Czech Technical University in Prague

Faculty of Information Technology

Department of Computer Systems

Bachelor's thesis

Analysis of Trust Methods in Ad-hoc and Sensor Networks

Pavel Goncharov

Supervisor: Ing. Alexandru Moucha, Ph.D.

30th April 2015

Acknowledgements

This thesis would not have been possible without the continous support of Ing. Alexandru Moucha, Ph.D., to whom I am very grateful. Dr. Moucha has been the most helpful and I consider it an honor to work with him. I would also like to thank Ing. Viktor Černý for providing translation of the abstract into Czech language.

Declaration

In Prague on 30th April 2015 . . . .

Citation of this thesis

Goncharov, Pavel. *Analysis of Trust Methods in Ad-hoc and Sensor Networks.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2015.

Abstrakt

Samoorganizující ad-hoc a senzorové sítě, které postrádají logickou infrastrukturu (prvky sítě nejsou rozděleny na přístupové a ostatní), nachází uplatnění v civilních a vojenských aplikacích vyžadujícíh vysokou autonomitu (např. vzdálená, nedostupná nebo nepřátelská území). Tento typ sítí se také upřednostňuje před sítěmi s logickou infrastrukturou při nasazení do dynamického prostředí. Bez logické infrastruktury narážíme na nový typ bezpečnostních problémů, jako jsou nucená spolupráce mezi prvky sítě a detekce záškodnických prvků. Bezpečnostní mechanizmy známé z běžných sítí nelze často použít, protože síťové prvky mají velmi omezený výkon i zdroj energie. Jako velmi efektivní bezpečnostní alternativu lze použít inovativní metody založené na důvěře. Tato práce obsahuje ucelenou analýzu a klasifikaci těchto metod. Součástí práce je i tvorba nových simulačních nástrojů. Tyto nástroje poslouží jako platforma pro implementaci a vylepšování současných metod a navíc umožní jednoduchou tvorbu metod zcela nových. V budoucnu by bezpečnostní mechanismy založené na důvěře mohly být implementovány v běžných ad-hoc sítích a tím umožnit jejich masové nasazení nebo rozšíření do nových oblastí použití.

Klíčová slova Důvěra, Ad-hoc síť, Senzorová síť, Simulátor sítě

Abstract

The infrastructure-less self-organizing ad-hoc and sensor networks are suitable for autonomous operation in both civil and military areas (e.g. remote, inaccessible and hostile territory). They are preferred to traditional infrastructure-based networks for deployment in dynamic environment. The absence of the infrastructure introduces several new security issues, such as enforcement of cooperation and malicious node detection. The limited computational power and energy of nodes makes the conventional security measures not applicable to these types of networks. The innovative trust-based security approach is one of the most effective alternatives. This work provides comprehensive analysis and classification of trust methods as well as introduces the brand new simulation tools. It contributes to the networking research by providing the means to optimize and enhance the creation of the new methods as well as to improve the existing ones. In the future, efficient trust-based security mechanisms can be implemented in ad-hoc and sensor networks to enable their mass production and introduce several new areas of application.

Keywords Trust, Ad-hoc network, Sensor network, Network simulator

Contents

List of Figures

Introduction

An extensive networking research has created new opportunities for companies, governments, individuals and improved the overall productivity. However, a traditional network has several drawbacks, which prevent us from realizing its full potential. One of the problems is that it requires established infrastructure in order to provide services. The setup and maintenance of such equipment can be costly, time consuming and infeasible in far remote areas. Infrastructure-less self-configuring networks provide the means to overcome these obstacles. Ad-hoc and specialized sensor networks are the most promising among them. They have a wide range of practical applications, which include and not limited to:

- Emergency management, such as detection and monitoring of Nuclear, Chemical, Biological attacks and natural disasters.
- Vehicle traffic monitoring
- Enemy movement detection on a battlefield
- Surveillance and alarm systems

The ad-hoc and sensor networks are particularly useful for a dynamic environment, that is, when the topology frequently changes. The infrastructure-less self-configuring network still have to be secured against possible attacks, just as any other network. However, standard security measures are not applicable due to several limitations and additional requirements. Efficient trust-based methods can be used to address the specific security issues in ad-hoc and sensor networks.

Several researches have proposed different trust methods in their latest works, but its unknown whether or not they are efficient. It is still an open problem with no standardized solutions. Its vital to verify the functionality of methods in a simulated environment before real life deployment, especially when taking into account the dynamic nature of these networks. This can be achieved with the help of simulation software.

The first goal of this thesis is to research and organize the particular security issues, specific to ad-hoc and sensor networks; to provide insights into the state of the art trust-based security methods, as well as to compare and analyze them. It should allow us to classify and generalize the trust mechanisms and their main principles. The second goal is to design and implement a flexible simulation software of trust methods for ad-hoc and sensor networks, easily expandable by the new methods.

In summary, this work focuses on the generalization and classification of trust concepts for ad-hoc and sensor networks as well as development of the trust-enabled network simulator, independent of the concrete trust methods.

State of the Art

In the context of ad-hoc and sensor networks, we define node as a physical device, connected to a network and capable of communication. The node is allowed to support any additional functionality as long as those basic requirements are fulfilled. The edge or link is defined as a communication channel, connecting any two nodes.

The routing in traditional infrastructure-based network is implemented with the help of dedicated devices. In case of infrastructure-less network, the routing is achieved via message forwarding between nodes. If any of the nodes have been compromised it can create a security breach and there are no built-in countermeasures. In addition to this, ad-hoc and sensor networks are bound by very strict constraints, such as limited computational power, storage, and bandwidth, which make it impossible to use conventional security methods. There are several possible approaches to this problem, such as:

- Trust-based approach [1] [2] [3] [4] [5]
- Key management-based approach [6]
- Data mining-based approach [7]
- Game theory-based approach [8]

The security in ad-hoc and sensor networks is an open problem with no standardized solutions. The research, based on these issues, is still ongoing and at this point it is not possible to decide whether one approach is better than the other. However, the trust-based approach have received a considerable amount of attention from the scientific world and is currently one of the most promising ones. Despite that, the individual trust methods have not been analyzed, generalized and standardized yet. The researchers struggle with the creation of the new trust methods and sometimes, perhaps unaware, redefine already existing principles. This problem is escalated by the total lack of support in the simulation software. The modern network simulators have not been adjusted to work with trust yet, which makes the evaluation of new and existing methods very difficult. The researchers are forced to use the self-developed simulators for testing of their methods, which makes the collaboration and comparative analysis almost impossible. The legitimacy of the published simulation results is questionable, considering that the used simulators are usually not available for the public. It makes the recreation of the simulation environment infeasible and thus leaves the results unvalidated.

Structure of the Thesis

The first chapter provides basic concepts and definitions which can be found throughout the thesis. They include ad-hoc and sensor networks as well as trust-based security approach and related topics. The second chapter describes the current security issues in ad-hoc and sensor networks and explains why traditional security measures are not sufficient. It includes the classifications of the common security requirements, the existing attacks on the network and the vulnerabilities specific to the ad-hoc and sensor networks. The third chapter provides description, classification and analysis of the existing trust methods, as well as their generalization. The fourth chapter consists of the analysis of existing network simulators, the advantages and disadvantaged associated with them as well as common properties of network simulators. The fifth chapter is dedicated to the newly developed simulator of trust-methods for ad-hoc and sensor networks. It includes the design choices made during the development process, structure of the application and principles of work.

Chapter 1

Basic Concepts and Definitions

The basic concepts and definitions which can be found throughout the thesis are defined and introduced in this chapter. They include ad-hoc and sensor network technology as well as trust-based security approach and related topics.

The ad-hoc and sensor networks are very similar, but not the same. However, we can still assume that the trust methods presented in this thesis are applicable to both ad-hoc and sensor networks, for more details please see 1.3 on page 9.

1.1 Ad-hoc Network

In this section we provide the definition and classification of ad-hoc network. The classification is done based on the type of communication: one-hop, multi-hop and topology: flat, hierarchical.

1.1.1 Definition

Ad-hoc network can be described as self-organizing, decentralized network which does not depend on preexisting infrastructure[9].

The main research subject of this thesis is trust, which is not affected by the means of connectivity between nodes in a network. It has been assumed that all the networks presented in the thesis are wireless, based on the author's preferences.

1.1.2 Classification of Ad-hoc Networks

1.1.2.1 Types of Communication

The simplest type of ad-hoc network is a single-hop connection between two wireless devices located within their transmission ranges. The communication can be established directly between nodes, without the support of in between medium. The entire network can be relocated together as a group, as long as the individual nodes remain in the range of wireless communicators. It also implies that the network can be dynamic to a degree, but the mobility and scalability are very limited. The standards for single-hop communication include:

- IEEE 802.15.4 (Zigbee)
- IEEE 802.15.1 (Bluetooth)
- IEEE 802.11 family

The multi-hop ad-hoc network consists of multiple nodes which usually operate in 802.11 wireless ad-hoc mode. When we considered one-hop communication, all nodes were connected directly on a link layer, which is different in case of multi-hop.

In a standard infrastructure-based network the routing and switching are provided by centralized dedicated devices which cannot be used in case of ad-hoc network. To implement a routing for the multi-hop communication, each node forwards packets to its neighbors, therefore eliminating the need for infrastructure. We can assume a network consisting of only 3 nodes: A,B,C, where (A,B), (B,C) are neighbors respectively, and (A,C) are out of transmission range, therefore unable to directly communicate. In order for node A to transmit a packet to node C it will have to use node B as and intermediate.

The multi-hop communication is a required prerequisite for the trust methods to function. In the one-hop network the nodes are able to directly communicate with any reachable destination and do not require the help of others. The main challenge of multi-hop network is to choose an appropriate neighbor, therefore maximizing the likeness of successful packet delivery. It is especially true, if we take into account the dynamic nature of the network. When new node connects, the existing ones have no mechanism to determine its reliability. In 1.4 we provide more details on this problem.

1.1.2.2 Network Topology

Ad-hoc networks can also be classified according to their topology. The nodes in a network are organized into two main groups: flat and hierarchal, each having its own set of applications.

Flat ad-hoc Network is defined as a network in which all nodes have the exact same responsibility and functionality. Each and every node is able and required to participate in the network activity equally. However, the structure of individual nodes, which includes the hardware, can still differ from one node to another. All control messages are transmitted globally, throughout the whole network. This topology is very productive for dynamic environment and becomes less efficient as the total number of nodes significantly increases.

