for supervised learning in feedforward artificial NNs. Rprop is one of the fastest weight update mechanisms [33].

Parameters of this learning method such as momentum and learning rate were left with default values, as they showed good performance. The increase factor η^+ is empirically set to 1.2 and the decrease factor η^- to 0.5, see Section 2.4.3 [33].

One hidden layer is sufficient for the large majority of problems [50]. We have performed several experiments that showed that one hidden layer is sufficient for our research.

Several experiments were conducted to figure out the best combination of NN parameters, such as the number of neurons in the hidden layer or the number of learning epochs. The learning process does not get the same results for the same inputs because the initial weights configuration is randomly generated. To minimize the impact of this randomness, ten detections and ten estimations for every investigated parameter combination were carried out. This experiment showed that the quality of the detection is influenced more by the number of learning epochs than by the number of neurons in the hidden layer.

The number of learning epochs changed the result quality significantly between 1000 and 2000. Other changes increased the quality of results very slightly. The number of learning epochs influences not only the quality of results but also the total time of learning process. Therefore, we chose 2000 learning epochs.

We wanted to pick up parameters both for detection and estimation. For 2000 epochs 5 neurons in the hidden layer showed the best results.

5.3.4 Experiments with Detection of Untrusted Nodes

During the experiments we tried various combinations of datasets for training and datasets for testing and obtained the following results.

5.3.4.1 Experiment 1

The NNs trained on Uniform-one dataset were tested again on data from Uniform-one dataset. The overall results can be seen in Table 5.7.

Table 5.7: Experiments with detection - overall results of R_D.

Experiment	Dataset trained	Dataset tested	Average	Median	Minimum	Maximum
Experiment 1	Uniform-one	Uniform-one	98.78	98.85	96.94	99.75
Experiment 2	Uniform-one	Normal-one	97.81	97.91	94.35	99.67
Experiment 3	Normal-one	Uniform-one	98.55	98.62	96.70	99.64
Experiment 4	Normal-one	Normal-one	98.22	98.35	95.14	99.74
Experiment 5	Uniform-one	Uniform-all	96.96	96.99	93.26	99.57

Detailed results of Experiment 1 can be found in Table 5.8.

Table 5.8: Experiment 1 - detailed results of R_D.

# of NN inputs (γ_i)	Average	Median	Minimum	Maximum	# of problem instances
3	96.01	97.03	85.88	100	121
4	96.63	99.65	82.68	99.98	99
5	97.57	99.64	87.98	99.98	78
6	98.62	99.73	90.35	100	61
7	98.58	99.63	89.85	100	77
8	98.81	99.65	89.15	99.98	57
9	99.32	99.61	95.13	99.98	46
10	99.56	99.65	97.43	100	74
11	99.48	99.59	97.43	99.93	68
12	99.46	99.63	90.65	99.93	71
13	99.38	99.65	93.98	99.98	63
14	99.54	99.58	98.35	99.90	65
15	99.38	99.58	93.80	99.88	61
16	99.44	99.55	96.28	99.93	68
17	99.45	99.53	98.13	99.93	94
18	99.48	99.55	98.20	99.98	119
19	99.43	99.48	98.30	99.93	218

Figure 5.11 contains detailed results of Experiment 1 represented by box-and-whisker plot.

The quality of detection depends on γ_i. It can be seen that R_D increases with γ_i and its dispersion decreases.

5.3.4.2 Experiment 2

This experiment uses the learned NNs from Experiment 1 to classify the testing data from Normal-one dataset. This experiment was an attempt to check the ability of NN trained on the uniform data to work with different type of data, namely data generated by normal distribution. The results are in Table 5.7. The quality is slightly worse than in Experiment 1.

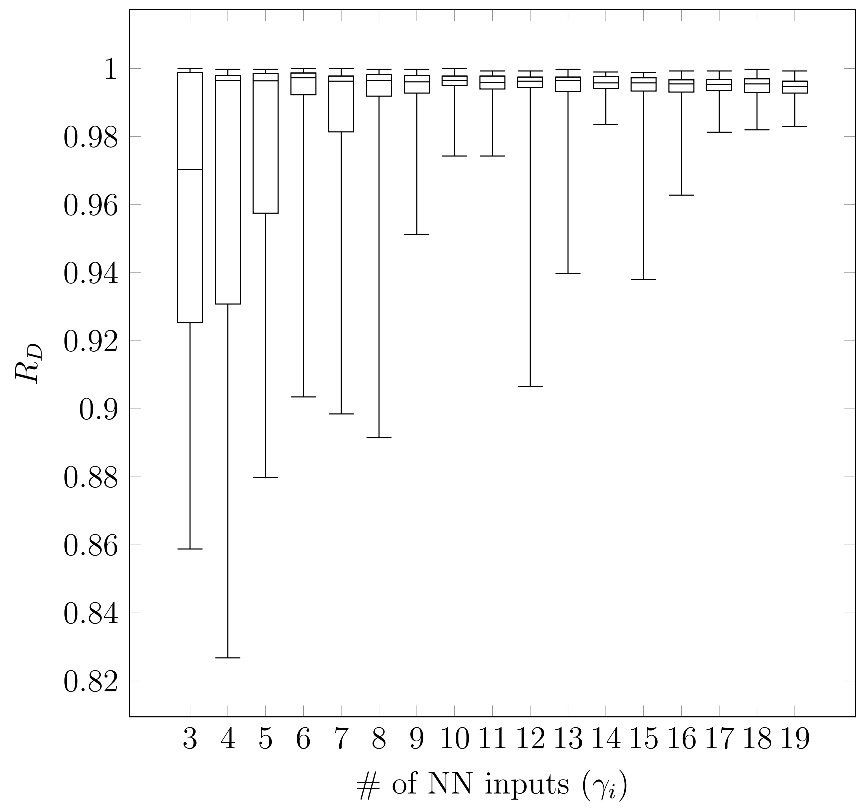


Figure 5.11: Experiment 1 detection - detailed results of R_D.

5.3.4.3 Experiment 3

The NNs trained on Normal-one dataset were tested on the data from Uniform-one dataset. It provided better quality than Experiment 2. The results can be seen in Table 5.7.

5.3.4.4 Experiment 4

This experiment tests the NNs trained on data with normal distribution on the same data. It is interesting, because it shows that the quality metric of NNs is slightly better on uniform data (Experiment 3) than on the normal data on which they were trained (Table 5.7).

5.3.4.5 Experiment 5

Since the results in the first four experiments were surprisingly good, we decided for another, harder, experiment with detection. What if we apply the NNs trained on the dataset with only one untrusted node to the networks with random number of untrusted nodes? For this experiment Uniform-all dataset was used as input for NNs trained on Uniform-one dataset. According to Table 5.7, this experiment provided slightly worse results than preceding experiments with detection, but the minimum R_D is still 93%.

5.3.4.6 Other experiments with detection

More experiments were conducted and other datasets were tried (with two untrusted nodes). However, the results were not so interesting and therefore they are not provided here, but in Appendix A.

5.3.5 Experiments with Estimation of Trust Values

For the problem of estimation no threshold value is defined.

5.3.5.1 Experiment 6

In this experiment the NNs trained on Uniform-all dataset are used. Testing was held on the same dataset. The overall results are shown in Table 5.9.

Table 5.9: Experiments with estimation - overall results of R_E.

Experiment	Dataset trained	Dataset tested	Average	Median	Minimum	Maximum
Experiment 6	Uniform-all	Uniform-all	0.9661	0.9676	0.9412	0.9776
Experiment 7	Uniform-all	Normal-one	0.9420	0.9449	0.9084	0.9581
Experiment 8	Normal-one	Uniform-all	0.9250	0.9263	0.8679	0.9773

It can be seen that the quality is over 96%, which is a very good result for the estimation of trust value of a node from the averaged values of paths' trust. If we take a look at the detailed results sorted by the number of inputs in Table 5.10 and Figure 5.12, we can notice worse quality for small γ_i. This behavior is similar to Experiment 1.

Table 5.10: Experiment 6 - detailed results.

# of NN inputs (γ_i)	Average	Median	Minimum	Maximum	# problem instances
3	92.75	96.12	81.23	98.18	121
4	93.75	97.50	80.20	98.22	99
5	94.78	97.52	82.47	98.38	78
6	96.13	97.60	85.26	98.18	61
7	96.32	97.59	85.01	98.23	77
8	96.47	97.60	83.34	98.32	57
9	97.26	97.65	89.34	98.30	46
10	97.52	97.64	93.23	98.13	74
11	97.69	97.71	95.05	98.23	68
12	97.50	97.67	85.02	98.12	71
13	97.46	97.70	89.53	98.28	63
14	97.70	97.71	96.33	98.23	65
15	97.31	97.64	88.32	98.18	61
16	97.66	97.71	93.36	98.22	68
17	97.64	97.67	95.35	98.20	94
18	97.70	97.70	97.10	98.32	119
19	97.57	97.72	64.67	98.28	218

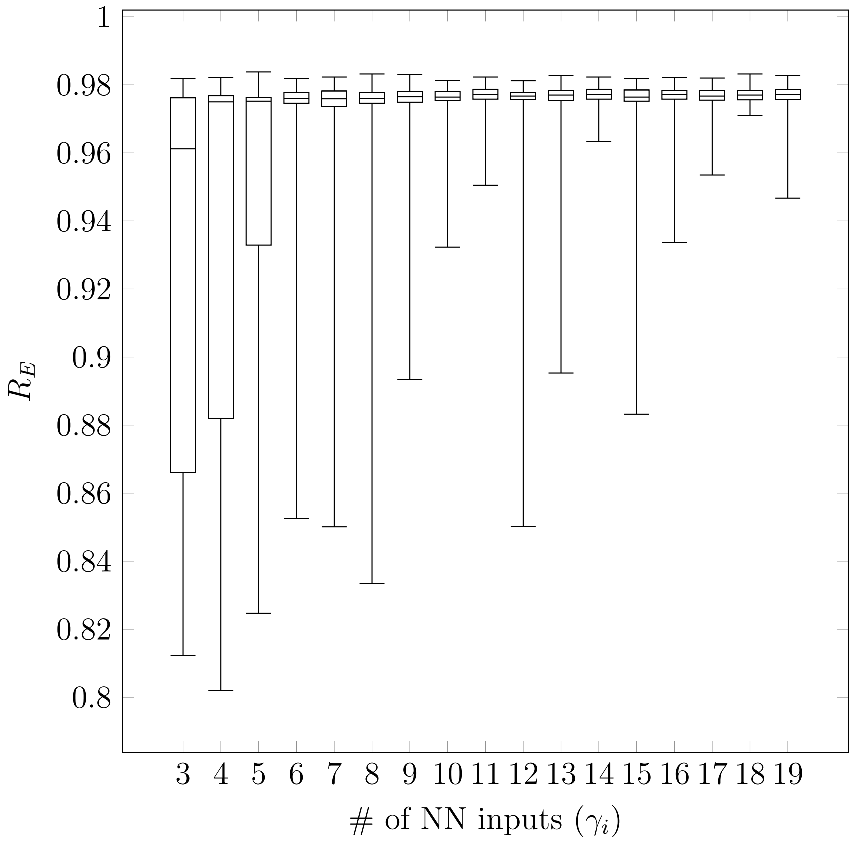


Figure 5.12: Experiment 6 estimation - detailed results of R_E.

5.3.5.2 Experiment 7

The previous experiment achieved good results on the ideal data. In this experiment the same NNs were tested on Normal-one dataset. This experiment should simulate the quality of NNs under different conditions. As it was expected, the quality is worse than in Experiment 6. However, according to Table 5.9, it is still over 94%.

5.3.5.3 Experiment 8

NN is trained on the data from Normal-one dataset, which represent WANETs with at most one untrusted node and normal distribution of PDR values. This NNs are tested on Uniform-all dataset, where all nodes in a WANET can be untrusted. Although the task looks difficult, NNs surprisingly coped well with it (see Table 5.9).

5.4 Assessment of the NeNTEA

5.4.1 Analyzing the Influence of Threshold Value τ and # of NN Inputs γ_i on R_D

According to our experiments shown in Figure 5.13, NNs can better detect an untrusted node when the threshold value is set close to 0.6. R_D is decreasing while going with τ towards the values of either 0 or 1.

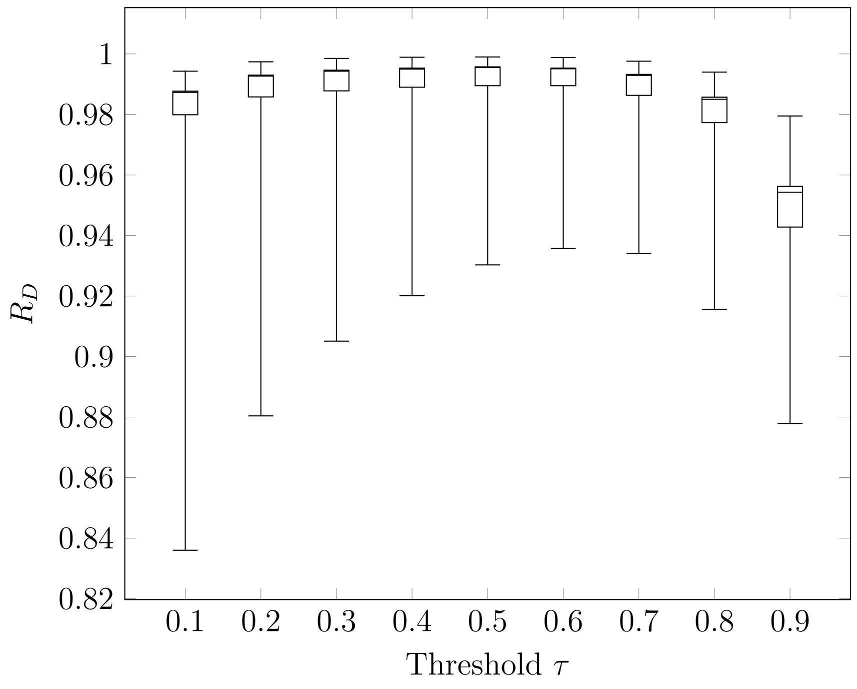


Figure 5.13: Dependency of results R_D on the threshold.

