Masaryk University

Faculty of Informatics

Reputation Systems in Wireless Sensor Networks

Bachelor Thesis

Petr Soběslavský

Brno 2009

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Advisor: RNDr. Petr Švenda

Acknowledgement

I express my gratitude to Petr Švenda for introducing me into the problematic of reputation systems and for our fruitful discussions.

Abstract

In this thesis we provide an overview of current research on reputation systems in wireless sensor networks. Several proposals are reviewed and compared. In the second part of our research, we developed a universal multipurpose framework for wireless sensor network simulations and implemented selected proposals. We defined a set of tests and evaluated reputation systems with respect to overall network performance, energy consumption and resistance against presence of selfish and malicious nodes. Based on our observations we propose a new reputation system for protecting data integrity in wireless sensor networks. We provide complete simulation results and recommendations for further research.

Keywords

Wireless sensor network, reputation system, routing, security, integrity, simulation.

Contents

1 Introduction

Wireless sensor networks have recently received a great deal of attention from scientific community. Small and cheap devices with low energy consumption and limited computational power can be deployed in large numbers to process great variety of tasks. As the cooperation between nodes is crucial for successful operation of the network, ensuring mutual trust between nodes becomes an important part of securing these networks against both external and internal attackers.

There have been several approaches proposed recently. It could be possible to ensure mutual trust by using some of key exchange algorithms, summarized in (1). However even with a sophisticated key exchange it is still possible that a node change its behavior during network operation. Therefore there should be a possibility for the network to dynamically react to presence of misbehaving nodes. One of approaches to handle this problem is using reputation system.

Reputation system is a system which enables network to dynamically evaluate *reputation value* of a node. Reputation value of a node is a level of trust other nodes have in the node. This value is computed using various inputs – past experiences with a node, observation of communication or opinion of other nodes.

In recent years several papers on reputation systems have been published. Each of them provides some proof of feasibility and efficiency. However we were not able to find any complex comparison of these proposals in similar conditions. The aim of our research was to create a multipurpose framework for reputation system simulation, implement selected reputation systems and compare their capabilities using a set of tests.

The thesis is organized as follows: First chapter introduces wireless sensor networks, security challenges and principles of reputation systems. Second chapter gives an overview of systems selected for our comparison. Third chapter describes multipurpose WSN simulation framework SensNet, which was developed as part of this research. Fourth chapter explains our simulation of reputation systems and analyzes results. In the fifth chapter we show how a reputation system can be used to protect network against malicious nodes. Sixth chapter concludes the paper. All simulation results are attached in Appendix A. Appendix B describes contents of the attached CD.

1.1 Wireless Sensor Networks

Wireless sensor network is a network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions. The deployment is either random or utilizes predefined locations. Typically a number of sensor nodes are scattered in the area, collect data and route them through a multi-hop structure to a specialized node referred to as a *sink* or *base station*. The base station then communicates the raw or processed data to the user via a traditional infrastructure network. This is illustrated in Figure 1.