Hierarchical ad-hoc network is defined as a network consisting from nodes, which are divided into clusters according to specific hierarchy. The nodes in hierarchical topology play their own roles in a network and typically have different hardware. The literature on hierarchical ad-hoc network generally defines two types of nodes:

Master node: often refereed to as cluster head (CH), is a node whose sole function is to communicate with the rest of the clusters in a network. There exist several algorithms for the selection of CHs.

Normal node: is able to communicate with the rest of the nodes in its own cluster or external nodes via the master node. Most of the communication takes place within a cluster, where all normal nodes are able to directly communicate with each other and do not require the assistance of master node.

The hierarchal topology is secured against failure of individual nodes in a way similar to the flat topology. If any node fails, it is usually possible to find a replacement for it, e.g., select a new CH.

1.2 Sensor Network

This section provides definition of the sensor network, its limitations, structure of the sensor node as well as practical applications.

1.2.1 Definition

The sensor network is a dynamic, infrastructure-less network, which provides the ability to monitor and analyze environmental or physical conditions and react to certain events [10]. The typical network consists of a set of deployed sensor nodes, which can be grouped into clusters, the means of connectivity (wireless in most cases), data sink and a base station (if applicable). The total number of nodes in a network can exceed several thousands.

As have already been mentioned in 1.1.1, we assume that all the networks are wireless. In addition to this, it is also assumed that all the networks are homogeneous with flat topology. The homogeneous network, is a network consisting of the identical interconnected nodes. The definition of the flat network topology can be found in 1.1.2.2.

1.2.2 Limitations of Sensor Networks

The sensor network as a subtype of ad-hoc network, inherits the associated challenges and is bound by additional constraints. One of the main drawbacks of the sensor networks is limited power [11], which is usually presented in a form of a simple battery. The sensor networks are often deployed in a desolate, inaccessible or enemy controlled areas, which makes further physical maintenance impossible. The live-time of individual node is determined by the energy consumption, which makes it vital to adopt efficient management of data traffic and node computations.

Network nodes are prone to failure due to many factors, such as physical damage, system faults and malicious attacks. On average nodes are much more vulnerable in comparison with wired or traditional wireless networks. The size of a network can increase up to the point when each node would have thousands of neighbors. It essentially makes forwarding more demanding and requires optimized security methods and routing protocols. A balance between functionality of node, its physical size, power consumption and cost is not easy to find.

1.2.3 Structure of a Wireless Sensor Node

A sensor node has to be small enough to fit into a 2 x 5 x 1 cm, 1 x 1 x 1 cm or even smaller dimensions. The later is also referred to as a sensor mote and theoretically can be microscopic or nanoscopic in size, but truly nano-scaled sensor devices have not been built yet [9]. A typical sensor node consists of four basic hardware components: computational unit, communication unit, set of sensors and power supply.

The computational unit is designed for management of the sensors and implementation of networking protocols. It has to support several operation modes for efficient power management under different conditions. The switching between operation modes helps to reduce energy consumption when the required computations are not demanding. A limited storage is usually associated with the computational unit and integrated into it.

The communication unit is a device capable of radio transmission (typically short range). One of the required functions is an ability to completely turn off the unit or put it into power saving mode as the major part of total node energy is drained by it.

A single node can be connected to a group of several sensors. It is also possible to combine different types of sensors to accomplish required task. The possible sensor types include: sound, temperature, pressure, vibration, motion and others. These sensors have to be configured to operate only when required, for example to take a measurement of a physical phenomena, store or forward the data and shut down to save the energy.

The power supply can be presented in a form of embedded battery, external unit or a device capable of natural energy harvesting, such as sun light absorption. In practice, the embedded battery with limited energy is most widely used. The power plays crucial role in the sensor network as the energy level of a power unit determines the live-time of the entire node. It can be explained by the fact that once deployed, the nodes cannot be physically maintained in an efficient manner.

1.2.4 Applications of Sensor Networks

The sensor networks have a great potential and can be applied in many different areas, which include and not limited to:

Military: sensor network is an attractive technology for battlefield surveillance, intelligence gathering and weapon targeting.

Environment: deployment of sensors in forests, oceans, mountains, on volcanoes for monitoring and emergency control.

Transport: regulation of the dense city traffic, accidents detection.

Agriculture: automatic management of crops and livestock, such as condition-based watering and waste disposal.

Medical treatment: patients condition monitoring in hospitals, drug administration.

Machinery maintenance: efficient equipment diagnostics and problem reporting.

1.3 The Comparison Between Ad-hoc and Sensor Networks

This section provides brief comparative analysis of ad-hoc and sensor networks.

From the definitions of ad-hoc and sensor networks presented in sections 1.1 and 1.2 respectively, it should become clear that sensor network is a special case, or in other words, a subtype of ad-hoc network.

The basic reason behind development of the ad-hoc network is to provide the means of spontaneous communication between devices in the close vicinity. The example can be a small office with a group of 3 or more desktop computers connected via wireless units or cross-cables, but in this thesis we are only interested in wireless connectivity. The network is expected to function without any established infrastructure or manual configuration and permit communication as soon as devices detect each other. This type of network correspond to the definition of wireless multi-hop ad-hoc network presented earlier in 1.1.2.1. In case of ad-hoc network, the nodes can be built from powerful hardware (laptops, tablets, smart-phones) and have little to no restrictions on bandwidth, energy and computations.

The wireless sensor network in its core is multi-hop ad-hoc network with additional limitations, clearly outlined in 1.2.2.

In this thesis we focus on the problem of trust in a wireless ad-hoc and sensor networks. The trust methods presented throughout this work are in their majority designed for wireless sensor networks, which makes them applicable for ad-hoc networks as well. In addition to this, we also ignore the hardware details and individual network protocols, which have no effect on the trust mechanisms. For more details on trust and related topics please see 1.4.

1.4 Trust

This section briefly introduces the concept of trust and provides basic definitions. The trust in general, as well as existing trust methods and models are described, analyzed and classified in 3.

1.4.1 Definition

We define trust as a trustworthiness of node B, in the opinion of node A. The trust is not symmetric, that is, if node A trusts node B it does not imply that node B has to trust node A.

The trustworthiness is a boolean value, which indicates that node A is certain, that node B will or will not perform the requested action. The action is any operation which node B is able to execute, but in the context of this thesis we are only interested in routing. The trustworthiness is algorithmically determined with the help of a trust value. The trust value, also refereed to as a trust level, is defined as mathematical representation of the interactions between the nodes in the network. The trust is crucial for design of secure systems and is the most important concept for this thesis.

1.4.2 Trust Model

A trust model is a representation of trust relationships between entities. There are two different types of trust models: distributed and centralized.

The core part of the centralized model is a Trust Authority (TA). An example of such model can be a Certification Authority (CA) which issues digital certificates to the entities. While this model can theoretically be used for ad-hoc and sensor networks, it still suffers from scalability limitations and its central nature. The malfunction of a Trust Authority, which in this case can be a base station, will cause the failure of an entire network. The nodes in ad-hoc and sensor networks directly communicate only with their neighbors and do not need to trust the whole network.

Due to these reasons, the *distributed trust model* is more appropriate for ad-hoc and sensor networks. In the distributed model, the trust is determined locally between the individual nodes and can differ from one node to another. The trust methods described in this thesis are all based on the distributed trust model, with the additional assumptions and limitations posed on by specific methods.

Chapter 2

Security in Ad-hoc and Sensor Networks

In this chapter we describe the current security issues in wireless ad-hoc and sensor networks and explain why traditional security measures are not sufficient. The classifications of the common security requirements, the existing attacks on the network and the vulnerabilities specific to the ad-hoc and sensor networks are included as well.

2.1 Current Security Issues in Ad-hoc and Sensor Networks

There exist two main problems which prevent us from unleashing the full potential of wireless ad-hoc and sensor networks: security and privacy. A mass scale application of this technology for the benefit of industry and public society can only be achieved when these obstacles are completely removed.

These two problems are composed from several factors, which are discussed and analyzed in this section.

2.1.1 Classification of Security Requirements

The wireless ad-hoc and sensor networks have several security requirements similar to that of traditional wireless networks. However, due to the additional constraints those requirements have been further extended [9]. We classify and summarize them in the following way:

Authentication. The authentication mechanism must be able to prevent the potential attacker from falsifying the origin of the message. In other words, it has to provide the means to check that the message was really sent by the specified sender. The authentication insures that the attacker will not be able to disturb the network by sending illegitimate messages.

Authorization. The authorization specifies access rights for the nodes and protects the network resources from been compromised. In order for nodes to gain access to the network and use its resources, they have to be authorized. It also applies to the entities which intend to use network services. Access control prevents the leakage of messages though malicious nodes and can be implemented with the use of threshold cryptography. The threshold cryptography is based on splitting the data into several parts which can be retried and combined back to the original form through authentication process.

Accounting. Accounting is a process of monitoring the consumption of network resources by individual nodes. It can include the amount of transferred data and accessed services. The suspicious node activity can be an indicator of network attack.

Integrity. The messages transmitted though the network have to be protected from been altered on purpose or due to network failure. The attacker may try to insert false data, resend or destroy it. It should not be possible for malicious node to modify the contents of the message. In other words, when the destination node receives a packet, it has to be exactly as it was sent. The integrity is typically achieved with the help of cryptography, such as hash methods and digital signatures.

Confidentiality. The confidentiality protects data in a network from been disclosed to perpetrator and is usually accomplished by encrypting techniques. Only the node for which the message was originally intended should be able to understand its content. Confidentially also means that the very fact of establishing communication has to be hidden if required.

Availability. The availability implies that network and its resources has to be always accessible for the legitimate nodes. The services must be provided in a timely manner and not be affected by the ongoing network attacks, such as resource depletion.

Non-repudiation. The non-repudiation has to insure that each and every node in the network cannot pretend that it has not participated in the communication. The node which has sent a packet should not be able to deny that fact. In ad-hoc and sensor network this security requirement is crucial for identification of malicious nodes. A good example can be a sensor network which contains compromised node A and secured node B. Let assume that node A tries to send classified data out of the network or generates invalid messages to deplete available resources. In order for node B to determine that fact and identify A as a malicious node, it will have to prove that A really has sent those messages.

Freshness. The network should be able to support a mechanism which prevents the perpetrator from reusing old, obsolete messages. It should not be possible for the attacker to send false, previously generated messages in order to disrupt the network activity.