Analysis of the influence of a particular WANET's topological properties is done by grouping all problems by the number of NN inputs γ_i (see Figure 5.14, horizontal axis). It could be seen that the more inputs a NN has, the better its performance is.

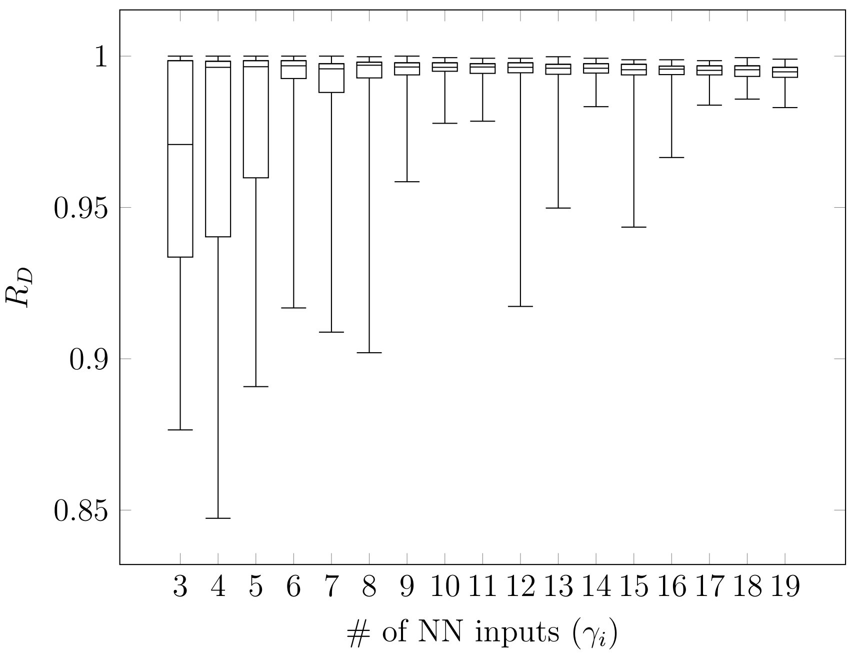


Figure 5.14: $\tau = 0.6$ - R_D dependency on γ_i.

5.4.2 Spoofed Input Vectors

The method's performance was previously evaluated with the assumption that all information provided by nodes about paths and their PDRs is correct. It was done for the proof-of-concept to determine if the method works. The next step is to examine how robust the method is to the provided incorrect information, thus NNs ability to generalize. The motivation for these experiments is the fact that nodes can provide intentionally or unintentionally incorrect data.

NNs are trained on correct data and then tested on some spoofed input, meaning some values of path PDRs are changed.

First, we describe different scenarios for the calculations of changes in input vectors. A random path (first and second scenarios) or node, which is a source of one NN input, (third and fourth scenarios) is selected for each instance to be a source of spoofed input:

1. ***ScenarioA:*** PDR of one randomly chosen path is inverted according to Equation 5.7.
2. ***ScenarioB:*** PDR of one randomly chosen path is changed according to Equations in (5.8).
3. ***ScenarioC:*** One randomly chosen element of NN input vector is inverted according to Equation 5.9.
4. ***ScenarioD:*** One randomly chosen element of NN input vector is changed according to Equations in (5.10).

Since $0 \leq PDR(P) \leq 1$, to invert the value, the *changed* $PDR^*(P)$ is calculated as:

$$PDR^*(P) = 1 - PDR(P), \tag{5.7}$$

The inversion changes PDR value drastically. Another way that may better reflect reality is to calculate the spoofed value is with the respect to the threshold τ. Cases that should be covered are depicted in Figure 5.15.

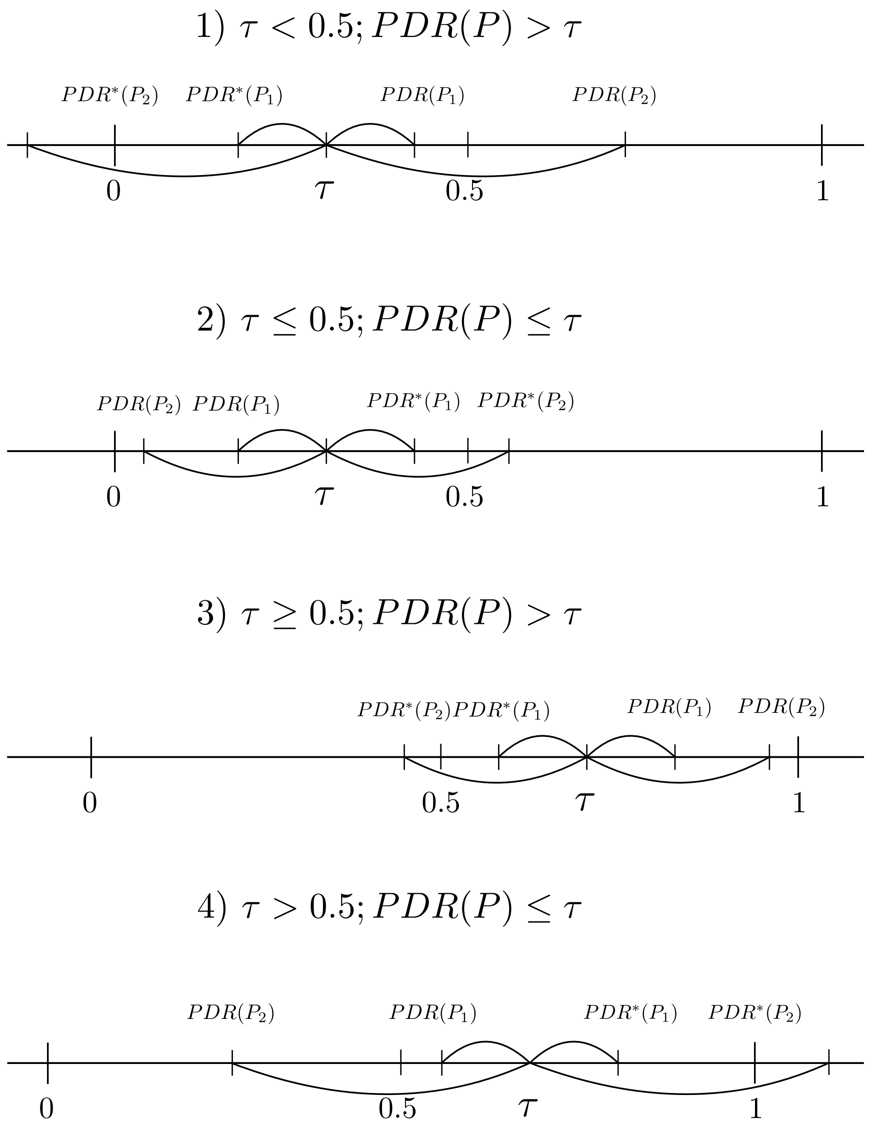


Figure 5.15: Cases of spoofing input vector.

This cases are covered by the following equations:

$$\begin{aligned}
&\tau < 0.5; PDR(P) > \tau : PDR^{*}(P) = max(0; 2\tau - PDR(P))\\
&\tau \leq 0.5; PDR(P) \leq \tau : PDR^{*}(P) = 2\tau - PDR(P)\\
&\tau \geq 0.5; PDR(P) > \tau : PDR^{*}(P) = 2\tau - PDR(P)\\
&\tau > 0.5; PDR(P) \leq \tau : PDR^{*}(P) = min(1; 2\tau - PDR(P))
\end{aligned} \tag{5.8}$$

The same calculations are applied for the element $X(NN_i)[\gamma]$ of the input vector of NN_i, see Equations 5.9 and 5.10.

$$X(NN_i)[\gamma]^{*} = 1 - X(NN_i)[\gamma], \tag{5.9}$$

$$\begin{aligned}
&\tau < 0.5; X(NN_i)[\gamma] > \tau : X(NN_i)[\gamma]^{*} = max(0; 2\tau - X(NN_i)[\gamma])\\
&\tau \leq 0.5; X(NN_i)[\gamma] \leq \tau : X(NN_i)[\gamma]^{*} = 2\tau - X(NN_i)[\gamma])\\
&\tau \geq 0.5; X(NN_i)[\gamma] > \tau : X(NN_i)[\gamma]^{*} = 2\tau - X(NN_i)[\gamma])\\
&\tau > 0.5; X(NN_i)[\gamma] \leq \tau : X(NN_i)[\gamma]^{*} = min(1; 2\tau - X(NN_i)[\gamma])
\end{aligned} \tag{5.10}$$

5.4.2.1 Comparison of Performance Metric R_D for All Scenarios of Spoofing

The experiments provide metrics R_D for the particular scenario, in the following paragraphs we compare the results of different scenarios.

As the median is not affected by outliers, we compare medians of success detection for all four scenarios. The comparison is depicted in Figure 5.16. We have the original implementation compared to our four scenarios. We can see that if the PDR value is inverted, the NN has a more significant difficulty in detecting it. Moreover, while inverting one NN input, our method has worse performance compared to just one path inversion. Nevertheless, the results are still pretty good. In the worst scenario (the third one), the method shows almost 90% detection success on average.

The comparison shows that NN performs better when spoofed trust values are calculated with respect to the threshold, according to (5.8) or (5.10). If we invert the input value, the performance of NN is worse. The explanation is simple: in case we respect the threshold, value alteration is smaller, thus NNs can generalize better. Even in the worst-case scenario, our method performance did not drop below 84%.

5.4.3 Discussion

5.4.3.1 Comparison with other methods for detection of untrusted nodes

Table 5.11 shows a comparison of our method with several other methods that deal with trust in WANETs and use similar performance metrics (detection ratio). The comparison is based, however, on data obtained with different simulators with different parameters and simulation scenarios. Unfortunately, no shared benchmarking methodology or environment exists. Implementation of methods is hard and beyond the scope of this dissertation.

Table 5.11: Comparison of different methods for detection of untrusted nodes.

Method	Description	# malicious nodes %	Detection ratio %	Reference
TDSR	Time-passed model	5	97	[72]
TSR	Fuzzy-logic rules prediction	5	98	[72]
PASID	Passive traffic monitoring	5	95	[55]
Fuzzy Trust	EigenTrust-based	10	90	[65]
Our	NN-based	5	98	-

5.4.4 Summary

First, we reviewed the assumptions for the application of NNs for estimating trust in WANETs. We stated the problem and suggested its solution using NNs. We developed a simulator of WANETs, containing a generator of datasets (see Section 5.3.1.2). These datasets were used to train and validate performance of NNs on various data. Our experiments show clearly that NNs can be effectively used for solving the problem of detection of untrusted nodes and trust estimation in WANETs.

From the experiment results it can be seen that our NN-based method provides worse detection of untrustedness for nodes with small γ_i (see Figure 5.11). Our explanation is the following: if there are only few paths passing through a node, it is harder to detect precisely that it is untrusted. NNs for nodes with $\gamma_i > 5$ show very good results. A similar behavior of NNs was observed in case of the trust estimation (see Figure 5.12).

It was also shown that NNs trained on data with normal distribution cope better with data with uniform distribution rather than with normal distribution.

Further, we analyzed the threshold value's influence on the method performance. Analysis showed that NNs perform best near $\tau = 0.6$ and slightly lose their ability to detect an untrusted node at τ values close to either 0 or 1. The analysis of the threshold value's influence was refined by grouping all problems by the number of NN inputs. This step allows confirming the assumption that the more inputs NN has, the better it performs.

The last part was dedicated to the generalization ability of the method. Having created four scenarios of spoofed input data, we have implemented them and compared the method performance. The comparison shows that NNs perform better when trust values are changed with respect to the threshold, as change in the input value in this case is smaller, thus NNs generalize better. For the worst-case scenario, our method performance did not drop below 84%.

The NeNTEA method is an input for our novel design in the next Chapter.

5.4.4.1 Benefits of the proposed approach

The application of our NN-based method to detect untrusted nodes and estimate trust values in a running WANET with a given topology is the following: we take trained NNs for the given topology, we collect path PDRs from the running WANET, use them as inputs to the NNs, and the NNs make decisions whether a given node is trusted or estimate the node trust values.

The benefit of the proposed approach is that we can simulate all possible node behaviors with the knowledge of the underlying paths and generate data for training. Therefore, the training of NN can be performed quickly without active measurement on a WANET. Comparing to other methods that need active measurements on the network to perform the learning phase [76], [39], our method does not, and that is its advantage. Passive collection of statistical data from regular communication lowers the network overhead.

The advantage of the NN-based estimation of node trusts is that some inputs can be spoofed, and the NN still predicts trust with high success [A.2].