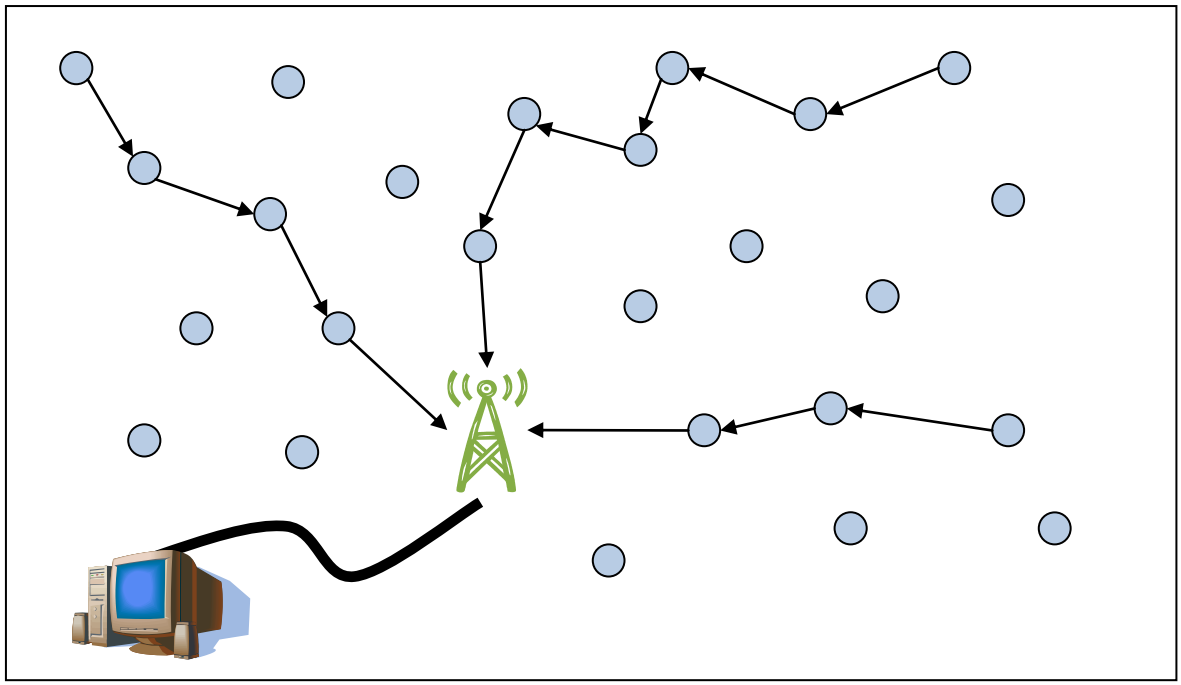

Figure 1 Illustration of a wireless sensor network

Every node in a sensor network is typically equipped with a radio transceiver, a small microcontroller and a battery. All memory, energy and computational resources of sensor nodes are usually very limited, which has to be kept on mind while designing protocols and applications.

Sensor nodes may change their location after initial deployment. Mobility can result from environmental influences such as wind or water, sensor nodes may be attached to or carried by mobile entities or sensor nodes may possess automotive capabilities.

Depending on the actual needs of the application, the form factor of a single sensor node may vary from the size of a shoe box (e.g., a weather station) to a microscopically small particle (e.g., for military applications where sensor nodes should be almost invisible). Similarly, the cost of a single device may vary from hundreds of Euros (for networks of very few, but powerful nodes) to a few cents (for large-scale networks made up of very simple nodes). (2)

1.2 Dynamic Source Routing Protocol

Sensor network is usually a wireless ad-hoc network, which means that each node supports a multi-hop routing algorithm. All proposals considered in this paper work on top of the Dynamic Source Routing algorithm (3). This algorithm is well suited for reputation systems a node sending packet has full control over the whole route and can easily avoid untrusted.

Dynamic Source Routing protocol is an on-demand protocol designed to restrict the bandwidth consumed by control packets by eliminating periodic table-update messages required in table-driven approach.

Operation of the protocol is depicted on Figure 2: Consider a source node S which wants to send a packet to a node D. If it has no routing information stored in its cache, it starts *route discovery*. It broadcasts **Route Request** to all nodes in its neighborhood. When a node receives Route Request packet, it first checks its *query number* field and if this is the first time it encounters this request, it adds its address at the end of the *route* field and forwards the requests to its neighbors. This way the request is flooded throughout the network. When the request arrives to the destination node D, it responds with **Route Reply** packet which is sent back to the source reversing the request route.

The source node and all nodes on the way store received route information in their cache. After some time a node has several routes to possible destinations and while sending own packets it uses some metric to choose the best route. In the standard DSR implementation a node always choose the shortest path (i.e. the path containing the lowest number of nodes). As we will see, reputation systems may use different metrics to avoid routing via untrusted nodes.