The ad-hoc and sensor networks are dynamic, which implies that legitimate nodes can freely join and leave, generating additional security issues. The two main problems that arise are forward and backward secrecy. The forward secrecy implies that a compromise of the key, which is currently in use, will not affect the future keys. On the other hand, the backward secrecy means that a compromise of a valid key wont allow the attacker to obtain the key used before. When we consider that its possible for a node to leave ad-hoc and sensor networks on a frequent basis, it becomes apparent that this node should not be able to recognize the contents of the future messages. This also means that newly connected node should not be able to compromise the messages, sent before it joined the network[9].

2.1.2 Classification of Existing Vulnerabilities

The following vulnerabilities are characteristic to ad-hoc and sensor networks:

Communication channel. Due to the nature of wireless technology, the communication channel can be compromised by eavesdropping as well as falsification of the messages.

Physical access to nodes. Ad-hoc networks, especially sensor networks, are typically deployed in remote inaccessible areas. The example can be a battlefield or hostile territory in general. It may also include the ocean floor, mountains or volcanoes. It is normally impossible to monitor the nodes themselves, protect them from physical damage or provide maintenance after they have been deployed. In a network consisting from thousands of nodes, the probability that an attacker will tamper several of them is very high. The potential attacker may attempt to replace the hardware components of the node, reprogram or simply destroy it. This in turn can lead to a loss of the sensitive data, bandwidth or failure of the whole network. Therefore, physical access to the nodes should be taken into account.

The lack of infrastructure. Ad-hoc and sensor networks do not depend on the established infrastructure. While this is one of the most important qualities of these networks, it also adds several problems for the design. The application of traditional security solutions, such as digital certificates becomes impossible. We can still make use of the secret key mechanism, assuming that a node possessing a valid key can be trusted. However, traditional key management methods require relatively high amounts of computational power and energy, which is typically unavailable for the ad-hoc and sensor networks.

The infrastructure-less nature of the network also implies that cooperation between all nodes has to be enforced. We can divide the problematic nodes into two groups: malicious (compromised) and selfish (refusing to cooperate). The cooperation is one of the main problems specific to ad-hoc and sensor networks and cannot be solved by cryptography alone.

Dynamic topology. In ad-hoc and sensor networks the topology constantly changes and has no static elements. By design, these networks expect that new nodes can be connected at any point in time and allows this behavior. This leads to a new security issue, as it creates the possibility for the attacker to insert malicious nodes into the network more easily.

2.1.3 Classification of Network Attacks

We classify attacks in ad-hoc and sensor networks based on their origin. The attack, originating from within the network, is defined as an *insider* attack. When network intrusion is performed from the external source, we define it as an *outsider* attack. A malicious, uncooperative node connected to the network belongs to the group of insider attacks. To show an example of outsider attack, we can think of a foreign entity, trying to gain access or disturb the activity of the network.

The attacks are then further divided into *passive* and *active* subgroups. In a scenario when intruder disturbs the network by directly modifying its behavior, the attack is classified as an active one. In case of passive attack, the intruder is monitoring, analyzing the network and typically stealing the confidential information [9].

The attacks can be classified further, based on the target of intrusion:

Availability and integrity: the attacker intends to prevent the network from providing its services. These types of attacks are focused on disturbing the normal operation of the nodes.

Confidentiality: the attacker is typically trying to obtain the content of transfered messages and analyze the network topology.

Integrity: the attacker aims to modify the data transmitted throughout the network. The malicious nodes can try to falsify the messages, illegitimately modify the content or resend obsolete packets.

Chapter 3

Trust Methods for Ad-hoc and Sensor Networks

In this chapter we provide description, classification and analysis of trust and related topics. The notion of trust itself has been defined in 1.4. This chapter includes an overview of the existing trust-specific problems and possible solutions, the analysis of the works related to trust in ad-hoc and sensor networks as well as comprehensive classification of those works and trust mechanisms in general.

The introduced trust-methods and models are works of their respected authors, but the analysis, classification and simulator of trust-methods are solely my work.

3.1 Overview

This section summaries the security problems and limitations of the ad-hoc and sensor networks as well as explains how trust methods can be used to solve those issues.

The security issues in ad-hoc and sensor networks, which were previously specified in 2.1, are similar to that of the traditional infrastructure-based wireless networks, but the conventional security methods cannot be applied due to several limitations 1.2.2. The ad-hoc and sensor networks, have to be protected from various types of attacks, which include: eavesdropping, analysis of network traffic, node replication and manipulation of packets. It is also vital to insure the fulfillment of several network requirements, such as privacy, authentication, authorization, accountability, integrity and many others 2.1.1. These issues apply to the traditional wireless networks as well, however there are several vulnerabilities specific to ad-hoc and sensor networks which has to be taken into account 2.1.2.

The research conducted in design as well as optimization of wireless ad-hoc network and sensor networks is still in its early stage. In order to fulfill potential of these technologies, the security issues in particular have to be addressed. The cryptography mechanisms, implemented in traditional wireless networks, are by themselves not sufficient to solve all security problems. The trust methods, on the other hand, can fill the security gaps and, in complex with cryptography, provide complete protection against unique security issues in wireless ad-hoc and sensor networks.

3.2 Analysis of Related Works

In this section we discuss and analyze the existing trust methods for ad-hoc and sensor networks.

There are several different trust methods and models proposed by researches up to now. However, the trust in wireless ad-hoc and sensor networks is still an open problem. There is no standardization or classification, and therefore no specialized books have ever been published. All trust methods presented in this thesis have been found in individual articles, journals and conference materials.

The described trust-methods and models are intellectual property of their respected authors, but the analysis, classification and developed simulator is solely my work.

3.2.1 Reputation-based Framework for High Integrity Sensor Networks

The analysis in this subsection is based on [1]. The author does not claim any rights on that work. One of the first works dedicated to the problem of trust in ad-hoc and sensor networks is a Reputation-based Framework for High Integrity Sensor Networks. This article describes the limitations of ad-hoc and sensor networks and promotes the use of trust management. The authors analyze unique security issues associated with these types of networks and explain why cryptography is not sufficient.

The article presents trust mechanisms based on the social science. The nodes belong to the community, which is the network. They share resources and have to contribute to the community as well. In the case of ad-hoc and sensor networks the main resources are the nodes themselves. They require other nodes to act as mediums and forward the packets for them. When a percentage of the nodes refuses to cooperate, the whole network can become nonoperational.

The nature of distributed systems requires each and every node to participate in the network activity and cryptography alone is not able to stop malicious behavior. The nodes can try to inject false messages or refuse to forward the packets. In order to prevent that, the reputation-based trust methods can be used. This security mechanism allows every node to increase the probability of successful data delivery through the selection of trusted nodes. If we assume perfectly secure environment, free from the attackers, the nodes might still generate false messages, due to the hardware malfunction. The trust mechanisms can detect and isolate such nodes.

The nodes generate the opinions of their neighbors in the form of trust values. The main sources are the previous interactions and recommendations from the neighbors. When the node successfully perform the action it was asked for, such as packet forwarding, its reputation will increase. In case it fails to do it, its reputation among the nodes will decrease accordingly. The authors also present Watchdog mechanism which allows the nodes to monitor such behavior.

3.2.2 A Security Framework with Trust Management for Sensor Networks

The analysis in this subsection is based on [4]. The author does not claim any rights on that work. The article describes trust method, based on the distributed trust model. According to the scheme, the trust value is formed from personal opinion of the evaluating node and recommendations of its neighbors. The personal opinion consists of an interactive behavior and cryptographic operation, which is defined as a packet validation. The cryptographic behavior is then divided into ordering, authentication, integrity and confidentiality. Each part of the schema is defined as a function, and can be evaluated with the help of proposed algorithms.

The ordering is defined as a function, which determines whether the forwarded packet originates from the base station and in addition to this verifies the packet freshness. The base station is typically a stationary computer operated by humans. It acts as a gateway for the ad-hoc and sensor networks and provides tools for network management. The presented trust model assumes that the base station is controlling the individual nodes in network by sending special validated messages. These messages are verified with the help of hash function.

Authentication and integrity is a function which provides protection against message modification and is implemented via authentication code mechanism.

Confidentiality is represented as a function, verifying whether or not it is possible to decrypt the cipher text of the message.

The result of cryptographic operation is a value which determines the validity of the forwarded message. The evaluating node stores this value to its local database for statistical analysis. The interactive behavior of the node consists of responsibility, positivity and cooperation.

Responsibility is defined here as a ratio of total number of replies to requests, sent by the evaluating node.

Positivity is a function which evaluate the ability of node to respond in the specified period of time.

Cooperation is provided with the help of sniffing. The evaluating node checks whether the packet was dropped or forwarded and stores the result in its database for future use. The authors advice to check for cooperation only if the values of responsibility and positivity are out of the required boundaries.

The trust scheme assumes that the personal recommendation is formed when node A receives the packet from node B, as well as when node A forwards the packet to node B. While the later has to be always taken into account when evaluating the trust value, the condition of the received packet is by itself insufficient. If we assume that node A receives the packet from the original source node D and finds out that the authentication, integrity and confidentiality were compromised, using the described mechanisms, it cannot really be used to determine which node is responsible for this. When node C receives compromised packet from node D, it will then have to lower the trust value of D. Unless the packet is dropped by node C at this point, it will then froward it to node B. And B, after evaluation of packet properties will decrease the value of node C, the same goes for node A in context of node B. In general, we can say that even though node M is a malicious node, which altered the packet in the middle of its way, all nodes which will forward that altered packet after M will be considered untrustworthy in a chain, due to the fact that nodes are able to interact with direct neighbors and not beyond that.

The problem is even more escalated if we assume that the first node which recognizes the altered/malicious packet will drop it and decrease the trust value of the source. It will lead to the situation where that first node is considered to be malicious as well, because it dropped the packet and violated the trust relationship by doing so. This issues have to be addressed and specific mechanisms for identification of initial source as well as prevention of false rumors spreading has to be found.

3.2.3 A Dynamic Trust Model for Mobile Ad-hoc Networks

The analysis in this subsection is based on [5]. The author does not claim any rights on that work. The article presents the trust model in which all nodes are initially authenticated and are assigned corresponding trust values. It is assumed that these values had been generated during previous interaction between the nodes and were restored when they reconnected. In case no trust information is available, in other words, no history of previous interactions, then nodes are assigned undefined trust value. This value is modified as new information becomes available, but at the beginning nodes have to take a risk when choosing next hop for packet forwarding.