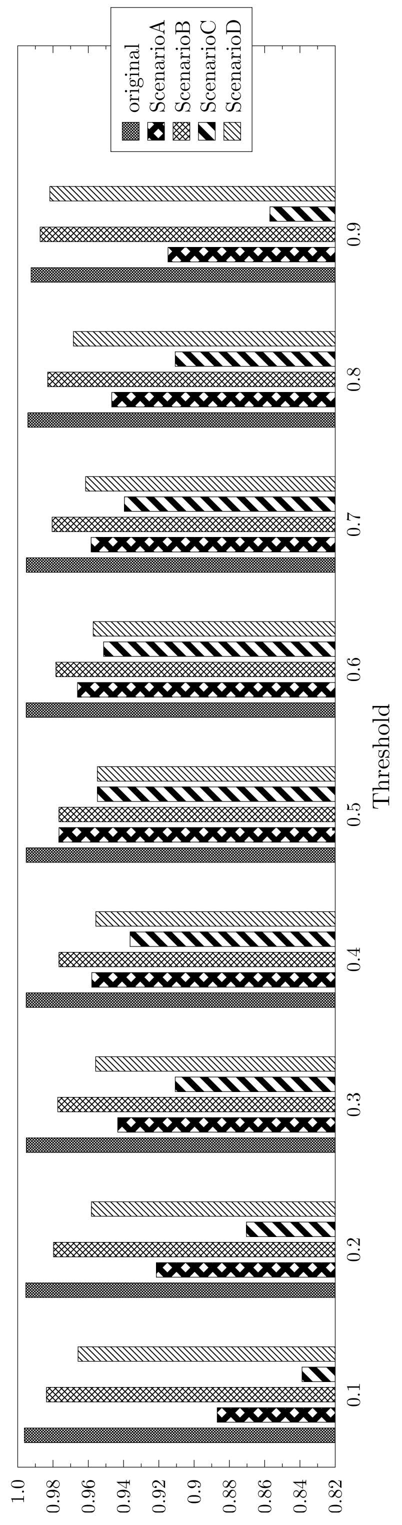


Figure 5.16: Comparison of the medians of R_D for original testing data generation and ScenarioA, ScenarioB, ScenarioC and ScenarioD.

Chapter 6

The 2nd Contribution: the TARA Method

The main idea of the approach is to enforce route discovery through the trusted nodes from outside of the routing protocol. Our trust mechanism (NeNTEA) [A.1] uses a neural network to estimate the trust of nodes. These trust estimations are then used to influence the route discovery phase of a reactive ad hoc routing protocol in such a way that more trusted routes are preferred. Untrusted nodes are penalized during the route discovery phase by delaying the RREQ messages of those nodes. Thus the node trust values influence the routing decisions indirectly to improve the performance of reactive ad hoc routing protocols. The previous approaches used trust values directly to make routing decisions.

We have built up a simulation proof-of-concept framework and conducted an extensive experimental evaluation of the proposed solution. As the ad hoc routing protocol, we chose the AODV protocol. DSR route discovery process is similar to AODV [53]. Thus, the results from the simulations with the AODV protocol can be applied to the DSR protocol, too.

Also, we perform a study of the ways the proposed solution influences the metrics of the routing protocol. Another problem is to find the best combination of parameters and to create recommendations on how to tune the method settings according to the particular use cases.

To the best of our knowledge, all previous methods of enhancing ad hoc routing with trust used the trust values directly in routing decisions. This is different from our approach in which route discovery messages through the untrusted nodes are penalized by delays, but the routing algorithm itself remains unchanged. This is one of the main contributions of this research.

6.1 Design of the TARA method

6.1.1 The Main Idea

The main idea is that the RREQ message delivery is delayed according to the trust of the previous hop node. The goal of the trust-aware route discovery is to prefer routes consisting of more trusted nodes. We will call this method TARA (Trust-Aware Reactive Ad hoc routing).

In the rest of the work, we will use the AODV protocol as the reactive ad hoc routing protocol. However, the proposed adaptivity of route discovery on trust values of nodes is independent of routing protocol implementation. It should work with any reactive ad hoc routing protocol that accepts the first obtained route.

To make the TARA method independent of the particular ad hoc routing protocol, the delaying of RREQ messages needs to be implemented as an interlayer of the ISO-OSI model. Such interlayer needs to be placed between the link and the network layer, and its job is to filter the RREQ messages coming from the link to the network layer. For each RREQ message, this interlayer decides how long the message will be delayed before sending it to the network layer. The delay value is derived from the estimated *PDR* value of the neighbor node that sent or resent the RREQ message. In this work, we consider three functions to calculate these delays, see Section 6.1.5.

6.1.2 Trust Distribution

In order to successfully deploy the suggested solution, each node needs to know the estimated *PDR*s of its neighbors. Our use case and communication model implies that the sink node always communicates with all other nodes. Thus, it can collect all necessary statistical data, estimate the trust of the nodes using NeNTEA, and distribute trust estimations to all nodes in the network. We assume that the topology of the network does not change or changes slowly in time, and each route found by the AODV protocol is used periodically or very often, therefore it does not expire.

In this work, we do not consider scenarios in which routes are rediscovered and the sink node recomputes the impacted trust values as a reaction to network changes. We will simply assume that all nodes receive the trust estimation, and we will focus on how the TARA method performs compared to the standard AODV protocol.

6.1.3 Definition of the Time Unit

Our simulation of the TARA method does not model real devices. Instead, it reproduces the results of their behavior. Thus, the simulation does not include the precise timing of the communication in a network. We decided to establish a reference operation that takes for simplicity constant time. This operation is the retransmission of the message to the next hop under ideal conditions. In reality the retransmission is not an atomic operation and it is influenced by many factors, such as node performance or the amount of data traffic going through that node. We expect that all nodes are identical and the data traffic is not too dense, therefore we could neglect these factors. We define *Time Unit (TU)* to represent the retransmission time. All metrics related to time use TU as a unit.

6.1.4 Performance Metrics

To better understand and evaluate the TARA method performance, we specify the performance metrics. We define four metrics as follows.

6.1.4.1 Network-wide Average PDR

In order to investigate the overall losses caused by the low PDR of some nodes, we define the *average PDR* metric (PDR_{avg}), which is the mean of all path PDRs in the network:

$$PDR_{avg} = \frac{\sum_{i=1}^{p} PDR(P_i)}{p}, \tag{6.1}$$

where P_i is the i-th path and $p = |\mathcal{P}|$ is the number of paths in the network. For our use case of sensors sending data to one sink, there are $p = n - 1$, where n is the number of nodes in the WANET.

6.1.4.2 Network-wide Average Path Length

When we incorporate trust into the routing protocol, it does not find the shortest paths anymore. It may pick more trusted but longer alternative routes. The metric called *average path length* (PL_{avg}) helps us understand how the average path length was influenced by incorporating trust into the route discovery mechanism. PL_{avg} is defined as:

$$PL_{avg} = \frac{\sum_{i=1}^{p} PL(P_i)}{p}, \tag{6.2}$$

where p is the number of paths P_i in the WANET.

6.1.4.3 Average and Maximum Delivery Delay of RREQ Messages

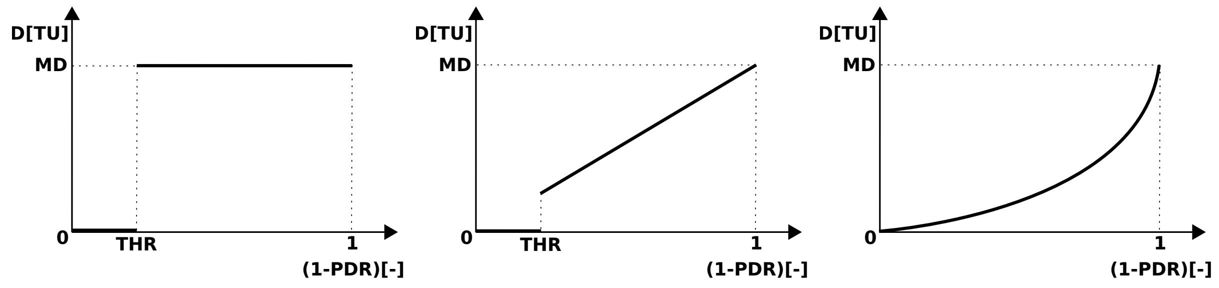
Since the TARA method delays the RREQ messages to improve the PDR of the paths in the network, it, on the other hand, prolongs the time of the new route discovery. We call it *route discovery delay (RDD)*. So metrics PDR_{avg} and RDD_{avg} are counteractive: improvement of one can deteriorate the other. In order to investigate these dependencies, we define two metrics:

- average route discovery delay (RDD_{avg}),
- maximum route discovery delay (RDD_{max}).

Both metrics are measured in *TU*s. The reason why we picked two metrics of *RDD* instead of one is that for some topologies the maximum reaches really impractical values, thus some routes cannot be found. This could be a problem for network applications when the routes to all nodes need to be found. For these purposes, it is more relevant to consider RDD_{max}. On the other hand, some network applications do not require to communicate with all nodes (for example, it is expected that some sensors are lost or no longer operational), and then the RDD_{avg} is a more useful metric.

6.1.5 Delay Functions

We consider three functions for delaying RREQ messages depending on a node *PDR*. Each delay function input consists of the estimated *PDR* and some function parameters and the output is the calculated *delay D* of the RREQ message in *TU*. Each function is described in detail in the following sections. The graphical representation of all functions is in Figure 6.1.



(a) Constant Delay function (b) Linear Delay function (c) Exponential Delay function

Figure 6.1: Delay functions.

6.1.5.1 Constant Delay Function

The most straightforward function that we experimented with is the Constant Delay function. The idea is to set the delay value to the maximal delay (*MD*) every time the estimated *PDR* of the neighbor node drops below the THR. The purpose of the THR value is not to penalize the nodes with relatively high *PDR*s. The value of *D* is calculated as a function of *THR*, *MD*, and estimated *PDR* (Algorithm 6.1).

We expect this function to have the worst results among all metrics. However, it helps us see whether delaying the RREQ messages can be successful at all. We also expect that higher *MD* values increase the resulting PDR_{avg} and both *RDD* metrics.

6.1.5.2 Linear Delay Function

The previous function does not take into account the value of the estimated *PDR*. The THR value helps solve this insufficiency only partially. The Linear Delay function delays

Algorithm 6.1 Constant Delay function specification

```
Input: $THR \in [0,1]$; $MD > 1$; $PDR \in [0,1]$
Output: $D \geq 0$
  if $PDR < THR$ then
    $D \leftarrow MD$
  else
    $D \leftarrow 0$
  end if
  return $D$
```

RREQ messages linearly according to the *PDR* of the node that sent or resent the RREQ message up to the *MD* value. And we use the threshold, too. The value of D is calculated as a function of *THR*, *MD*, and estimated *PDR* (Algorithm 6.2).

Algorithm 6.2 Linear Delay function specification

```
Input: $THR \in [0,1]$; $MD > 1$; $PDR \in [0,1]$
Output: $D \geq 0$
  if $PDR < THR$ then
    $D \leftarrow MD * (1 - PDR)$
  else
    $D \leftarrow 0$
  end if
  return $D$
```

In this case, we expect that the *THR* value does not have a positive impact on the results. We also expect that higher *MD* values increase PDR_{avg} and both *RDD* metrics.

6.1.5.3 Exponential Delay Function

The last suggested function delays RREQ messages more for the nodes with small estimated *PDR* and less for nodes with higher estimated *PDR* in comparison with the Linear Delay function. The differences are clear from Figure 6.1. The shape of the curve is determined by the following function depending on three parameters - *BASE*, *MD*, and *PDR* (Algorithm 6.3).

Algorithm 6.3 Exponential Delay function specification

```
Input: $BASE \geq 2$; $MD > 1$; $PDR \in [0,1]$
Output: $D \geq 0$
  $D \leftarrow MD * (BASE^{1-PDR} - 1)/(BASE - 1)$
  return $D$
```

As we can see from the function specification, the *BASE* value influences the bend of the curve, and the *MD* value scales the curve in the vertical direction. We expect that this function could have better results than the Linear Delay function because it penalizes more the nodes with worse *PDR*. We also expect that the *RDD* metrics values could be smaller in comparison with the Linear Delay function for the similar PDR_{avg} values.

6.1.6 Assumptions

In this part, we present implementation and results of the TARA, method for the incorporation of node trusts into reactive ad hoc routing protocols. The TARA method does not require changes to the routing protocol itself. Instead, it influences the routing choice from outside by delaying the route request messages of untrusted nodes. The method was simulated on the use case of IoT/sensor nodes sending data to a sink node. The results of experiments showed that the method improves the packet delivery ratio in the network by about 70%. We report results of the extensive amount of experiments providing an understanding of criteria for achieving an optimal trade-off between packet delivery ratio and route discovery delay requirements.

However, for the NN-based trust estimation method to perform reasonably, network topology should change slowly. Since the method proposed in this work is designed to run in symbiosis with the NN-based trust estimation method, we assume that the positions of all network nodes are fixed. Considering the communication model, the environmental conditions may change (e.g., some obstacles appear), so even with the fixed topology, *PDR*s of nodes and thus their trusts may change in time.

6.1.7 Simulator

The simulator is written in Python and has a modular structure. Each logical component of the simulated problem was implemented as a separate module. Therefore, functionality can be easily changed by replacing the module. The basic structure is shown in Figure 6.2. The logical functioning and behavior of individual modules are described in the following subsections.

6.1.7.1 Layout Module: Modeling the Network

A WANET is modeled as a graph with set of vertices $\mathcal{N} = \{N_1, N_2, ..., N_n\}$ that represent network nodes and set of edges $\mathcal{L} = \{L_1, L_2, ..., L_e\}$. Every edge represents a bidirectional link between exactly two nodes. The set of nodes and links together create a WANET topology.