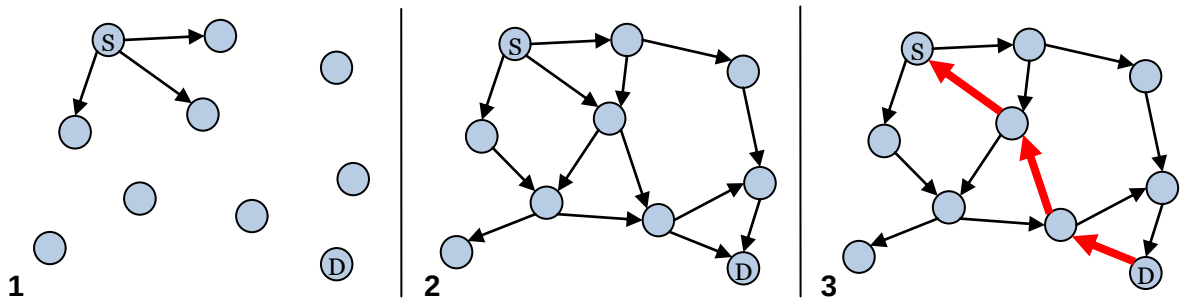

Figure 2 DSR Routing: 1. Sending Route Request, 2. Request propagation, 3. Sending Route Reply

1.3 Security in Wireless Sensor Networks

There are many security issues common with usual wireless ad-hoc networks. In order to evaluate security of a network, we must define suitable threat model. The threat model formulates the hypothesis regarding the attacker's capabilities and its possible behavior. A common assumption is the Dolev-Yao model (4). In this model, the attacker can gain control over the communication channel and hear the messages between the parties, intercept them, prevent their delivery and also forge its own messages into the system.

However, as sensor network may operate in an unattended environment, the possibility of attacker gaining control over some of the communicating nodes and acquiring all the information stored within it, has to be considered. As was already mentioned, this is also why securing sensor network cannot rely only on secure key establishment, but also has to take into account possibility of node compromising during network operation.

Attacks on wireless sensor network can be categorized as active or passive:

- **Passive attacks** – The eavesdropped can continuously monitor network communication and use traffic analysis to discover communication patterns.
- **Active attacks** – The attacker can capture a sensor node, steal all information stored in it and use its identity and reputation for further operation in the network.

Attackers can be classified into two categories depending on their abilities:

- **Mote-class attacker** – The attacker has access to few ordinary sensor nodes with lesser capability and might only be able to jam the radio link in its immediate vicinity.
- **Laptop class attacker** – The attacker has access to more powerful devices with greater battery power, more capable processor and a high-power transmitter with a sensitive antenna.

It is generally assumed that the environments in which the sensors are deployed are risky and untrusted. Each sensor trust itself, but sensors do not trust each other. Further it is assumed that all compromised sensors are compromised by the same attacker and that there is no upper bound on the number of compromised nodes. However, the attacker cannot compromise the base station, which is typically resourceful and well protected. (5)

There is one other risk for network functionality, which is specific for wireless sensor networks. As energy resources are very limited, nodes might have tendency to save as much energy for their longer operation as possible. While such behavior is beneficial for a node itself it might be a risk for overall network functionality. Such node, called **selfish node** can

refuse to forward packets for other nodes or give false replies to routing protocol requests in order to avoid becoming part of packet routes.

Apart from selfish node, there might also be a **malicious node**. Such node can intentionally deteriorate network functionality by flooding it with false information, generate excessive traffic in order to consume energy of other nodes, impersonate another node or intentionally change data stored in packets it is forwarding. It is also possible that several malicious nodes coordinate their efforts using secret communication channel and carry on so-called *collusion attack*.

All over this research we make a clear distinction between selfish nodes, which are trying to save energy for them and malicious nodes, which are trying to directly attack network in order to significantly decrease its functionality or to gain access to sensitive data. As the terminology is not unified yet, it is possible that other papers use different words. Widely used expression *misbehaving node* can refer to both selfish and malicious node.