The trust value is raised or lowered, based on the cooperation of the evaluated node. When it forwards the packets or drops them the trust value has to be changed accordingly. The authors also propose to integrate trust into the routing protocols in order to find the most trusted path in the network. The trust level requirement specifies the trust value needed to forward specific message. The most crucial messages have to be forwarded via the most trusted route, while less important ones can be transfered through path with lower overall trust value. To implement this, each node has to keep its own trust table which contains all its direct neighbors. When the node is asked to forward the packet, it has to look up the value in the table and compare it with the required one. In case there are no neighbors it should send the error message.

In this method, it is assumed that every node is equipped with an intrusion detection system, capable of monitoring neighboring nodes. It can be presented in the form of special hardware or software module and depends on the implementation. The intrusion detection system is able to detect malicious behavior and report it to nodes in the range. It is also assumed that the nodes, receiving such reports can combine them and filter out the false ones. In other words, the trust values locally stored for each neighbor of the node are constantly updated though previous interactions or references. If we compare it with the model described in 3.2.2, the main difference is that instead of asking for references each time the node performs evaluation, they are instead continuously transmitted by intrusion detection system. This allows the node to immediately make a decision, because it already has all available information.

This trust mechanism also introduces the concept of reference aging, which protects the nodes from obsolete recommendations. The aging mechanism requires the timestamps to be attached to all transmitted messages. When the node is combining the reports to calculate trust values, it transform the timestamps of each message into the factor which is then multiplied with original trust value.

The described method introduces several new concepts of the trust management in ad-hoc and sensor networks. They include and not limited to: integrated trust routing, message dependent trust value evaluation, the broadcast of reports via intrusion detection system and aging of recommendations. However, one of the problems that might affect the network performance is an overload of trust references. As we have described before, the nodes in ad-hoc and sensor networks have very limited power and it is advisable to turn off all hardware components, when they are not in use, to save energy. The continuous broadcast of the reports by intrusion detection is expected to deplete the battery of both source and destination nodes fairly quickly. The simulation of the method is required to get specific statistics.

3.2.4 Reputation-based Trust in Wireless Sensor Networks

The analysis in this subsection is based on [2]. The author does not claim any rights on that work. In that paper the intrusion detection system is used to monitor and observe behavior of nodes. It is refereed to as a Watchdog mechanism, and differs from the previous works by the concept of reference uncertainty. The uncertainty assumes that due to several factors, such as network attacks and faulty hardware, a percentage of reported references can be unreliable. To solve this problem, the node only uses provided references when there are at least X of them. In other words, if X or more nodes has sent the same recommendations, only then it should be considered legitimate.

The results obtained from the Watchdog mechanism and references are represented as binary events (p,n), where p and n are positive and negative outcomes respectively. The authors also take into account that Watchdog mechanism can generate false results due to hardware malfunction and malicious attacks. In order to address this problem, at least X occurrences of the event have to be observed.

The overall trust value is calculated using the conditional probability. This work can help us to improve the general concept of trust management by introducing new important functionality and solutions.

3.2.5 Probabilistic modelling and recursive bayesian estimation of trust in wireless sensor networks

The analysis in this subsection is based on [3]. The author does not claim any rights on that work. The article characterizes the problem of trust as uncertainty and proposes to use probability in order to compute trust values. The authors has introduced the Gaussian Trust and Reputation System which is based on the statistical approach and focuses on the continuous data transfer.

This paper also describes two main sources of information for trust evaluation in ad-hoc and sensor networks from the perspective of probability. The direct observations are obtained through continuous monitoring of behavior of the node. The second hand information is gained through the recommendations from the nodes. This data is then combined and used to evaluate the reputation of the node. The authors describe these observations as binary events and propose to use statistical methods for computations.

It is worth mentioning that even though the majority of researches propose to use the same sources of information, generated in the same way, the actual computational algorithms are different and the statistical approach is one of them.

3.3 Classification of Trust Methods for Ad-hoc and Sensor Networks

Based on the analysis of existing trust-methods in wireless ad-hoc and sensor networks, we were able to classify and summarize the main concepts and ideas

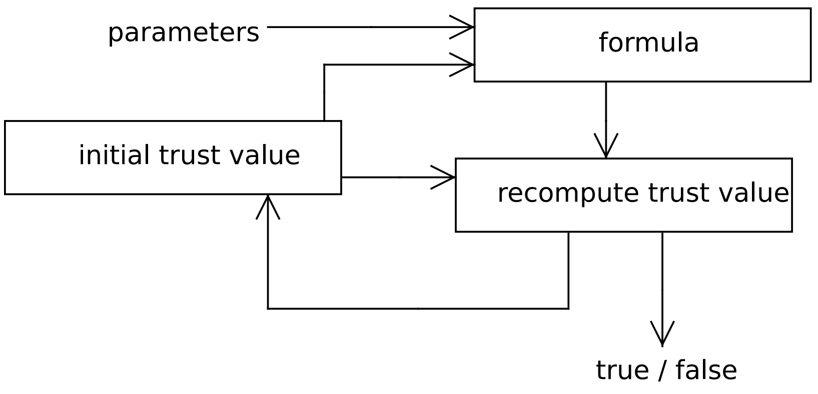


Figure 3.1: Trust function

behind this mechanism. Those findings have been presented and explained in this section.

The trust in wireless ad-hoc and sensor networks closely mimics the social relationships. The nodes are communicating with each other based on the trust levels, established through previous interactions. The main principle behind the trust is to insure that nodes will not perform malicious actions to disrupt the network.

Figure 3.1 describes trust as a function, which accepts input parameters, performs the computations based on specific formulas, modifies the trust value and evaluates the trustworthiness, which is either true or false.

3.3.1 The Main Aspects of Trust

We can define three main aspects of trust: formation, evolution and propagation.

Trust formation in wireless ad-hoc and sensor networks is a mechanism which assigns initial trust values for each node. There are two ways to implement this: restore previously generated trust values or assign default trust levels to all of the nodes.

It is possible to restore previously generated trust values of all nodes if they were a part of a trusted network in past. It is also possible to perform the same operation for individual nodes joining an existing network. There are two requirements for that: the network which previously contained the nodes has to be trusted; it has to be possible to obtain and keep the trust values of individual nodes after they have been disconnected. This method can be useful when an entire network or its part has to be relocated or reconnected.

However, it is typically more convenient to assign the default trust values. In this case, the nodes have to be willing to take a risk in order to initiate a communication with unknown entities.

Trust evolution is a procedure of trust value adjustment through continuous interaction between the nodes. The node can modify the trust value of another node based on the following factors: the previous direct interactions with that node and/or the recommendations received from neighbors.

In the infrastructure-less wireless ad-hoc and the sensor networks the most important for us is to insure that the nodes successfully forward the packets they are entrusted with. To distinguish between legitimate and uncooperative malicious nodes, its necessary to carry out periodic update of trust values. If a node misbehaves, its trust value should be changed accordingly, the same applies to the situation when the node successfully performs the required action. Depending on the particular trust model, it is possible to assign trust level requirements for messages. The most important messages, containing confidential information can have the highest trust value requirements, while not critical ones can be transfered through less trusted path. In that case the impact of unsuccessful packet forwarding can vary and must be taken into account when modifying trust values. We can justify it by the fact that a node takes a certain risk when it chooses the next hop for packet forwarding. The greater that risk is, the more should it affect the trust.

There are two factors which play the major role in trust evolution: the *direct communication* with the node and *third party references.* In *direct communication* the nodes are able to modify the trust value of the neighbor based on their own experiences, while *third party references* provide the trust values generated by other nodes.

Trust propagation is a process of trust value spreading between neighboring nodes through the exchange of references. The propagation is directly connected to trust evolution and is one of the main factors affecting it. When the node is trying to choose a neighbor for packet forwarding, it has to know its trust value. If we assume that local trust information is available and sufficient,then the node can safely use it. However, when there is no such data, the only way to evaluate the trustworthinesses of the node is to ask for recommendations. The node which previously interacted with the target node can provide such recommendations in the form of reference packets.

3.3.2 Trustworthiness Evaluation

When node A determines whether it can trust node B, the trustworthiness of node B has to be evaluated. Figure 3.2 represents the process of trustworthiness evaluation in details, which is an integral part of trust evolution.

To make it clear, we assume that the activity starts when node A selects a neighbor for evaluation, we call it node B, and ends after the trustworthiness has been determined. Node A has to check for the locally stored trust value of node B. Depending on the trust value availability there are two possible ways to proceed: broadcast reference request to get the recommendations or directly use the available value.

If node A broadcasts reference request, it is typically configured to wait for the specified time interval, so that all of the neighboring nodes are able to react. Node A then collects all replies, combines them and assigns a trust value based on the results. This newly generated trust level replaces the local one, previously stored in the node.

Assuming the trust value has been generated from references or the local trust value is available, the node will then compare it with default or implicitly specified required trust level. The trustworthiness of evaluated node B will be determined, based on the result of that comparison. If we assume that no replies had been received, then the local trust value would remain the same and the evaluated node B would not be trusted due to insufficient information. The default trust value in the diagram represents a number, which is considered to be high enough to trust a node and is shared across the whole network.

In practice, if node A cannot trust the evaluated node B, it will then repeat the process until a trustworthy node is found or no more neighbors are left. In case of later, the node will have to take a risk or abort the communication. The exact computational formulas and algorithms have to be provided by trust methods and their corresponding models. The specific details are heavily dependent on the implementation and can vary.

3.3.3 Evolution of Trust

As has already been mentioned, the trust evolution is a process of trust value modification through direct interaction between the nodes or provided references. In the context of ad-hoc and sensor networks, the only important operation for us is packet forwarding. Thats why we describe the trust evolution in figure 3.3, assuming that node A intends to forward a packet via node B. In practice, if the forwarding is replaced by any other action, the rest of this activity diagram will remain completely the same.

First of all, node A has to chose a neighboring node B and evaluate trustworthiness, as shown in 3.2.

If the chosen node has been determined untrustworthy, the whole process has to be repeated. Otherwise, node A will use trustworthy node B to forward a packet. Node A will then monitor that packet and check whether it was successfully forwarded or dropped. When node A forwards the packet via node B, it can then shortly after change the normal operation mode to promiscuous. While in this mode, node A is able to perform packet sniffing, which in turn, allows it to monitor the behavior of node B.