The parameters for generating the set of network nodes are the 2D area size, the default value is 100x100, and the number of nodes, the default value is 100. We use the uniformly random distribution to generate the positions of nodes in the selected 2D area.

6.1.7.2 Topology Module: Generating Topologies

One of the main property of each topology is its *density* that is related to the number of links between nodes. Density highly influences the number of alternative paths between

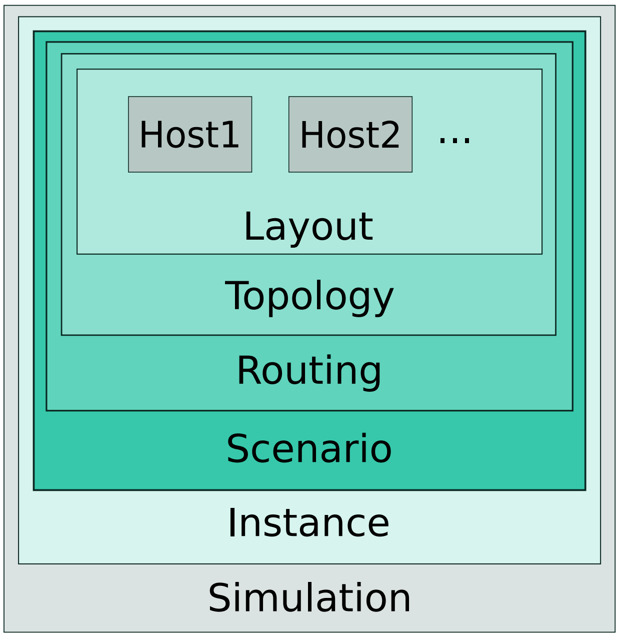


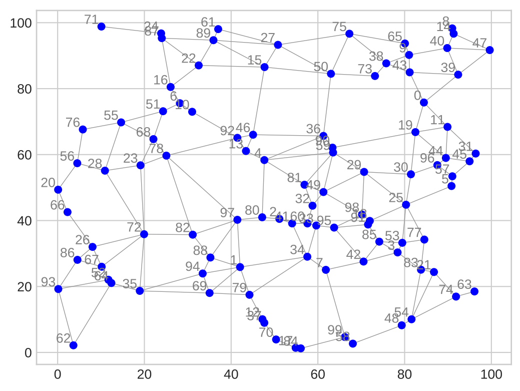
Figure 6.2: Module architecture of the simulator.

any two nodes in the network. Together with the total number of nodes, density is the most important parameter to observe when observing the behavior of routing mechanisms. To describe the density, we borrowed the definition from the graph theory. We use the *average vertex degree (AVD)*, which tells us how many neighbors each node has on average.

When generating a topology for a given 2D random distribution of node positions, we want to create a connected topology, and at the same time, we want to achieve a specific density. We decided to use the topology control algorithm called Lune β-skeleton [10]. The performance of this algorithm is influenced by parameter β that produces the topologies with various densities. A smaller β value creates a higher density network while a greater value produces a lower density network. An example of the difference between topologies created with various β parameters is in Figure 6.3. $\beta = 1$ produces a topology with more links and higher AVD compared to $\beta = 2$.

The advantage of this method is that for $\beta \leq 2$, it is ensured that the resulting topology is always connected if it was connected before the application of the algorithm. In our case, we started with the complete graph topology and then reduced the number of links with the help of this algorithm.

The resulting AVD of each topology depends on the β parameter and on the initial positions of all nodes. Therefore, the same β value produces topologies with slightly different AVDs. The dependency of the average vertex degree on the β value is shown in Figure 6.4.


(a) $\beta = 1$

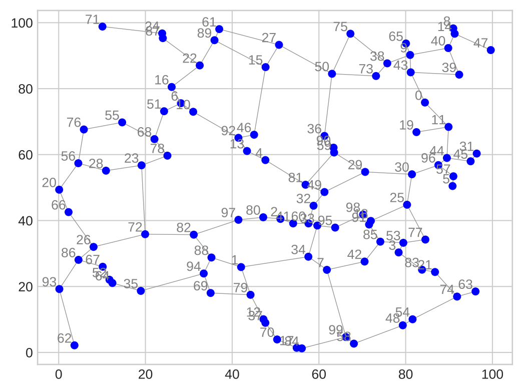

(b) $\beta = 2$

Figure 6.3: Two networks with the same distribution of nodes in the area but with different densities controlled by the β parameter.

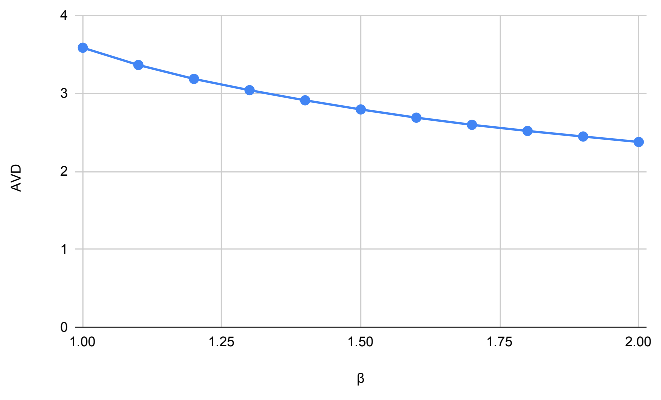


Figure 6.4: Average vertex degree as a function of the β parameter.

6.1.7.3 Routing Module: AODV Implementation

Our simulator does not implement the full specification of the AODV protocol. We implemented only the result of the AODV protocol operation. No node-to-node communication is simulated. We focused only on the way how the AODV protocol discovers the new paths and how this process is influenced by our delay mechanism.

Since the AODV route discovery process is based on flooding, it always finds the fastest discovered path. To simulate this behavior, we use Dijkstra's algorithm for searching shortest paths where the distance is the delay in TUs. If the trust penalization of a RREQ message is not involved, the delay between neighboring nodes is exactly 1 TU. Otherwise, the delay is 1 TU plus penalization D computed by the given delay function (Constant, Linear, or Exponential). If Dijkstra's algorithm returns more paths with the same delay, one of them is randomly selected. This simulates the real ad hoc routing where the fastest discovery depends (randomly) on various physical constraints (e.g., communication load).

We also implemented both AODV modes, namely when the Destination only flag is turned on (DFT) or off (DFF).

6.1.7.4 Scenario Module: Trust Distribution

This module allows to change the behavior of nodes in terms of trust.

In order to investigate the performance of the TARA method, we prepared several scenarios that differ by the number of nodes with $PDR < 1$, i.e., potentially untrusted nodes (depending on the threshold value). Nodes with $PDR < 1$ were selected randomly. The PDR values for these nodes were generated uniformly from the interval $[0, 1]$. We have limited the simulation scenarios to cases where the percentage of nodes with $PDR < 1$ are 10%, 20%, 30%, 40%, and 50%. In the following sections, we denoted these trust scenarios as SC-10, SC-20, SC-30, SC-40, and SC-50.

6.1.7.5 Instance Module

The Instance module defines the parameters of one specific simulation instance. Specific values of parameters of one simulation instance are called its *configuration*. Table 6.1 lists all parameters of the simulation instance configuration and their default values.

Table 6.1: Parameters of simulation instance.

Parameter	**Default Value**
2D area size	100x100
distribution of node positions	uniform
link type	bidirectional
number of nodes n	100
β-parameter	1.0
trust distribution scenario	SC-50
Delay function	none
Destination only flag	DFT

Default configuration simulates a general use case of reasonably large networks. Area and distribution of node positions allow to create variety of topologies in order to test fairly the TARA performance.

6.1.7.6 Simulation Module

The purpose of the Simulation module is to run simulation instances and collect, process, and save the simulation results. The unit of the simulation is 1 *experiment* which is 100 runs of one simulation instance configuration, each with different node positions and therefore with different topologies, generated by the Lune β-skeleton for the given β.

6.2 Experiments and Results

Within each experiment, we calculate PDR_{avg}, PL_{avg}, RDD_{avg} (see Section 6.1.4) as average over all generated 100 topologies of the experiment. RDD_{max} is the maximum over all the 100 topologies, representing the worst RDD. This allows to choose the best parameters of the delay function. More specifically, if some particular combination of parameters produces a reasonable PDR_{avg}, but at the same time the value of RDD_{max} is too high, it just means that the overhead of the route discovery is too high, and such combination of parameters is not acceptable. The goal of experiments is to find values of configuration parameters that maximize PDR_{avg} and minimize RDD_{avg} at the same time.

For each combination of parameters, we simulate two versions of the outputs depending on whether the Destination only flag is enabled (DFT) or not (DFF).

The *reference experiment* is an experiment with default values (see Table 6.1) of configuration. Especially, no delay function is used and, therefore, the AODV protocol is run in the standard mode, and untrusted nodes are not penalized.

The performance of each experiment will be compared relative to the performance of the reference experiment. We will enumerate the relative change for a particular metric δ_{metric} by formula (6.3).

$$\delta_{metric} = \frac{Value_{metric} - ReferenceValue_{metric}}{ReferenceValue_{metric}} * 100\% \tag{6.3}$$

$\delta_{metric} = 0\%$ means that the metric value has not changed compared to the reference value. Also, δ_{metric} can be more than 100% (see Table 6.5).

6.2.1 Dependence of the TARA Method Performance on the Delay Function

The configuration of the reference experiment is 2D area 100x100, uniform distribution of 100 nodes, bidirectional links, $\beta = 1.0$, and no delay function is applied (trust distribution scenario is irrelevant). The performance results of the reference experiment are shown in Table 6.2.

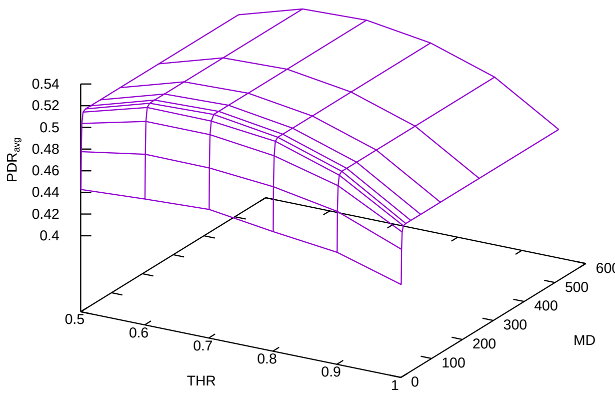
Table 6.2: Performance metrics for the reference experiment.

Flag	$\boldsymbol{PDR_{avg}}$	$\boldsymbol{PL_{avg}}$	$\boldsymbol{RDD_{avg}}$	$\boldsymbol{RDD_{max}}$
DFT	0.35	6.13	6.13	16
DFF	0.34	7.28	1.42	15

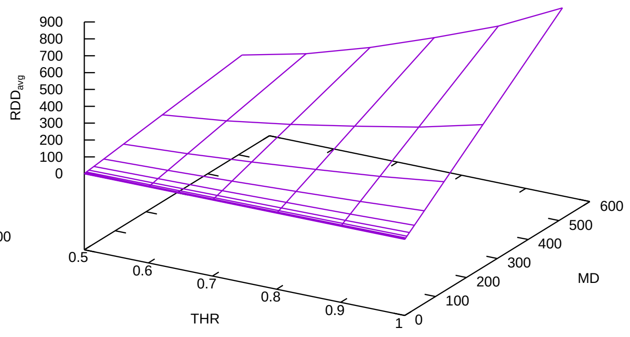
When no delay function is applied, delay at each hop is 1 TU. Thus, $RDD_{avg} = PL_{avg}$ if DFT, whereas $RDD_{avg} < PL_{avg}$ if DFF.

6.2.1.1 Constant Delay Function

The Constant Delay function serves as a proof of concept that should show us whether the TARA method has the potential to improve the performance of a WANET with untrusted nodes. For the experiments with the Constant Delay function, we chose the most pessimistic scenario SC-50. Experiments were conducted with several combinations of MD and THR parameter values for both DFT and DFF. The performance results for DFT are plotted in Figure 6.5 and for DFF in Figure 6.6.



(a) PDR_{avg}



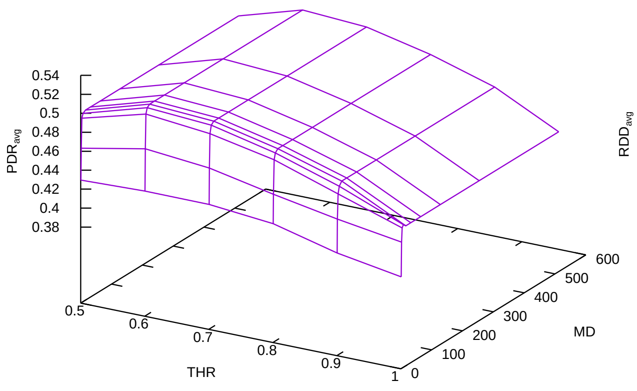
(b) RDD_{avg}

Figure 6.5: Constant Delay function + DFT.

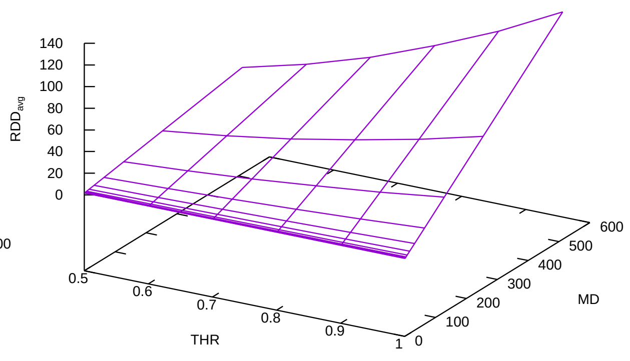
It follows from these figures that PDR_{avg} achieves maximum for THR around 0.7. The reason is that the Constant Delay function penalizes nodes with $PDR < THR$. If $THR = 1$, then nodes with high PDR values are penalized by the function, making it impossible to find routes through them. On the other hand, $THR = 0.5$ is a bad choice, as nodes with relatively low PDR values are not penalized. Thus THR value of these boundary values gives worse results.

The maximum value of PDR_{avg} and the minimum value of RDD_{avg} and combinations of delay function parameters to reach them are presented in Table 6.3.

The Constant Delay function does not reflect differences in trust values at all. Smaller values of MD are not enough for trusted path selection; on the other hand, too great



(a) PDR_{avg}



(b) RDD_{avg}

Figure 6.6: Constant Delay function + DFF.

Table 6.3: Constant Delay function parameters that provided the best values of metrics.

Flag	Perfor. metrics	Best value	δ_{metric}	Parameters		Other metric values for context			
				THR	MD	PDR_{avg}	PL_{avg}	RDD_{avg}	RDD_{max}
DFT	PDR_{avg}	0.53	51%	0.7	16	-	7.60	17.90	102
	RDD_{avg}	7.22	18%	0.6	1	0.45	6.37	-	19
DFF	PDR_{avg}	0.52	54%	0.6	16	-	8.57	2.82	63
	RDD_{avg}	1.57	11%	0.6	1	0.43	7.70	-	18

MD value does not make a significant improvement of PDR_{avg} but instead worsens route discovery delay RDD_{avg}.

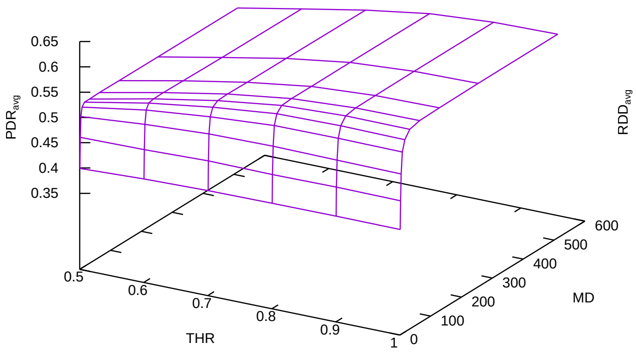
The best PDR_{avg} value is achieved for $MD = 16$. Further increasing of MD does not bring additional improvement, only worsening the RDD_{avg}. Results of the experiments with the Constant Delay function validated the proof of concept. The TARA method improved PDR_{avg} by 51% (DFT) and 54% (DFF) compared to the reference experiment.

6.2.1.2 Linear Delay Function

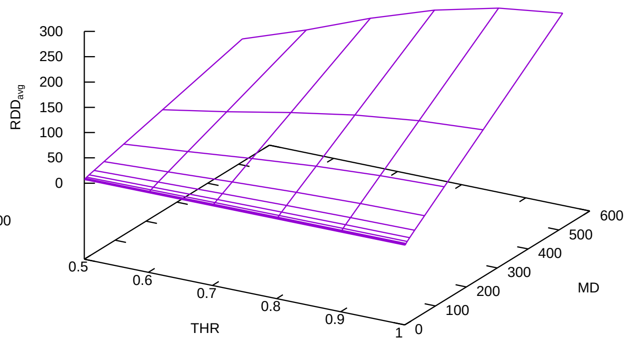
Result plots for the Linear Delay function have similar shapes compared to the Constant Delay function, compare Figures 6.7 and 6.8 with Figures 6.5 and 6.6. However, with the same configuration parameters, the TARA method with the Linear Delay function achieves significantly better PDR_{avg} and RDD_{avg}.

For the Linear Delay function, smaller MD values penalize all nodes almost the same way, and penalization became fairer with greater values.

THR value does not influence PDR_{avg} significantly. That is because the linearity of the delay function already plays the same role as the THR in Constant Delay function.

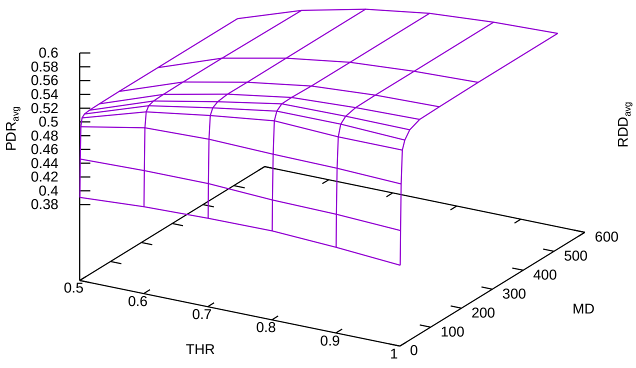


(a) PDR_{avg}

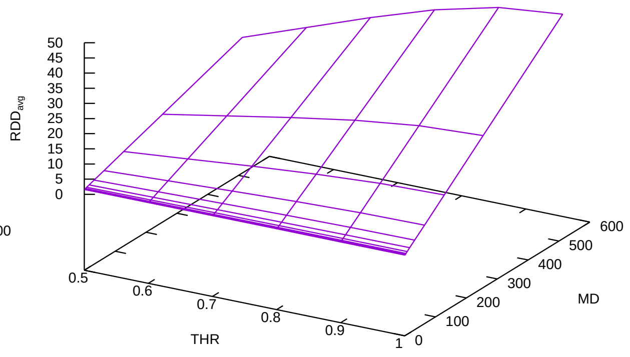


(b) RDD_{avg}

Figure 6.7: Linear Delay function + DFT.



(a) PDR_{avg}



(b) RDD_{avg}

Figure 6.8: Linear Delay function + DFF.

Further dependence found in results is that the RDD_{avg} depends on MD of the Linear Delay function. Increasing MD prolongs the path length and increases the route discovery time. RDDs for the Linear Delay function are better than for the Constant Delay one. The explanation is that by penalizing worse nodes more, the Linear Delay function generally delays RREQ messages less in comparison to the Constant Delay function.

The best metric values and combinations of the Linear Delay function parameters to reach them are presented in Table 6.4.

Table 6.5 presents the combinations of parameters for the Linear Delay function to find the best trade-off between contradictory requirements to maximize PDR_{avg} and minimize RDD_{avg}. Clearly, large MD increases significantly RDD_{avg} while PDR_{avg} improves very little. This table can be used to find the best configuration parameters once the real-world application gives us the relative importance of maximizing PDR_{avg} vs minimizing RDD_{avg}.

Table 6.4: Linear Delay function parameters that provided the best values of metrics.

Flag	Perfor. metrics	Best value	δ_{metric}	Parameters		Other metric values for context			
				THR	MD	PDR_{avg}	PL_{avg}	RDD_{avg}	RDD_{max}
DFT	PDR_{avg}	0.60	71%	1	512	-	8.64	273.75	1626
	RDD_{avg}	6.79	11%	0.5	1	0.40	6.17	-	19
DFF	PDR_{avg}	0.58	72%	1	512	-	9.56	49.05	1129
	RDD_{avg}	1.50	6%	0.5	1	0.39	7.81	-	18

Table 6.5: Linear Delay function parameter combinations for the best trade-off.

Flag	Parameters		Results			
	THR	MD	PDR_{avg}	$\delta_{PDR_{avg}}$	RDD_{avg}	$\delta_{RDD_{avg}}$
DFT	0.5	4	0.50	42%	7.97	30%
	0.6	8	0.54	52%	9.82	60%
	0.8	4	0.52	47%	9.09	48%
	0.9	8	0.56	59%	11.66	90%
	0.9	16	0.58	65%	15.92	160%
	1	32	0.59	69%	24.90	306%
DFF	0.6	4	0.51	51%	1.73	22%
	0.6	8	0.53	57%	1.95	37%
	0.7	4	0.51	51%	1.80	26%
	0.7	8	0.55	61%	2.07	46%
	0.8	8	0.56	65%	2.17	53%
	0.8	16	0.57	68%	2.82	98%
	0.9	32	0.58	70%	4.41	210%

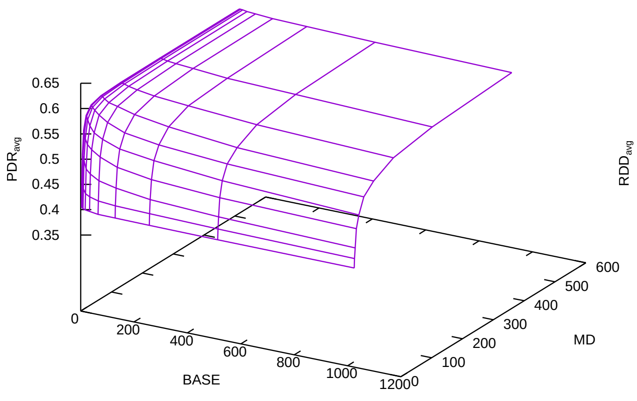
6.2.1.3 Exponential Delay Function

Results for the Exponential Delay function are shown in Figures 6.9 and 6.10.

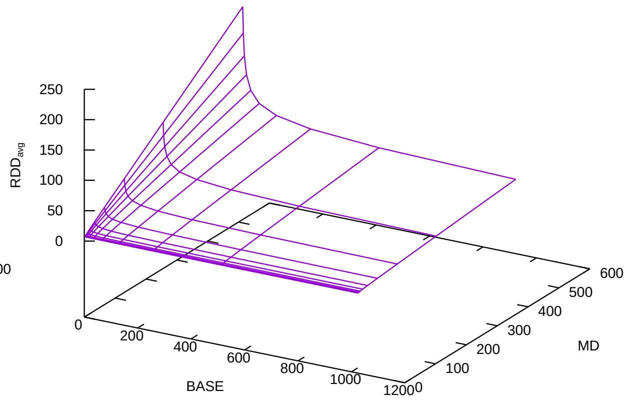
The best metric values and combinations of Exponential Delay function parameters to reach them are presented in Table 6.6.

Table 6.7 represents the combination of parameters for the Exponential Delay function for the best trade-off between maximizing PDR_{avg} and minimizing RDD_{avg} for the practical application of the TARA method.

If the Exponential Delay function is compared with the Linear Delay function, Expo-

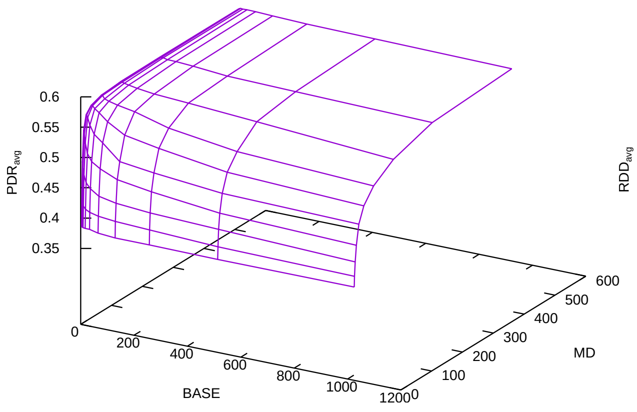


(a) PDR_{avg}

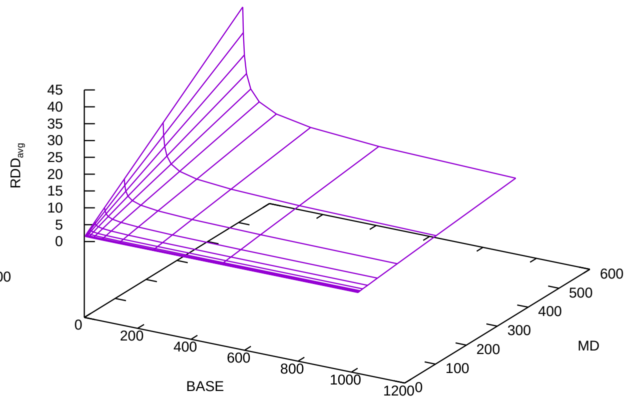


(b) RDD_{avg}

Figure 6.9: Exponential Delay function + DFT.



(a) PDR_{avg}



(b) RDD_{avg}

Figure 6.10: Exponential Delay function + DFF.

nential Delay penalizes nodes with high PDR less, depending on the $BASE$ value.

As with previous functions, greater MD increases RDD_{avg} but using a greater $BASE$ with the same MD helps reduce the delay, but at the same time, produces worse PDR_{avg}.

6.2.2 Dependence of the TARA Method Performance on the Network Density

The importance of the β parameter that determines the density of the network is shown in Figure 6.11 for the Exponential Delay function. In networks with greater β and, therefore, smaller density, fewer alternative routes can be found, thus less space is left for improving PDR_{avg}. Models DFT and DFF differ marginally.

Table 6.6: Exponential Delay function parameters that provided the best values of metrics.

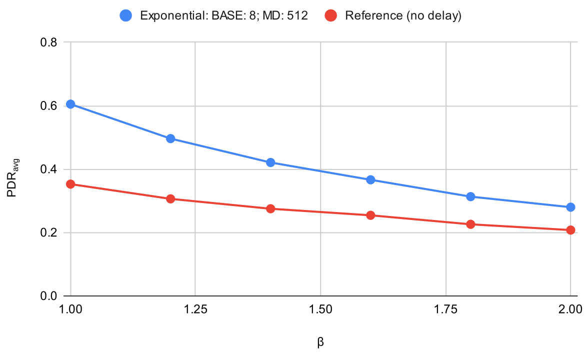
Flag	Perfor. metrics	Best value	δ_{metric}	Parameters		Other metric values for context			
				$BASE$	MD	PDR_{avg}	PL_{avg}	RDD_{avg}	RDD_{max}
DFT	PDR_{avg}	0.61	72%	8	512	-	8.82	145.93	1105
	RDD_{avg}	6.38	4%	1024	1	0.40	6.14	-	17
DFF	PDR_{avg}	0.59	74%	8	512	-	9.81	26.74	892
	RDD_{avg}	1.45	2%	1024	1	0.38	8.32	-	16

Table 6.7: Exponential Delay function parameter combinations for the best trade-off.