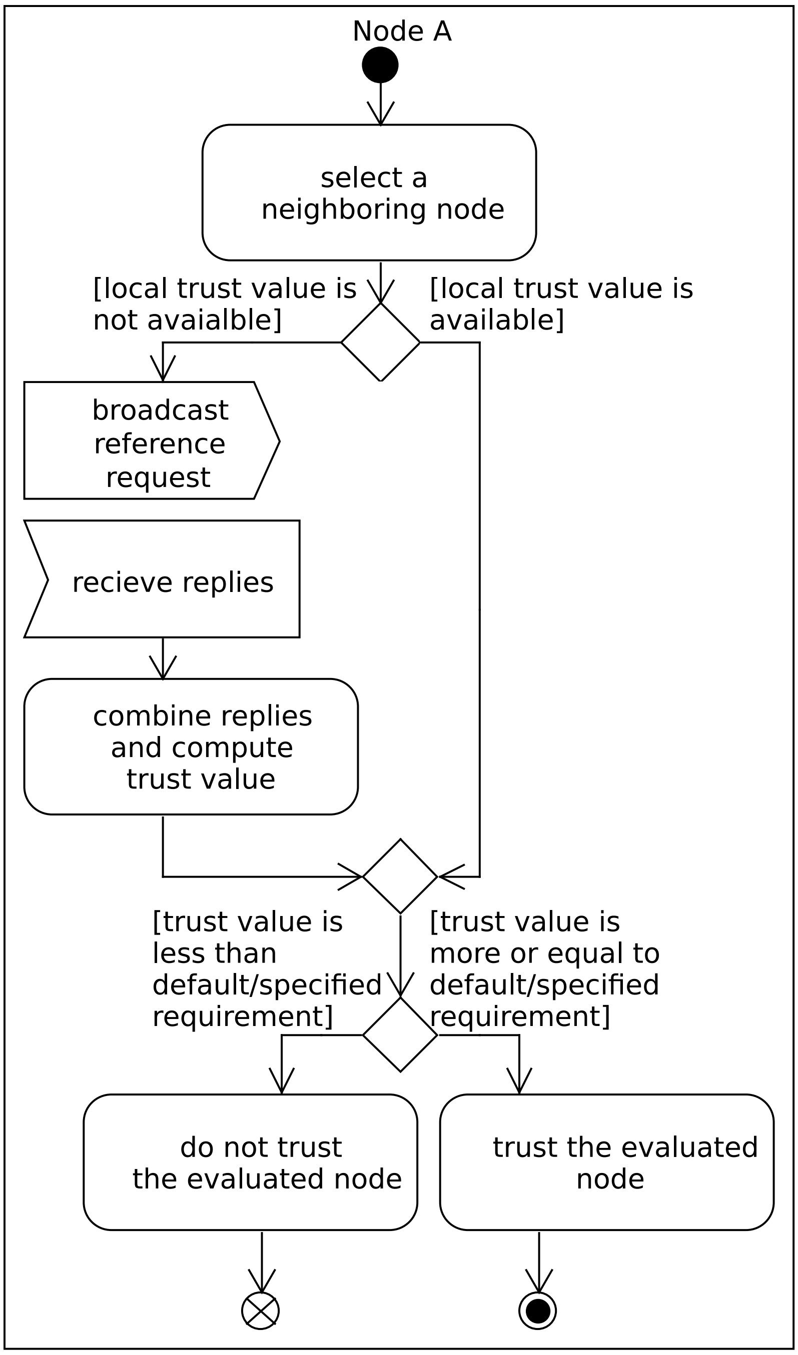

Node A
select a neighboring node
[local trust value is not avaialble]
[local trust value is available]
broadcast reference request
recieve replies
combine replies and compute trust value
[trust value is less than default/specified requirement]
[trust value is more or equal to default/specified requirement]
do not trust the evaluated node
trust the evaluated node

Figure 3.2: Activity diagram of trustworthiness evaluation

Based on the observed behavior of node B, there are two possibilities: node A will increase the trust value of node B and complete the activity; node A will decrease the trust value of node B and repeat the whole activity from the beginning. The amount of trust value to be increased/decreased can be the same for all actions/types of packets, or depend on the importance of that action/packet. Node A will have to repeat the whole process in case of unsuccessful packet forwarding, due to that fact that originally node A had intended to perform this action and it ultimately failed. On the other hand, the activity will be completed, if the packet is successfully forwarded. Assuming that previous operation has failed, we can also imagine that when node A selects a new node to forward a packet, it can select node B again. It can be explained by the fact, that node B can still have high enough trust value to be considered a trustworthy node.

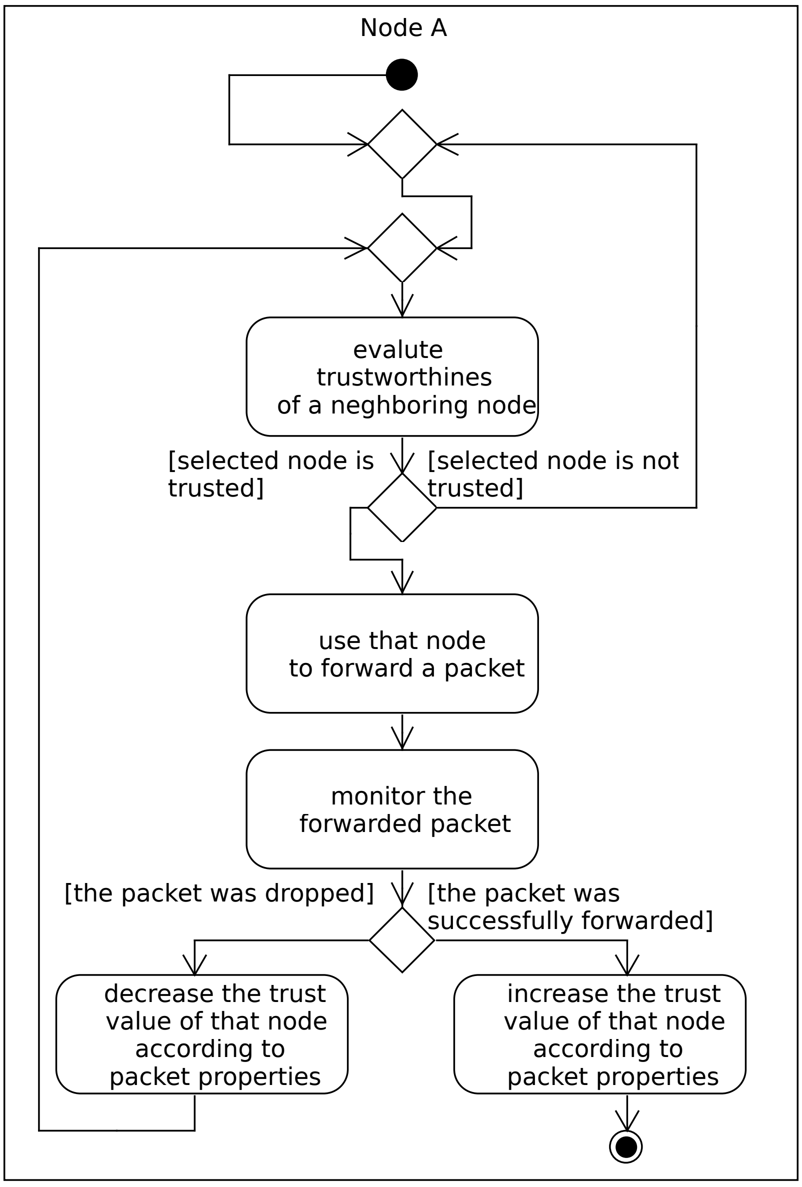

Node A
evalute trustworthines of a neghboring node
[selected node is trusted]
[selected node is not trusted]
use that node to forward a packet
monitor the forwarded packet
[the packet was dropped]
[the packet was successfully forwarded]
decrease the trust value of that node according to packet properties
increase the trust value of that node according to packet properties

Figure 3.3: Activity diagram of trust evolution

Chapter **4**

Simulation of Trust Methods for Ad-hoc and Sensor Networks

In this chapter we provide the analysis of existing network simulators, the advantages and disadvantaged associated with them. We also describe the usefulness and practical applications of the network simulation software as well as common problems.

4.1 Overview of Network Simulation

This section describes the basic principles of network simulation, its advantages in comparison with other testing tools and existing problems.

The simulation software is highly valued for its contributions to the development of new network technologies. The theoretical analysis of the proposed systems is a good estimation of the overall performance and functionality, however its usually not sufficient. The simulations are vital for providing greater details of the designed network and are performed before real-live deployment. They allow to recreate flexible, manageable environment which corresponds to the real live conditions up to some degree. The simulation software provides the means to observe the behavior of individual nodes or network as a whole. These observation are used to generate statistical data, which estimates the expected behavior of the network. It is typically infeasible to perform the tests with the real devices each and every time the design is changed. It can be explained by the following factors: the cost of devices can be too expensive; it can be required to test a specific module or function of the device, not taking into account the rest of it; the performance of the hardware and software greatly depends on the scale of the network. The simulation allows to perform repeated tests, adjust the scale of the network and modify the core mechanics in a fast and convenient way.

The correctness of obtained simulation results, in its majority, depends on the end user. The produced data can be unreliable, if the given simulator directives are incomplete or ambiguous. To perform the correct simulation, its required to insure that theoretical model is implemented in the comprehensive way and no important details are missing. In other words, the reliability of the conducted experiments cannot be guaranteed by the simulator alone, but also depends on the settings of the model.

4.2 Analysis of Existing Simulation Software

There are several networks simulators to choose from. In this section we describe the most popular ones, analyze their advantages and disadvantages and compare them with each other. This section also explains why the existing network simulation software is not sufficient for trust evaluation.

ns-2 [12] is one of the most frequently used simulation software. It was initially started by an open-source community in 1989. The simulation suit itself is programmed in C++, while the user's scripts are written in OTcl. Otcl is an object oriented version of Tool Command Language (Tcl), designed by David Wetherall at MIT. The integration of OTcl language into the ns-2 simulator has made it easier for the users to design simulation scenarios. The source code of the simulator does not need to be recompiled every time in order to perform the tests, instead, its enough to modify the simulation script, which is given to the application as input file.