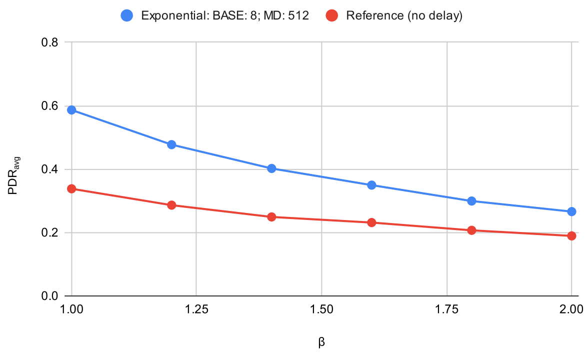
Flag	Parameters		Results			
	$BASE$	MD	PDR_{avg}	$\delta_{PDR_{avg}}$	RDD_{avg}	$\delta_{RDD_{avg}}$
DFT	4	8	0.55	55%	10.38	69%
	4	32	0.59	69%	19.26	214%
	8	16	0.57	62%	12.29	100%
	16	16	0.56	59%	11.26	84%
	16	32	0.59	66%	14.94	144%
	32	8	0.52	47%	8.71	42%
DFF	4	32	0.58	71%	3.56	151%
	8	16	0.56	66%	2.35	66%
	8	64	0.58	73%	4.73	233%
	16	16	0.55	63%	2.18	53%
	32	8	0.49	46%	1.78	25%
	1024	8	0.45	32%	1.58	11%

6.2.3 Dependence of the TARA Method Performance on the Network Size

From the experiments, it became clear that the size of the network has no effect on the performance of the TARA method. Figure 6.12 shows the results for the Linear Delay function.

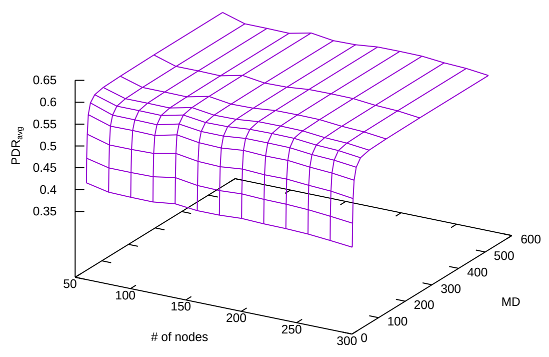


(a) DFT

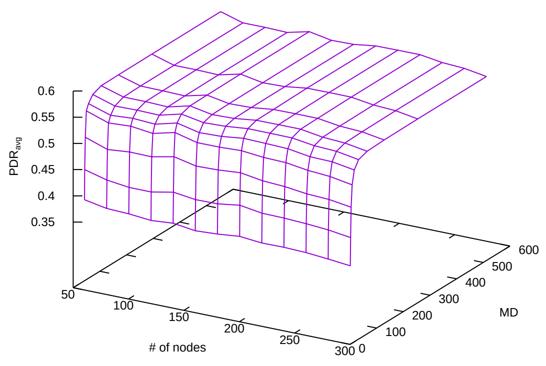


(b) DFF

Figure 6.11: Dependence of PDR_{avg} on the β parameter compared to the reference experiment (with no delay).



(a) DFT



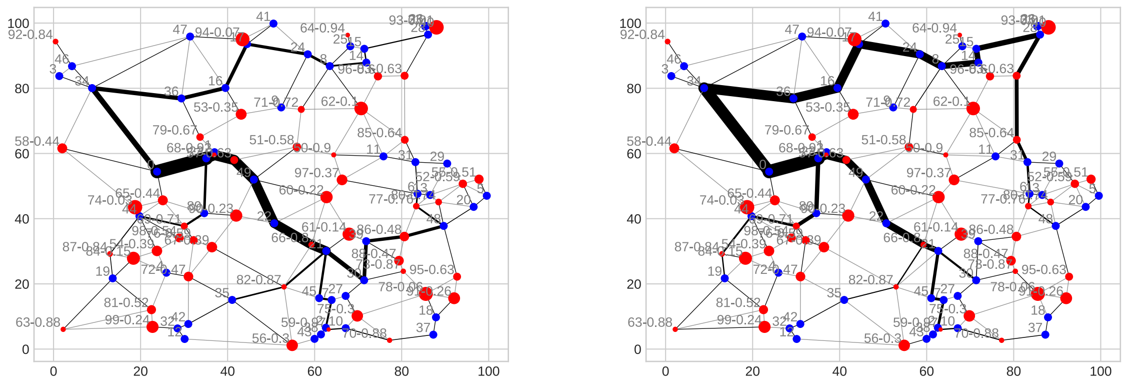
(b) DFF

Figure 6.12: Dependence of PDR_{avg} on the network size (Linear Delay: $THR = 1$).

6.2.4 Dependence of the TARA Method Performance on the Destination only Flag

Figure 6.13 shows how the setting of the Destination only flag influences construction of paths. Blue nodes have $PDR = 1$ and red nodes have $PDR < 1$. The size of the red nodes grows with decreasing of their PDR value. The value of PDR is written next to the node ID. Paths in Figure 6.13 were constructed with the default configuration and Exponential Delay function with $BASE = 8$ and $MD = 512$.

Performance metrics for Constant, Linear, and Exponential Delay functions with DFT vs. DFF are depicted in Figures 6.5 vs. 6.6, 6.7 vs. 6.8 and 6.9 vs. 6.10, respectively. DFT allows to reach greater values of PDR_{avg} for the price of longer RDD_{avg}. RDD_{avg} values of DFT are by one order greater than the corresponding results of DFF (see Table 6.8, compare DFT and DFF sections).



(a) DFF (b) DFT

Figure 6.13: Paths created with different Destination only flag setting.

6.2.5 Dependence of the TARA Method Performance on Trust Distribution Scenarios

We have conducted several experiments to analyze how scenarios of malicious behavior influence the performance of the TARA method. The percentage of potentially untrusted nodes is fixed in each scenario, ranging from 10% to 50%. Experiments with all combinations of the *BASE* and *MD* parameters were helpful, but for simplicity and clear comparison, Table 6.8 shows the results only for $BASE = 8$ and $MD = 512$.

Table 6.8: Different trust distribution scenarios.

Flag	**Scen.**	PDR_{avg}	**Ref.**PDR_{avg}	$\delta_{PDR_{avg}}$	RDD_{avg}	**Ref.**RDD_{avg}	$\delta_{RDD_{avg}}$
DFT	SC-10	0.99	0.83	19%	10.71	6.13	75%
	SC-20	0.95	0.69	39%	23.08	6.13	276%
	SC-30	0.89	0.55	63%	39.77	6.13	549%
	SC-40	0.75	0.44	71%	84.63	6.13	1281%
	SC-50	0.61	0.35	72%	145.93	6.13	2281%
DFF	SC-10	0.99	0.80	24%	2.56	1.42	80%
	SC-20	0.95	0.64	49%	4.54	1.42	220%
	SC-30	0.89	0.50	76%	8.92	1.42	527%
	SC-40	0.74	0.42	78%	16.79	1.42	1081%
	SC-50	0.59	0.34	74%	26.74	1.42	1781%

For scenarios with few untrusted nodes, the TARA method is not as efficient. Its benefit increases with the percentage of untrusted nodes.

6.3 Assessment of the TARA method

Results of experiments for all delay functions show that increasing maximum delay value *MD* improves network-wide average *PDR* significantly, but the effect weakens with greater values. Moreover, the initial increase of *MD* gives a quick boost for the *PDR* metric, but a further increase of *MD* is not effective. Improvement of *PDR* by the TARA method, compared to the reference experiment with no delay function applied, is significant for all delay functions, but Exponential Delay provided the best improvement.

THR values do not influence PDR_{avg} as significantly as the *MD* values do. For the Linear Delay function, *THR* has influence mainly on the RDD_{avg}, see Figure 6.14. The reason is that nodes with *PDR* above *THR* are not penalized, so no delay is added for them. Thus route discovery, in general, is faster.

If the Exponential Delay function is compared with the Linear Delay function, Exponential Delay penalization reflects better the differences in trust values of nodes. Depending on the bending of the exponential curve, penalization of nodes with high *PDR* starts slowly, which is sufficient, and penalization of nodes with low *PDR* value grows rapidly.

PL_{avg}, and RDD_{avg} metrics need to be minimized, contrary to the average PDR_{avg}, which needs to be maximized. Due to the delay function, improving the *PDR* means worsening *PL* and *RDD*. Greater *MD* value prolongs the path length and increases the route discovery time.

In general, applying the TARA method approach in more dense networks gives greater PDR_{avg} improvement. At the same time, the network size, under the assumption of the same density, has no influence on the performance of the TARA method.

DFT provides more trusted and shorter paths for the cost of much greater RDD_{avg}. DFF copes better with discovering paths quickly but produces longer paths and cannot reach so high PDR_{avg} as DFT.

Experiments with different scenarios showed that networks with a greater amount of untrusted nodes have a greater potential for improvement.

The aim of the experiments was to find the best combination of TARA method parameters to achieve two contradictory goals, which should also be kept in mind when applying the TARA method:

- ◦ maximize packet delivery ratio in the network PDR_{avg}
- ◦ minimize route discovery delay RDD_{avg})

This trade-off for Linear and Exponential Delay functions, both with DFT, can be studied from Figures 6.14 and 6.15. Both PDR_{avg} and RDD_{avg} for all experiments of the particular delay function with DFT are shown. The goal is to find such delay function parameters (x-axis) where the PDR_{avg} value (blue) is high, and at the same time, the RDD_{avg} value is low (red).

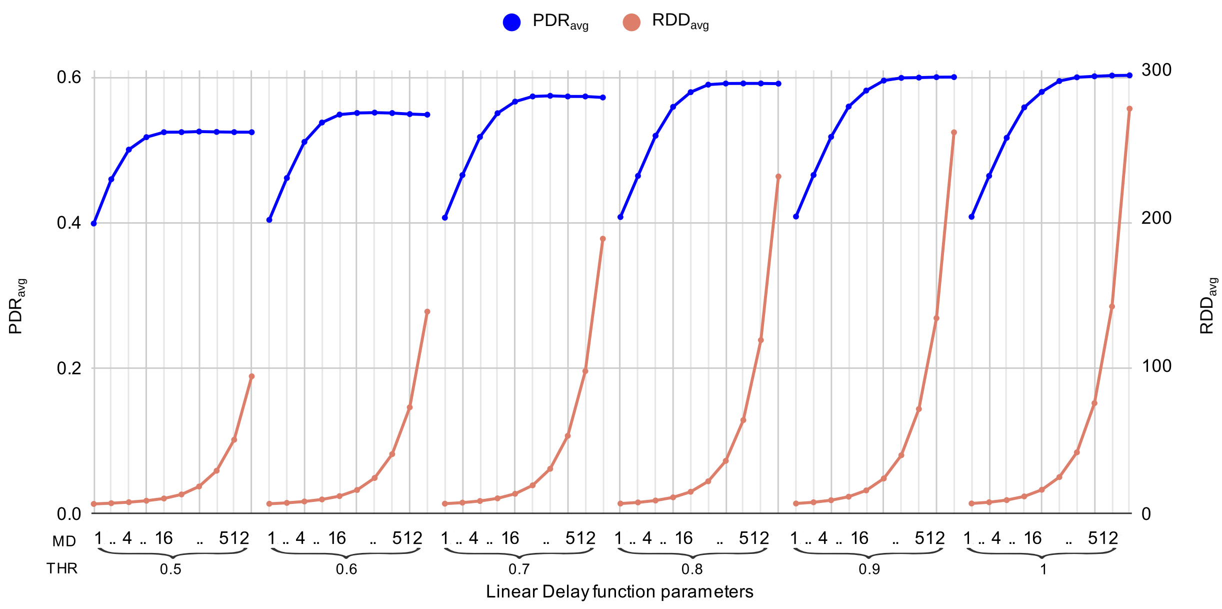


Figure 6.14: Plots from Figure 6.7 (Linear Delay function + DFT) sliced along the *THR*.

These charts visually show the dependence of performance metrics on the delay function parameters.

6.3.1 Security Analysis: Prevention Against Attacks

The TARA method can protect against blackhole and greyhole attacks. Routing loops are generally solved by default by routing protocol, in particular, AODV has mechanism to calculate TTL value, and DSR is by nature, loop free. Our protocol does not prevent an attacker from injecting data packets.

The TARA method does not attempt to provide anonymous routing, thus does not prevent from passive attacks.

6.3.2 Discussion

According to the AODV RFC [20] if a node obtains more replies for its RREQ message, it also processes them, and if they have a fresher sequence number, or the same sequence number, but the number of hops is smaller, then the routing table is updated with the new route. This is something to keep in mind when implementing the TARA method.

If there are alternative paths and the route discovery is delayed according to the trust of the nodes along those paths, the first RREP message is the fastest, meaning that path is the most trusted one. RREP for the other alternative paths RREP for the other alternative paths may take longer to arrive, implying that those paths pass through less trusted nodes.