The exceptional popularity as well as open-source nature of ns-2 simulator brings several benefits to the project, such as the continuous introduction of new plug-ins. These extensions enable the simulator to support new types of networks and additional functionality. However, it also introduces new issues and the most notable of them is complexity. The simulator is written as a solid-state application, without possibility to divide it into corresponding modules. It has to support the whole complex of functions, introduced in the official releases, and be backwards compatible. Theses factors together make the ns-2 simulator too hard to operate for unexperienced users. The available functionality can be considered to be redundant or overwhelming. The simulator is designed for general purposes network simulation. While this is certainly a great advantage, it also means that the simulator is not optimized for any particular type of network. The official package does not include statistical tools, which makes it harder to collect and analyze the data.

ns-3 [13] simulator is an updated version of ns-2. We analyze it separately, because its very different from its predecessor. Both versions of the simulator are widely used, depending on the individual preferences. The core of ns-3 simulator is written in C++, just like ns-2, however, there is a big difference in the simulation scripting. The ns-3 simulator allows to create testing scenarios in standard C++ or python language. It can be explained by the fact, that the ns-3 simulator is relatively recent project and nowadays its not a problem to frequently recompile the sources. The core of ns-3 simulator is left open for the possibility of adding new extensions. The ns-2 simulator uses OTcl language for scripting, while ns-3 supports only C++ and python. In other words, ns-3 simulator does not support the testing scenarios designed for its predecessor, ns-2. The existing simulation scenarios have to be rewritten from scratch, this in turn lead many researches to use ns-2 instead. The ns-3 introduces several new tools, but the majority of upcoming features is still in development.

GloMoSim simulation software has received a considerable amount of references from researches and is programmed in Parsec. In addition to functionality introduced in ns-2, it also supports testing in more complex parallel environments. The simulation scenarios are divided into subprocesses or threads, run in parallel. The parallel execution is one of the main advantages of GloMoSim and was later adopted in the new simulation projects. The GloMoSim used to be the second most attractive simulator for the researches after ns-2, however, the development of GloMoSim has been discontinued. It also does not provide sufficient documentation materials, which makes it difficult to use.

QualNet [14] simulator is a commercial project based on the open-source GloMoSim. The original design of GloMoSim has been heavily improved, and in addition to this, several new features have been added. The QualNet allows the user to perform simulations with additional set of available network protocols. The graphical user interface is also provided and makes it easier to prepare the testing environment. The QualNet, as a commercial software, provides comprehensive user manual and periodically receives new updates.

Opnet [15] is also a popular commercial software, just like QualNet. It provides the functionality to simulate discrete events and is typically used for testing of traditional types of networks. The base package includes a set of predefined models for both wired and wireless networks. In order to perform the simulation, one of the models has to be used. These templates are designed and verified by the developer for end users. However, it is also possible to create custom models with the help of graphical tools. To define a new model, the user has to specify a finite state automaton and describe all states with the help of Proto-C programming language. The creation of the custom models is certainly a great feature, but it can be too complicated for unfamiliar users.

OMNeT++ [16] is a discrete event driven simulation software programmed in C++. OMNeT++ was originally designed as a general-purpose simulator, but its primarily used for networks. The INET package enables support of common network protocols, while Castalia is able to simulate several types of wireless ad-hoc networks. The OMNeT++ consists of a set of separate modules, which can be included together to satisfy the user's requirements. The separation of the modules makes the simulation suite more flexible and easier to use. The OMNeT++ community also provides comprehensive documentation and tutorials which are extremely useful for both new and experienced users. A network description is a special high-level programming language designed specifically for the creation of simulation scenarios in OMNeT++.

Jist [17] is one of the relatively new simulators written in Java language. The simulator itself is structured into modules, which correspond to different functionality and network types. In Jist its possible to simulate the wireless ad-hoc network with the help of Swans package. The simulation scenarios have to be written in Java programming language. The nodes in the network are represented in the form of Java classes and have to be specified by the user. Its also possible to prepare simulation directives in the configuration file and pass it as input parameter for the application. Jist simulator also supports execution of parallel environments for improved performance. One of the drawbacks of this simulation software is the lack of predefined models. However, the main problem is the suspension of the simulator development. Its unclear whether or not the project will be continued.

In this thesis we have presented and analyzed the most popular network simulators, which were arbitrary chosen. There are several advantages and disadvantages associated with every simulator. The ns-2 is currently used by the majority of researches, but it lacks the modular structure and the creation of simulation scenarios can be very complicated for unexperienced users. The ns-3 simulator has been improved over its predecessor, however its still in the early production stage. The ns-3 has been designed and structured with the future extensions in mind, but at this point the considerable part of the functionality has not been implemented yet. It also requires the user to be familiar with C++ or Python languages, which take considerable amount of time to learn. The GloMoSim has introduced the parallel environment simulation, however the project was discontinued and the existing user documentation is insufficient. The commercial simulator QualNet was built on the core of GloMoSim and includes additional features. While QualNet has been improved over its open-source predecessor, it still has several drawbacks, such as inability for the user to implement new custom modules. The Opnet simulator is another popular commercial software. The drawbacks of Opnet are similar to that of ns family, as it requires the user to be experienced with finite state automatons and Proto-C language. The OMNeT++ is a discrete even driven simulator, which supports wireless ad-hoc networks with the help additional packages. The OMNeT++ is flexible and provides comprehensive documentation, but it requires the user to learn network description programming language. The simulator itself was not originally designed for the network simulation and, in turn, creates additional issues. The Jist simulator by itself does not supports ad-hoc networks and requires additional packages for that. It also forces the user to learn a Java programming language in order to write simulation scenarios, which can be a problem.

Chapter 5

The New Simulator of Trust Methods for Ad-hoc and Sensor Networks

In this chapter we introduce the brand new network simulation software, designed specifically for the trust mechanisms in ad-hoc and sensor networks.

The problems described in 4.2 are common for almost all existing simulation software, but the testing of trust mechanisms for ad-hoc and sensor networks is the most important for this thesis. The relatively new wireless ad-hoc networks and its subtypes have little to no support in the modern simulators. In the majority of software they are 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.

The simulation of trust methods is especially problematic. The trust as an open research problem has not received enough attention from the software developers. In fact there are no network simulators which have sufficient support of this concept. When the researches propose new trust methods, the testing is typically performed with the help of improvised simulation software or not at all. The results produced by these applications are often not optimized and unreliable. The new methods require a new simulator for the testing, which creates an overheard for the research projects and makes it hard to collaborate.

5.1 Technical Design Choices

In this section we describe the choices of technical tools used for implementation of the application, such as programming language, graphical user interfaces and tools for drawing.

5.1.1 Programming Language

The simulation software for trust methods in ad-hoc and sensor networks has been written entirely in C++ programming language. The choice of the language has been made, based on several criteria:

Popularity The C++ is extremely popular among leading IT corporations and is commonly known to the majority of developers. This in turn, enables efficient maintenance of the application and possibility of extension in the future. The C++ programming language has been introduced for the first time in late 1983. It has been greatly improved ever since, through the course of several development iterations. Each iteration of the programming language introduces new functionality and provides backward compatibility for the previous ones. The International Organization for Standardization has been documenting each new version of the C++ since 1998, with the latest standard published in 2014 under the name ISO/IEC 14882:2014[18].

Functionality The C++ programming language is very flexible and provides low-level interfaces for the communication with kernel of the operating system and memory manager. At the same time, it also supports high-level abstraction, which makes the language universal and suitable for the development of almost any type of application or operating system.

Cross-Platform The open source communities as well as private companies have shown a great interest in C++ programming language. The native C++ libraries have been written for almost all modern operating systems, which allows developers to adapt their source code for different platforms with little to no changes to the original program.

Those qualities has made C++ programming language a very attractive choice for the development of our network simulator. The software has been developed with future improvements in mind, which can be designed more efficiently with C++ programming language. The application is not bound to one specific platform, but can be instead compiled for most of the modern operating systems. The C++ language has allowed us to develop comprehensive graphical user interface without penalty to overall performance.

5.1.2 Graphical User Interface and Drawing Tools

The application supports advanced graphical user interface, developed with the help of GTK+3.0 libraries[19] for C++. The GTK+3.0, originally developed as widget toolkit for the GIMP image editor, is a cross-platform GUI development kit. The GTK+3.0 has been chosen for the implementation of our simulation software mostly due to the same reason as C++. The GTK+3.0 libraries has been written not only for the C/C++ but also for almost all of the modern programming languages, which include and not limited to: Javascript, Perl, Python, Vala, D, FreeBASIC, Haskell, Java, Lua, Pascal, Ruby. The GTK+3.0 is able to run on Windows, OS X and Unix-based operating systems, while using native interfaces, such as X11. These factors make GTK+3.0 kit almost universal in nature, which in turn influenced our choice.

The GTK+3.0 provides basic widgets, such as text entry fields, buttons, key capture mechanism, packaging of widgets into containers such as panes, boxes etc. GTK+3.0 allows simple and efficient configuration and layout management for its widgets, which can be achieved with GTK+3.0 builder tool. This tool allows to specify the GUI structure in XML language and save it to a separate file. The application is able to read the GUI specifications during run time and configure itself accordingly. The GUI can be easily changed by modifying the xml file only, the recompilation of the executable file is not required. The separation of source code and GUI definitions allows to efficiently manage the layout and packaging of widgets. This is especially important if there is a sufficiently large number of GUI elements.

One of the considerable design issues that we came across during the development process was drawing of network components. It was required to find a way to represent the nodes, the links between them, the communication range, energy consumption and others. The elements had to be drawn procedurally, that is, they had to be generated during the run time. The drawings also had to be produced as vector graphics and be able to react to dynamically changing network state. In addition to it, the application was designed to support sufficiently large amount of components, which had to be generated and displayed at the same time.

The first idea during the development process was to generate the static images with the help of gnuplot application and then display them for the user to create the effect of animation, but we have found several problems with this approach. The gnuplot is able to generate vector graphics, however the subsequent calls to the external application could dramatically decrease the performance. The gnuplot itself requires an additional data to specify the image, and even with the help gnuplot C++ libraries, it still would have created an overhead.

The solution to this problem was found in the cairo graphics library[20]. It supports vector image generating, runs on the most of the modern platforms and performs very efficiently, even for the large amount of data. The cairo graphics library has fulfilled all our requirements for the drawing mechanisms, and in addition to this, cairo can be seamlessly integrated into the GTK+3.0 GUI tool kit. The gnuplot itself is using cairo libraries to generate the pdf and png files, starting from version 4.4 and later[21]. It is clear that the direct calls to cairo from within the application are much faster. It also allowed as to generate completely custom graphic elements, which would have been hard to impossible with the gnuplot.

These special features, provided by GTK+3.0 together with cairo libraries, are the major reasons behind our choice of development tools for graphical user interface.

5.2 Application Design Choices

In this section we describe the choices made during the design stage of application from the perspective of functionality and principles of operation.

5.2.1 The Description of Simulation Scenario

The simulation scenario is a key part of the software. We define simulation scenario as a set of ordered functions, which specify the network topology and behavior of its individual components. The simulation scenario has to be created by the user, and the user alone is responsible for its correctness.

There were several different approaches to the implementation of simulation scenario. The initial idea was to represent the scenario as a separate file, prepared in advance by the user and given to the application as input parameter. In order to realize that, it was required to create a parser for that file as well as simulation directives and keywords. The parser had to be implemented as a part of our network simulator and be able to recognize the custom directives. The application then would have to construct and perform the concrete commands based on them. The directives and their format could be taken from the existing languages such as xml, however it was better to create the brand new ones. This can be justified by the fact, that the parsing mechanism could be simplified if it was accepting only a small set of custom keywords and directives. At the initial design phase the application was supposed to be based on those concepts, however it then became apparent that the overhead was to high. It would have been problematic to define directives which would be simple enough for the user to understand and at the same time provide required functionality. That problem could be solved, but the real issue was found the in parser and related to it mechanisms. The implementation complexity of these tools proved to be close to that of a simplified compiler for the programming language. It required the special grammar as well as comprehensive lexical analyzer, and even if it had been achieved, it would still greatly slow down the application.

The solution to the problem was found shortly after and was inspired by the design of ns-3 software, described in 4.2. The ns-3 accepts simulation scenario in pure C++ or Python language, compiles it and links with the rest of source files. In other words, the simulation scenario is a source file, which has to be compiled together with the simulator itself. We have decided to use the same concept and represent the simulation scenario as a C++ source file, but we also introduced very unique concepts specific to our simulator only.

In our simulator of trust methods for ad-hoc and sensor network, the simulation scenario is represented in the form of a C++ source file, which is then compiled and linked to the rest of application binary files. To speed up the compilation process, the simulator was developed in such a way that it allows the user to compile scenario only, and then link it to the existing binaries, therefor eliminating the need for recompilation of the core source files every time. Before the initial start of the application, the user have to compile the core files with the help of defined Makefile and make utility. The core source code is then saved as binary object files and kept through out the operating time. To use a specific simulation scenario, the user have to save the file using predefined name and compile it with make utility and defined Makefile. The Makefile if configured to automate the whole process, which makes the compilation simple and efficient.

5.2.2 The Inner Structure of the Simulation Scenario

The simulation scenario consists of three functions, which have to be specified by the user. We will omit the concrete names of functions and data structures, as they are not important for us. The user is able to specify any specific trust methods through these three main functions as well as general network behavior and topology. It makes the software extremely flexible and simplifies the process of defining new trust methods and network configuration. These main functions are then called by the simulator, which also implies that incorrectly specified simulation scenario will lead to the possible crash of application. Its the user's responsibility alone to insure that the simulation scenario has been defined correctly, and all of the required parameters have been set.

The user is able to freely use additional high-level predefined functions to create a simulation scenario. They have been prepared to be called by user through out the whole scenario. Its assumed that the user is not supposed to have any knowledge of the inner structure of the simulator, which is achieved with the help of those high-level functions. They have been designed to be as much user friendly as possible, so that even unexperienced C/C++ users will be able to grasp the meaning behind them.

The user is allowed to define any number of additional functions and data structures inside the simulation scenario, or in external file, included into the simulation file. They are allowed to be used in the same way as predefined ones, and can possibly help the user with specification of the simulation scenario.

5.2.2.1 Network Initialization Function

The first function is used to configure the network topology and specify the directives for the simulator. We call this function network initialization. The user have to set the basic configuration with the help of specialized functions. These parameters include and not limited to: wireless transmission frequency, minimum and maximum allowed power, data size and data rate, the total number of nodes used in simulation, the amount of energy individual nodes have, their positions in the network, the reliability of the nodes, the trust relationship between nodes, the directives for the simulator to send virtual messages and many others.

The network simulator allows the user to manually define every element and parameter of the network to simulate the required topology very precisely. In case the user requires to generate a topology for statistical analysis, its possible to use predefined simulation functions for that. They support the creating of network layout automatically, generating it in a random way, but still insuring that the positions of the nodes do not overlap and that they are all interconnected. There are three different functions for automatic nodes' layout generation, which can be further configured with the user parameters. The layout functions allow to create any type of network topology automatically or manually, or generate random topology and then modify its part manually to provide greater details.

The simulation software also supports the Save and Load operation for network topologies. Its possible to save any topology to a file or load the previously saved one. These function provides user with the ability to run tests in the completely recreated environment.

The user is also able to setup the logs of application in network initialization function. The simulator supports 3 logs with different levels of information detail. Those logs are: full log, which provides complete information about every action the simulator takes, every event, and all changes of parameters; summary log, which provides information about source of the message, destination, taken route and the result; the statistics log, which only shows the basic overall values such as number of undelivered messages, the amount of total spent energy and others. Its also possible to efficiently setup the logs with the help of predefined functions. The user is able to turn off any of the logs in any combination to improve the performance and save memory. Its also possible to automatically save the logs on disk in any combination.

The previously mentioned reliability of the node, specifies the probability of success for all of operations, performed by the node. This parameter is extremely important for the simulations. The nodes with very low reliability are considered to be malicious or uncooperative ones, which cause damage to the network. However, its also possible to decrease the probability for a very small percent to simulate the real environment challenges, such as interference.

The user is able to simulate the process of message delivery in the network. It is possible to explicitly specify the source and destination of the message, and the amount of subsequent messages to be sent. The network simulator also allows to specify one or both, source and destination node, to be random, which is perfect for testing. The nodes will not be randomly chosen one single time, but instead a random node will be selected for every subsequent message. Its better to explain this property on a simple example. The user specifies random source, random destination and n subsequent messages to be sent.

For every message (we have n of them), both source and destination will be randomly chosen, which implies that in practice, with sufficiently large number of messages, every single node will at least ones be a source or a destination or both.

5.2.2.2 Trustworthiness Evaluation Function

The trustworthiness evaluation function accepts two parameters, nodes A and B. It performs the evaluation and determines whether or not A trusts B. The exact process of evaluation has to be specified completely by the user and it is expected that the function will return a boolean value, indicating the result of this evaluation. The function is defined between each pair of nodes, which form a route from source to destination. It is called every time, when the message travels from one node to the next one on route.

This function has a free form which makes it very flexible. The user is able to define any specific trust method inside this function. The only requirement is to insure that the function returns a boolean value. In practice, the user may, but not required to, use predefined functions to read the local trust value of nodes, compare it to a defined threshold, request for the references if needed and make a decision based on all available information.

5.2.2.3 Trust Modification Function

The trust modification function is used to define the actions, that have to be taken after the message has been successfully delivered to the destination, or undelivered. This function is defined in the similar way as trustworthiness, that is, it is called on every pair of nodes, which form the route from source to destination.

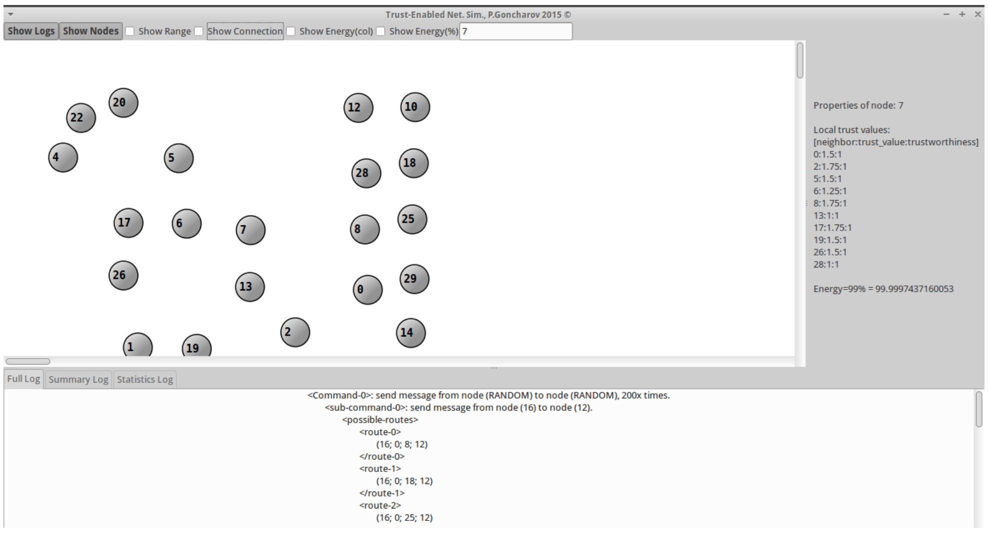
The users has access to the data structures of both nodes, source and destination, as well as the result of operation. In this context we are specifically interested in message forwarding. The user is supposed to decide how trust value, and possibly additional parameters will change in case of success or failure.

5.3 GUI Description and Examples of Usage

This section describes the graphical user interface of the developed simulator, the functionality behind it and the examples of usage.

The graphical user interface of our simulator provides comprehensive representation of the state and processes of the network. You can see an example of normal program operation on fig. 5.1 and fig. 5.2. At the top of the window, the user can find Show Logs and Show Nodes toggle buttons as well as several check buttons. The Show Logs toggle button can show/hide the lowest graphical panel containing the logs, helping the user to focus on either the topology or logs and save screen space. The Show Nodes toggle button allows the user to enable or disable the drawing of the network nodes completely. This function has been added to help the user boost performance when simulating very large topologies. The user is able to hide the nodes and focus on different part of the interface, or hide the rest and show the nodes. It helps to improve the performance in critical situations. The various check buttons allow the user to toggle on/off the drawing of individual network components such as wireless transmission range, the virtual representation of links between nodes, the percentage of energy remaining and the coloring of the nodes, based on the percentage of the remaining energy. The coloring function allows the user to visually grasp overall state of remaining energy in all nodes. The nodes gradually change their color from bright green to bright red, representing 100% - 0 % of the remaining energy. This function can be completely disabled, in the same way as all others.

The text field in the top panel is able to accept the user input in the form of unique identification number of the node. When the input is received, the application will automatically show the panel at the right side of the window, containing all the available information about the specified node. This panel shows all of the one-hop neighbors of the node; their trust values in opinion of the specified node; the trustworthiness, evaluated with the help of user defined function; the remaining energy of the node in percentages as well as the exact value of remaining energy in joules. The user is able to close this panel by pressing 'Escape' button. The middle of the window is used for the drawing of the network topology. The middle panel allows the user to see graphical representation of the nodes and related elements, which have been specified earlier in the text. The lowest panel of the window contains the tabs, each tab representing one of the logs. The logs will or will not appear on this panel based on the configuration of the user, specified in the simulation scenario. The lowest panel will also display a loading screen, if the data has not been processed yet.