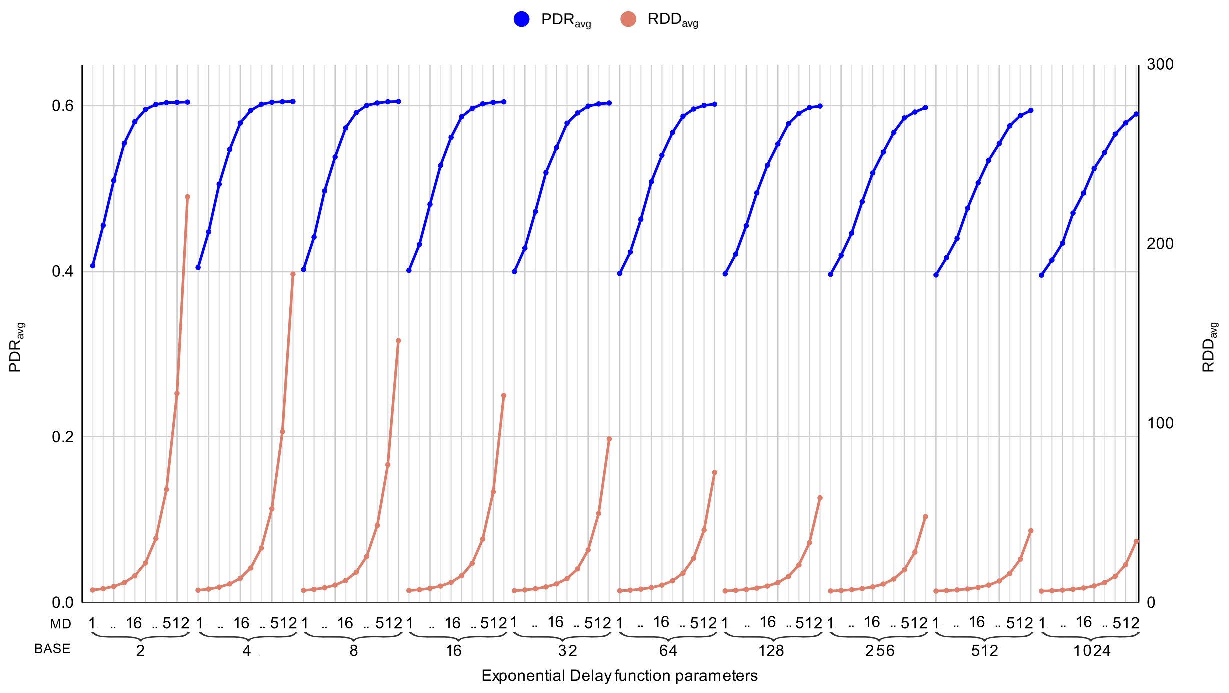


Figure 6.15: Plots from Figure 6.9 (Exponential Delay function + DFT) sliced along the *BASE*.

The less trusted path could be shorter (have fewer hops), and AODV will accept it. The goal of our method is not to change the implementation of AODV. Our interlayer that delays packets should discard the repeated RREP messages to the same RREQ message.

Another aspect to consider is the behavior of the DSR protocol, which stores several route entries for the same destination. The problem is identical to AODV, and the interlayer can address it in the same way.

Unlike AODV and DSR, the Better Approach to Mobile Ad-hoc Networking (BATMAN) routing protocol [46] is a proactive routing protocol. It was developed as an alternative to OLSR [17]. But if we analyze the BATMAN operation mode, the analogy with AODV route discovery can be seen. Each node periodically broadcast originator messages (OGMs) to its neighbors. This message is forwarded to the neighbors' neighbors, flooding the network similarly to AODV RREQ message. The fastest route is selected. Considering this properties, we assume, that the TARA method can be integrated into the BATMAN protocol.

6.3.3 Summary

The TARA method performance does not depend on the number of nodes in the WANET making it highly scalable.

Results of experiments show that different delay functions improve the average packet delivery ratio up to 78%. In essence, when using the method, parameters should be selected to trade-off between maximizing trust and minimizing route discovery delay at the same time.

The TARA method fulfilled the goal of enhancement of reactive ad hoc routing protocol with trust mechanism without the need to change the implementation of the routing itself. Various parameter combinations were tested and analyzed to provide recommendations for choosing TARA method parameters, depending on the application in a specific WANET.

The TARA method was discussed in the context of implementation in a WANET, and future work could address the concerns.

Chapter 7

Conclusions

Chapter 1 introduced the topic of the dissertation and stated goals of the research. The main motivation is that WANETs nature implies specific conditions and requirements for providing trustable and reliable communication. The main goal is to create a novel method for managing the trust in WANETs.

A theoretical background, namely definitions related to WANETs, routing protocols, concept of trust, and neural networks are provided in Chapter 2. Chapter 3 summarizes previous results and related work including methods that use neural networks and approaches of strengthening routing protocols in WANETs with trust.

Chapter 4 defines the research problem this dissertation is to solve. The ways of detecting node trust are discussed, and the problem is formulated mathematically.

Chapters 5 and 6 present the main contributions of the dissertation.

In Chapter 5 a novel TMS called NeNTEA for the estimation and prediction of node trust is presented. NeNTEA is based on neural networks. A technique for generating data for NN training is specified. Testing environment, specification of the experiments and their results interpretation is provided. At first, proof-of-concept experiments proved the validity of the NeNTEA method, which shows in average 98% accuracy of the classification and 94% of the regression problem. Further conducted experiments analyzed the method performance under malicious conditions. In the worst scenario, NeNTEA shows almost 90% detection success on average.

Chapter 6 is dedicated to the TARA method. Its main idea is to delay messages according to trust while the ad hoc routing algorithm remains unchanged. Three delay functions are defined and described. Assumptions about their performance are mentioned. The results of experiments show that different delay functions improve the average packet delivery ratio up to 78%.

7.1 Summary

The review and analysis of related work and studies demonstrated that secure and trusted communication in WANETs remains a challenging issue. Cooperation and trust establishment are significant challenges in WANET security. Moreover, vast diversity of types and application scenarios for WANETs imply additional difficulties for specifying requirements on design of secure protocols.

The research presented in this dissertation thesis addresses the problem of untrusted nodes detection in a WANET and applying that knowledge to enhance routing protocol. A method to detect untrusted nodes and to estimate the value of trust using NNs was constructed. The proposed method for evaluation of trust is based on NNs. In the second step, the research takes the method and applies it to improve the performance of reactive ad hoc routing protocols. The proposed method was simulated to be applied to the AODV routing protocol, but can be used with any reactive ad hoc routing protocol.

7.2 Contributions of the Dissertation Thesis

The contributions of this dissertation thesis are:

- a formal definition of the problem of trust management;
- development of a novel trust management method based on NN - NeNTEA;
- analysis of the NeNTEA sensitivity for the incorrect data provided;
- development of a method to apply trust on reactive ad hoc routing protocols - TARA.

An important contribution of the research is a verification of the hypothesis that NNs are applicable on the problem of trust in WANETs. The proof of concept was accomplished successfully.

It is also confirmed that synthetic generation of WANET traffic in a simulator is sufficient for the training of a NN that is then capable to accurately estimate trust in a WANET.

The work presents a formal mathematical ground of the method for trust evaluation in WANETss.

The sensitivity analysis of the method dependency on τ value was performed.

The model designed for the proof of concept has several simplifications. Research directed to the creation of more realistic model. Furthermore, by providing the solution with malicious information, we have executed the method's security analysis.

After we have proposed a method to evaluate the trust of the particular node in a WANET using NNs, the research is devoted to designing a way to integrate trust in reactive routing protocols. The challenge was to do it without changing the routing algorithm, i.e., to influence the routing decision from the outside. NeNTEA can also be used for other applications than routing, such as malicious attack detection, data aggregation, QoS.

A method to enhance trust in reactive routing protocols, called TARA, is introduced, and analysis of the implementation and settings of the TARA method are provided. The method's main idea is to delay the route discovery messages of untrusted nodes, forcing the more trusted paths to be chosen, so the goal of enhancing routing protocol without changes to the protocol itself is fulfilled. Moreover, TARA method is independent of the TMS used in the particular WANET.

7.3 Future Work

Future work will focus on creating the distributed version of the NeNTEA method, which will overcome the current approach's limitations.

Also, further research can concentrate on an attempt to investigate trust evaluation methods for other types of node maliciousness rather than packet dropping, e.g., packet misforwarding.

As for the TARA method, the author of the dissertation thesis suggests to explore the following:

- It would be interesting to add other delay functions to extend the model.
- The implementation of our methodology could be further improved by adding communication delay.
- Apply the method to other communication models.
- Consider the implementation of the method on some testbed.

Bibliography

[1] IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pages 1–4379, 2021.

[2] A. Aggarwal, S. Gandhi, N. Chaubey, and K. A. Jani. Trust Based Secure on Demand Routing Protocol (TSDRP) for MANETs. In *2014 Fourth International Conference on Advanced Computing Communication Technologies*, pages 432–438, 2014. ISBN: 978-1-4799-4910-6.

[3] A. Ahmed, K. A. Bakar, M. I. Channa, K. Haseeb, and A. W. Khan. TERP: A Trust and Energy Aware Routing Protocol for Wireless Sensor Network. *IEEE Sensors Journal*, 15(12):6962–6972, 2015.

[4] W. Alnumay, U. Ghosh, and P. Chatterjee. A Trust-Based Predictive Model for Mobile Ad Hoc Network in Internet of Things. *Sensors*, 19(6), 2019.

[5] F. Anjum and P. Mouchtaris. *Secure Routing*, chapter 4, pages 69–119. John Wiley & Sons, Ltd, 2007.

[6] R. Azmi, M. Hakimi, and Z. Bahmani. Dynamic Reputation Based Trust Management Using Neural Network Approach. In *IJCSI International Journal of Computer Science Issues (IJCSI)*, page 161, September 2011.

[7] C. A. Balanis. *Antenna Theory: Analysis and Design.* Wiley-Interscience, 2005.

[8] R. K. Bar, J. K. Mandal, and M. M. Singh. QoS of MANet Through Trust based AODV Routing Protocol by Exclusion of Black Hole Attack. *Procedia Technology*, 10:530 – 537, 2013. First International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA) 2013.

[9] A. Beheshtiasl and A. Ghaffari. Secure and Trust-Aware Routing Scheme in Wireless Sensor Networks. *Wireless Personal Communications*, 107:1799–1814, 2019.

[10] M. Bhardwaj, S. Misra, and G. Xue. Distributed Topology Control in Wireless Ad Hoc Networks using β-Skeletons. In *IEEE Workshop on High Performance Switching and Routing (HPSR)*, page 5. IEEE, 2005.

[11] S. Biswas, T. Nag, and S. Neogy. Trust based energy efficient detection and avoidance of black hole attack to ensure secure routing in MANET. In *Applications and Innovations in Mobile Computing (AIMoC), 2014*, pages 157–164, Feb 2014.

[12] S. Buchegger and J.-Y. Le Boudec. Nodes bearing grudges: towards routing security, fairness, and robustness in mobile ad hoc networks. In *Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403–410, Canary Islands, Spain, 2002. IEEE Comput. Soc.

[13] M. Bursell. *Trust in Computer Systems and the Cloud*. John Wiley & Sons, Inc., 2022.

[14] L. Buttyán and J.-P. Hubaux. *Security and Cooperation in Wireless Networks: Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing*. Cambridge University Press, 2007.

[15] N. Chaki and R. Chaki, editors. *Intrusion Detection in Wireless Ad-Hoc Networks*. CRC Press, 2014, 1st edition.

[16] J.-H. Cho, A. Swami, and I.-R. Chen. A Survey on Trust Management for Mobile Ad Hoc Networks. *Communications Surveys Tutorials, IEEE*, 13(4):562–583, Apr 2011.

[17] T. H. Clausen, C. Dearlove, P. Jacquet, and U. Herberg. The Optimized Link State Routing Protocol Version 2. RFC 7181, Apr. 2014.

[18] M. Conti and S. Giordano. Mobile ad hoc networking: milestones, challenges, and new research directions. *IEEE Communications Magazine*, 52(1):85–96, jan 2014.

[19] K. S. Cook, editor. *Trust in Society*. Russell Sage Foundation Series on Trust, 2003.

[20] S. R. Das, C. E. Perkins, and E. M. Belding-Royer. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, jul 2003.

[21] M. Dubey, P. S. Patheja, and V. Lokhande. Reputation based trust allocation and fault node identification with data recovery in MANET. In *Computer, Communication and Control (IC4), 2015 International Conference on*, pages 1–6, Sept 2015.

[22] T. Eissa, S. Abdul Razak, R. H. Khokhar, and N. Samian. Trust-based routing mechanism in MANET: Design and implementation. *Mobile Networks and Applications*, 18:666–677, 2013. ISBN: 1383-469X.

[23] E. Eziama, K. Tepe, A. Balador, K. S. Nwizege, and L. M. S. Jaimes. Malicious Node Detection in Vehicular Ad-Hoc Network Using Machine Learning and Deep Learning. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, 2018.

[24] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[25] S. Guleng, C. Wu, X. Chen, X. Wang, T. Yoshinaga, and Y. Ji. Decentralized Trust Evaluation in Vehicular Internet of Things. *IEEE Access*, 7:15980–15988, 2019.

[26] B. Hammi, S. Zeadally, H. Labiod, R. Khatoun, Y. Begriche, and L. Khoukhi. A secure multipath reactive protocol for routing in IoT and HANETs. *Ad Hoc Networks*, 103:102118, 2020.