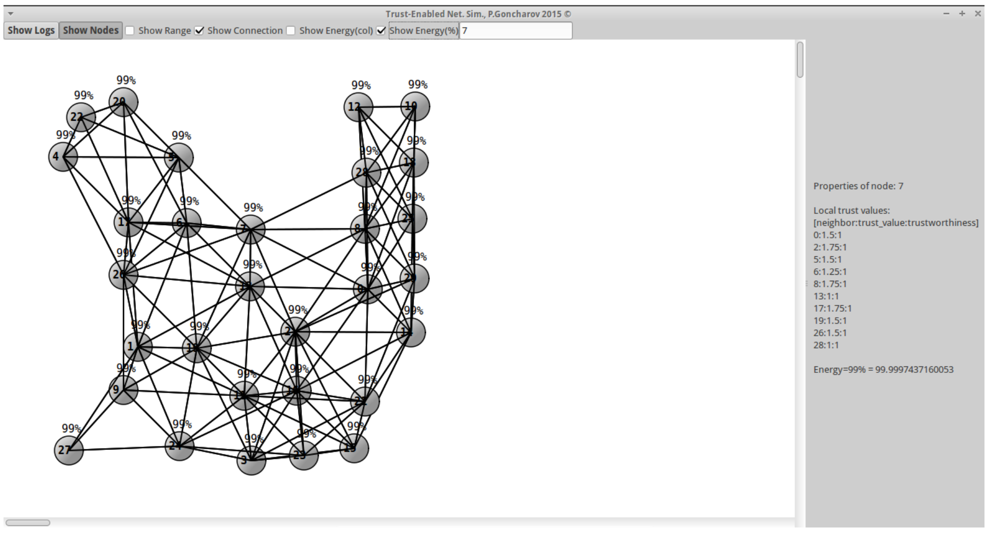
```
Properties of node: 7

Local trust values:
[neighbor:trust_value:trustworthiness]
0:1.5:1
2:1.75:1
5:1.5:1
6:1.25:1
8:1.75:1
13:1:1
17:1.75:1
19:1.5:1
26:1.5:1
28:1:1

Energy=99% = 99.9997437160053
```

```
<Command-0>: send message from node (RANDOM) to node (RANDOM), 200x times.
   <sub-command-0>: send message from node (16) to node (12).
      <possible-routes>
         <route-0>
            (16; 0; 8; 12)
         </route-0>
         <route-1>
            (16; 0; 18; 12)
         </route-1>
         <route-2>
            (16; 0; 25; 12)
```

Figure 5.1: GUI example 1



```
Properties of node: 7

Local trust values:
[neighbor:trust_value:trustworthiness]
0:1.5:1
2:1.75:1
5:1.5:1
6:1.25:1
8:1.75:1
13:1:1
17:1.75:1
19:1.5:1
26:1.5:1
28:1:1

Energy=99% = 99.9997437160053
```

Figure 5.2: GUI example 2

Conclusions

The infrastructure-less self-organizing ad-hoc and sensor networks can be very useful for operations in remote, physically unaccessible or dangerous areas. Some common examples are: enemy territory, ocean floor, uninhabited forests and mountains. At the same time, it is possible to use these networks for monitoring patient's well being in a hospital, vehicle traffic management, alarm systems, parking lots and many others. The ad-hoc and sensor networks share many common problems with other types of infrastructure-less and infrastructure-based networks, however there are several particular issues. These issues include and are not limited to: the enforcement of cooperation between nodes, efficient energy consumption and isolation of compromised nodes. While some of these problems can be resolved with the help of cryptography, it does not provide the complete solution. The trust-methods can ensure the cooperation of nodes, identify and isolate malicious nodes and, at the same time, do not require very high amount of energy and computational power.

This work described the security issues, specific to the ad-hoc and sensor networks and justified the trust as a possible solution. It was able to analyze several proposed trust methods for ad-hoc and sensor networks, to emphasize their advantages and disadvantages. Based on the analysis of concrete methods, the comprehensive classification and generalization of trust mechanisms, applicable to all trust methods were made. This work identified the particular features of the existing methods and summarized them in a single complete trust system. The structure of this system was based on the three aspects of trust, proposed by the author: formation, evolution and propagation. Those individual components were defined and described in great detail. The trust system itself was generalized as a function, independent from the concrete trust methods. The simulator of the trust methods for ad-hoc and sensor networks was developed. The simulator was designed to be very flexible and completely independent from the specific trust methods. The new methods can be easily defined and tested. The developed simulator is able to not only perform the computations, but also visualize the topology and properties of nodes for the user. The simulator is highly configurable and allows the user to adjust it for specific needs. The source code is completely platform independent, which makes it possible to compile the application for Windows, Unix, OS X operating systems and others. The simulator is able to operate with very large topologies, however it requires a relatively high computational power, as expected.

The ad-hoc and sensor networks have to be secured in order to properly function, there are however no established ways to achieve that. The trust methods can be used to solve this problem, which makes them extremely important for the development and mass deployment of this type of networks. The analysis, classification and generalization of the trust methods together with the developed simulator can be used to evaluate and improve the existing trust methods as well as to create the new ones.

This research was performed based on the wireless flat ad-hoc and sensor networks, however it was shown that the trust is independent from the means of connectivity, structure of the nodes and network topology. With the future research, it is possible to extend a general trust system to support hierarchical topology as well. In the future this can be achieved with the help of separate trust relationships layers for each type of the node. The simulation software was already developed with the ability to work with wireless flat ad-hoc and sensor networks in mind, however it was designed to be easily expandable. The simulator is already able to operate with the wired networks and has all prerequisites to add the support for hierarchical network topology in the future. The simulation software can also be extended to support different trust level requirements for messages as well as trust-enabled routing, which is already a part of the simulator, but is currently not accessible for the user directly.

Bibliography

[1] Ganeriwal, S.; Srivastava, M. B. Reputation-based Framework for High Integrity Sensor Networks. In *The 2nd ACM Workshop on Security of Ad-hoc and Sensor Networks*, Washington DC, USA, 2004.

[2] Chen, H.; Wu, H.; Zhou, X.; et al. Reputation-based Trust in Wireless Sensor Networks. In *International Conference on Multimedia and Ubiquitous Engineering (MUE '07)*, Seoul, Korea, 2007.

[3] Momani, M.; Challa, S. Probabilistic Modelling and Recursive Bayesian Estimation of Trust in Wireless Sensor Networks. *Bayesian Network*, 2010.

[4] Yao, Z.; Kim, D.; Lee, I.; et al. A Security Framework with Trust Management for Sensor Networks. In *The 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks*, 2005.

[5] Liu, Z.; Joy, A. W.; Thompson, R. A. A Dynamic Trust Model for Mobile Ad-hoc Networks. In *The 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS '04)*, 2004.

[6] Arora, B. Role of Key Management for Detection of Malicious Nodes in Wireless Sensor Networks. *TECHNIA âĂŞ International Journal of Computing Science and Communication Technologies*, volume 5, no. 2, 2013.

[7] Ezeife, C.; Ejelike, M.; Aggarwal, A. A Sensor-Based Online Mining Wireless Intrusion Detection System. In *International symposium on database engineering & applications IDEAS*, 2008.

[8] Estiri, M.; Khademzadeh, A. A Theoretical Signaling Game Model for Intrusion Detection in Wireless Sensor Networks. In *14th International*

Telecommunications Network Strategy and Planning Symposium (NETWORKS), 2010.

[9] Anderson, J.; Shafer, L.; Nahavandi, S.; et al. (editors). *Mobile Ad Hoc Networking Cutting Edge Directions.* Hoboken, New Jersey, USA: John Wiley & Sons, Inc., All rights reserved, second edition, 2013.

[10] Loo, J.; Mauri, J. L.; Ortiz, J. H. (editors). *Mobile Ad Hoc Networks Current Status and Future Trends.* 6000 Broken Sound Parkway NW, Suite 300 Boca Raton, FL 33487-2742, USA: CRC Press Taylor & Francis Group, 2012.

[11] National Institute of Standards and Technology, Wireless Ad Hoc Sensor Networks. [Online; accessed 5-April-2015]. Available from: `http://www.antd.nist.gov/wahn_ssn.shtml`

[12] The Network Simulator - ns-2. [Online; accessed 22-March-2015]. Available from: `http://nsnam.isi.edu/nsnam/index.php/User_Information`

[13] The Network Simulator ns-3. [Online; accessed 22-March-2015]. Available from: `https://www.nsnam.org/`

[14] QualNet Network Modelling Platform. [Online; accessed 22-March-2015]. Available from: `http://web.scalable-networks.com/content/qualnet/`

[15] OPNET Modeler. [Online; accessed 22-March-2015]. Available from: `http://www.riverbed.com/products/performance-management-control/opnet.html`

[16] OMNeT++ Discrete Event Simulator. [Online; accessed 22-March-2015]. Available from: `http://www.omnetpp.org/`

[17] JiST / SWANS Scalable Wireless Ad hoc Network Simulator. [Online; accessed 22-March-2015]. Available from: `http://jist.ece.cornell.edu/`

[18] ISO/IEC 14882:2014. [Online; accessed 5-April-2015]. Available from: `http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=64029&ICS1=35&ICS2=60`

[19] The GTK+ Project. [Online; accessed 5-March-2015]. Available from: `http://www.gtk.org/`

[20] Cairo Graphics. [Online; accessed 7-March-2015]. Available from: `http://cairographics.org/`

[21] GNUPLOT Version 4.4.0. [Online; accessed 2-March-2015]. Available from: `http://www.gnuplot.info/announce.4.4.0`

Appendix A

Acronyms

GUI	Graphical user interface
WSN	Wireless sensor network
CH	Cluster head

Appendix B

Contents of Enclosed Data Storage

```
text ................ the directory with thesis text and LaTeX source files
  P.Goncharov_Bc.Thesis.pdf .......... the thesis text in PDF format
  src ........................... the directory with LaTeX source files
app ..................... the directory with developed network simulator
  x64 ............................................... 64 bits version
    bin ......... the directory with executable and supplementary files
    doc ......... the directory with technical documentation (doxygen)
    src ........................ the directory with source files (C++)
    doxygen_config_file ............. the doxygen configuration file
    Makefile .......... the Makefile for compilation of the application
    readme.txt ....... the file with brief description of the application
    run.sh ......... the bash script for manipulation of the application
```