[27] G. Hatzivasilis, I. Papaefstathiou, and C. Manifavas. SCOTRES: Secure Routing for IoT and CPS. *IEEE Internet of Things Journal*, 4(6):2129–2141, dec 2017.

[28] S. Hazra and S. Setua. Trusted AODV Protocol against Blackhole Attack in Wireless Ad-hoc Network. 2013.

[29] F. Hu, B. Chen, D. Shi, X. Zhang, and H. Z. Pan. Secure Routing Protocol in Wireless Ad Hoc Networks via Deep Learning. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2020.

[30] Y. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Wireless Networks*, 11:21–38, 2005.

[31] E. Y. Imana, F. M. Ham, W. Allen, and R. Ford. Proactive reputation-based defense for MANETs using radial basis function neural networks. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, July 2010.

[32] R. H. Jhaveri, N. M. Patel, Y. Zhong, and A. K. Sangaiah. Sensitivity Analysis of an Attack-Pattern Discovery Based Trusted Routing Scheme for Mobile Ad-Hoc Networks in Industrial IoT. *IEEE Access*, 6:20085–20103, 2018.

[33] E.-Y. Ji, Y.-J. Moon, J. Park, J.-Y. Lee, and D.-H. Lee. Comparison of neural network and support vector machine methods for Kp forecasting. *Journal of Geophysical Research: Space Physics*, 118(8):5109–5117, 2013.

[34] D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728, feb 2007.

[35] A. Jøsang and S. Pope. Semantic constraints for trust transitivity. *Proceedings of the 2nd Asia-Pacific Conference on Conceptual Modelling*, pages 59–68, 2005.

[36] A. Jøsang and S. Presti. Analysing the Relationship between Risk and Trust. In C. Jensen, S. Poslad, and T. Dimitrakos, editors, *Trust Management*, volume 2995 of *Lecture Notes in Computer Science*, pages 135–145. Springer Berlin Heidelberg, 2004.

[37] E. Karapistoli, I. Mampentzidou, and A. A. Economides. Environmental monitoring based on the wireless sensor networking technology: A survey of real-world applications. *International Journal of Agricultural and Environmental Information Systems (IJAEIS)*, 5(4):1–39, 2014.

[38] A. Karpathy. Neural Networks Part 1: Setting Up the Architecture. Notes for CS231n Convolutional Neural Networks for Visual Recognition, Stanford University, 2015.

[39] T. Kavitha, K. Geetha, and R. Muthaiah. India: Intruder Node Detection and Isolation Action in Mobile Ad Hoc Networks Using Feature Optimization and Classification Approach. *Journal of Medical Systems*, 43(6):179, May 2019.

[40] M. Kubisch, H. Karl, A. Wolisz, L. Zhong, and J. Rabaey. Distributed algorithms for transmission power control in wireless sensor networks. In *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, volume 1, pages 558–563, 2003.

[41] H. Li and M. Singhal. Trust Management in Distributed Systems. *Computer*, 40(2):45–53, Feb. 2007.

[42] X. Li, Z. Jia, P. Zhang, R. Zhang, and H. Wang. Trust-based on-demand multipath routing in mobile ad hoc networks. *IET Information Security*, 4(4):212, 2010. ISBN: 1751-8709.

[43] J. Loo, J. L. Mauri, and J. H. Ortiz, editors. *Mobile Ad Hoc Networks: Current Status and Future Trends.* CRC Press, 1st edition, 2012. https://doi.org/10.1201/b11447.

[44] N. Marchang and R. Datta. Light-weight trust-based routing protocol for mobile ad hoc networks. *IET Information Security*, 6(2):77–83, June 2012.

[45] A. Mishra. *Security and Quality of Service in Ad Hoc Wireless Networks.* Cambridge University Press, 2008.

[46] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich. Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.). Internet-Draft draft-wunderlich-openmesh-manet-routing-00, Internet Engineering Task Force, Apr. 2008. Work in Progress.

[47] S. Nissen. Neural Networks Made Simple. 2:14–19.

[48] S. Nissen. Implementation of a Fast Artificial Neural Network Library (FANN). *Department of Computer Science, University of Copenhagen (DIKU)*, 2003.

[49] G. Orr, N. Schraudolph, and F. Cummins. Momentum and Learning Rate Adaptation. Lecture Notes for CS-449: Neural Networks, Willamette University, 1999.

[50] G. Panchal, A. Ganatra, Y. Kosta, and D. Panchal. Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. *International Journal of Computer Theory and Engineering*, 3(2):332, 2011.

[51] M. S. Pathan, N. Zhu, J. He, Z. A. Zardari, M. Q. Memon, and M. I. Hussain. An Efficient Trust-Based Scheme for Secure and Quality of Service Routing in MANETs. *Future Internet*, 10(2), 2018.

[52] C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, Feb 1999.

[53] C. Perkins, E. Royer, S. Das, and M. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE Personal Communications*, 8(1):16–28, 2001. cited By 652.

[54] U. Pesovic, J. Mohorko, K. Benkivc, and Z. Cucej. Single-hop vs. Multi-hop - Energy efficiency analysis in wireless sensor networks. In *Telekomunikacioni forum TELFOR 2010, Srbija, Beograd*, pages 471–474, 2010.

[55] C. Piro, C. Shields, and B. N. Levine. Detecting the Sybil Attack in Mobile Ad hoc Networks. In *2006 Securecomm and Workshops*, pages 1–11, Aug 2006.

[56] A. A. Pirzada, A. Datta, and C. McDonald. Incorporating trust and reputation in the DSR protocol for dependable routing. *Computer Communications*, 29(15):2806–2821, sep 2006.

[57] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Neural Networks, 1993., IEEE International Conference On*, pages 586–591. IEEE, 1993.

[58] T. K. Saini and S. C. Sharma. Recent advancements, review analysis, and extensions of the AODV with the illustration of the applied concept. *Ad Hoc Networks*, 103:102148, 2020.

[59] R. F. Sari, A. Syarif, K. Ramli, and B. Budiardjo. Performance evaluation AODV routing protocol on ad hoc hybrid network testbed using PDAs. In *2005 13th IEEE International Conference on Networks Jointly held with the 2005 IEEE 7th Malaysia International Conf on Communic*, volume 1, pages 6–11, Nov 2005.

[60] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[61] S. Shanmuganathan and S. Samarasinghe. *Artificial Neural Network Modelling.* Springer Publishing Company, Incorporated, 1 edition, 2016.

[62] D. L. Silver. Time complexity of backpropagtion algorithm in neural networks.

[63] H. Simaremare, A. Abouaissa, R. F. Sari, and P. Lorenz. Security and performance enhancement of AODV routing protocol. *International Journal of Communication Systems*, 28(14):2003–2019, sep 2015.

[64] B. Solhaug, D. Elgesem, and K. Stolen. Why Trust is not Proportional to Risk. In *The Second International Conference on Availability, Reliability and Security (ARES'07)*, pages 11–18, April 2007.

[65] S. Subbaraj and P. Savarimuthu. EigenTrust-based non-cooperative game model assisting ACO look-ahead secure routing against selfishness. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):78, 2014.

[66] F. Thachil and K. Shet. A Trust Based Approach for AODV Protocol to Mitigate Black Hole Attack in MANET. In *2012 International Conference on Computing Sciences (ICCS)*, pages 281–285, Sept 2012.

[67] G. Upton and I. Cook. *A Dictionary of Statistics*. Oxford University Press, 2008.

[68] A. Varga. OMNeT++ Discrete Event Simulator. Distributed under Academic Public License.

[69] U. Venkanna, J. K. Agarwal, and R. L. Velusamy. A Cooperative Routing for MANET Based on Distributed Trust and Energy Management. *Wireless Personal Communications*, 81(3):961–979, apr 2015.

[70] J. Wang, Y. Liu, and Y. Jiao. Building a trusted route in a mobile ad hoc network considering communication reliability and path length. *Journal of Network and Computer Applications*, 34(4):1138–1149, jul 2011.

[71] Widrow, Morrow, and Gschwendtner, editors. *DARPA Neural Network Study*. AFCEA Intl, 1988.

[72] H. Xia, Z. Jia, X. Li, L. Ju, and E. H.-M. Sha. Trust prediction and trust-based source routing in mobile ad hoc networks. *Ad Hoc Networks*, 11(7):2096 – 2114, 2013.

[73] H. Xia, Z. Jia, and E.-M. Sha. Research of trust model based on fuzzy theory in mobile ad hoc networks. *Information Security, IET*, 8(2):88–103, March 2014.

[74] B. Yang, R. Yamamoto, and Y. Tanaka. Dempster-Shafer evidence theory based trust management strategy against cooperative black hole attacks and gray hole attacks in MANETs. In *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, pages 223–232, Feb 2014.

[75] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung. Blockchain-Based Decentralized Trust Management in Vehicular Networks. *IEEE Internet of Things Journal*, 6(2):1495–1505, 2019.

[76] D. Zhang, F. R. Yu, R. Yang, and H. Tang. A Deep Reinforcement Learning-Based Trust Management Scheme for Software-Defined Vehicular Networks. In *Proceedings of the 8th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, DIVANet'18, page 1–7, New York, NY, USA, 2018. Association for Computing Machinery.

[77] D.-g. Zhang, J.-x. Gao, X.-h. Liu, T. Zhang, and D.-x. Zhao. Novel approach of distributed & adaptive trust metrics for MANET. *Wireless Networks*, 25:3587–3603, 2019.

[78] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.

Reviewed Publications of the Author Relevant to the Thesis

[A.1] Trofimova, Y.; Moucha, A. and Tvrdik, P. Application of Neural Networks for Decision Making and Evaluation of Trust in Ad hoc Networks. In: *the 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, Valencia, Spain. June, 2017.

The paper has been cited in:

- Wang, Jingwen, Xuyang Jing, Zheng Yan, Yulong Fu, Witold Pedrycz, and Laurence T Yang. A Survey on Trust Evaluation Based on Machine Learning. *ACM Computing Surveys, Vol. 53, No. 5, Article 107*. September, 2020.
- Hussain, Rasheed. Trust in VANET: A Survey of Current Solutions and Future Research Opportunities. *IEEE Transactions on Intelligent Transportation Systems, Vol. 22, No. 5*. May, 2021.

[A.2] Trofimova, Y.; Fesl, J. and Moucha, A. Performance Analysis of Neural Network Approach for Evaluation of Trust in Ad hoc Networks. In: *the 11th International Conference on Advanced Computer Information Technologies (ACIT)*. IEEE, Deggendorf, Germany. September, 2021.

[A.3] Trofimova, Y. and Tvrdik, P. Enhancing Reactive Ad Hoc Routing Protocols with Trust. In: *Future Internet*, Vol. 14. MDPI, Basel, Switzerland. January, 2022.

Remaining Publications of the Author Relevant to the Thesis

[A.4] Trofimova, Y. *Security of Wireless Ad hoc Networks: Concept of Trust.* Ph.D. Minimum Thesis, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic, May, 2017.

[A.5] Trofimova, Y.; Moucha, A. and Tvrdik, P. Application of Neural Networks for Decision Making and Evaluation of Trust in Ad hoc Networks. In *the 6 th Prague Embedded Systems Workshop (PESW).* Roztoky u Prahy, Czech Republic, June, 2018.

Remaining Publications of the Author

[A.6] Fesl, J.; Konopa M.; Jelinek, J; Trofimova, Y; Janecek, J; Feslova, M; Cerny, V. and Bukovsky, I. Guarantee encrypted protocols really privacy? *the 20th European Conference on Cyber Warfare and Security*. Chester, UK. June, 2021.

Appendix A

Other experiments

A.1 Datasets

*Dataset 4, **Uniform-two**:* represents WANET with two untrusted nodes. Node PDRs are generated using uniform distribution.

*Dataset 5, **Normal-two**:* represents WANET with two untrusted nodes (similar to Dataset 4), but node PDRs are generated using normal distribution.

A.2 Results of experiments

Tables A.1 and A.2 provide overall results of the various training and testing dataset combinations. Generally they cover more possibilities, but correspond to the same conclusions made from the experiments in the Chapter 5.

Table A.1: Experiments with detection - overall results of R_D.

Experiment	Dataset trained	Dataset tested	Average	Median	Minimum	Maximum
Experiment 9	Uniform-all	Uniform-one	97.15	97.65	90.57	99.69
Experiment 10	Uniform-all	Uniform-all	97.77	97.92	93.95	99.76
Experiment 11	Uniform-all	Normal-one	96.58	96.89	90.58	99.66
Experiment 12	Normal-one	Uniform-all	97.41	97.59	93.75	99.65
Experiment 13	Uniform-two	Uniform-two	98.64	98.71	96.54	99.76
Experiment 14	Normal-two	Normal-two	98.16	98.31	94.78	99.75

Table A.2: Experiments with estimation - overall results of R_E.

Experiment	Dataset trained	Dataset tested	Average	Median	Minimum	Maximum
Experiment 15	Uniform-one	Uniform-one	97.21	97.28	96.16	97.66
Experiment 16	Uniform-one	Uniform-all	94.44	94.59	91.36	96.58
Experiment 17	Uniform-one	Normal-one	95.26	95.37	93.44	96.10
Experiment 18	Uniform-all	Uniform-one	95.81	96.00	92.78	97.14
Experiment 19	Normal-one	Uniform-one	92.51	92.51	86.44	97.20
Experiment 20	Normal-one	Normal-one	91.42	91.45	84.98	96.27
Experiment 21	Uniform-two	Uniform-two	97.14	97.24	95.91	97.66
Experiment 22	Normal-two	Normal-two	91.91	92.02	87.06	96.29