1.4 Reputation System

Reputation system is a decentralized distributed system of reputation evaluation and information exchange between nodes in wireless sensor network. Every node contains a module which calculates reputation values of other nodes.

Possible inputs taken into consideration are:

- **Own experience** – Node's own experience with a neighbor. For instance, how many route requests were correctly answered by the neighbor, how many packets the neighbor forwarded, how many packets were received from the neighbor
- **Observations** – Node can switch its transceiver into promiscuous mode and listen to outgoing communication of its neighbor in communication range. This way it can check, whether a neighbor reacts correctly to requests, for instance, whether it really forwards packets it is requested to forward.
- **Neighbor notifications** – Reputation system can use information shared between nodes. Node can exchange reputation information with its direct neighbors to get access to observations it could not make by itself.
- **Further information propagation** – Reputation information can be propagated further throughout the network via multi-hop routes.
- **Information from nodes along the route** – While trying to find a route to a destination in DSR algorithm, a node creates route request, which is then propagated through the network. When destination is found, the route information is returned via the

new route back to source route. Nodes along this route can observe behavior of involved nodes and report the sender if these nodes are behaving correctly or not.

- **Information from route requests** – Route requests can be used for cheap and easy propagation of reputation information. The observations and reputation values can be included in a route request and all nodes receiving this request can incorporate them into their own evaluation.

As we will see, different reputation systems use different subset of mentioned inputs and different formulas for reputation evaluation.

Common assumption is that a good reputation system should cause selfish behavior disadvantageous and motivate nodes to behave correctly. It should be possible to isolate malicious node from the network, but in such a way that isolated nodes still fulfill their duties against other nodes. Some sort of re-socialization and re-integration for nodes that change their behavior should also be possible. (6)

2 Selected Reputation Systems

We have selected five several proposals for our comparison. All proposals work on top of Dynamic Source Routing algorithm and work in similar way: Each node in the network observes all or some of its neighbors, evaluate their reputation value and spread reputation information to other nodes. The systems differ in the way this information is spread and in the formula used to evaluate reputation value. The aim of a system for the node is to be able to find the best possible route for sending own packets and eventually to prevent misbehaving nodes from using the network.

Some of the proposals also consider further security measures, i.e. using encryption or hash chains to protect reputation system messages. We agree that ensuring integrity and confidentiality in the sensor network by means of cryptography is a critical issue, however we believe that it should be dealt with separately in a wider context of the system and that the reputation information should be protected in the same way as the data.

There is one frequently cited proposal called CONFIDANT ((6) and (7)), where the evaluation of received alarm messages is based on trust levels established by PGP encryption system. The problem of this approach is that even a node highly trusted in terms of PGP can be selfish in terms of routing. As we think that PGP trust and routing trust are two different thinks and we wanted to focus on the essential principles of reputation systems, we decided not to consider CONFIDANT in our comparison.

In our research we wanted to find out which reputation system is the most effective, which means to answer the following questions:

1. What reputation data should be shared between nodes to remain sufficiently effective?
2. How should be the reputation value evaluated?
3. How to prevent nodes from misbehaving?

This section overviews selected proposals with respect to these questions and provides a short theoretical comparison.

2.1 Systems Overview

2.1.1 Watchdog and Pathrater

First published in (8) Watchdog was the first proposal and inspiration for further solutions. It introduces the *watchdog* mechanism and proposes route evaluation in DSR be based on reputation values of other nodes.

As sensor nodes use wireless all-direction antennas, it is possible for a node to observe behavior of its neighbors and check whether they follow the protocol correctly. When node A is forwarding a data packet, it knows addresses of next two hops in the route – nodes B and C. Because B is in communication range of A, message forwarded from B to C can also be overheard by node A as shown in Figure 3.

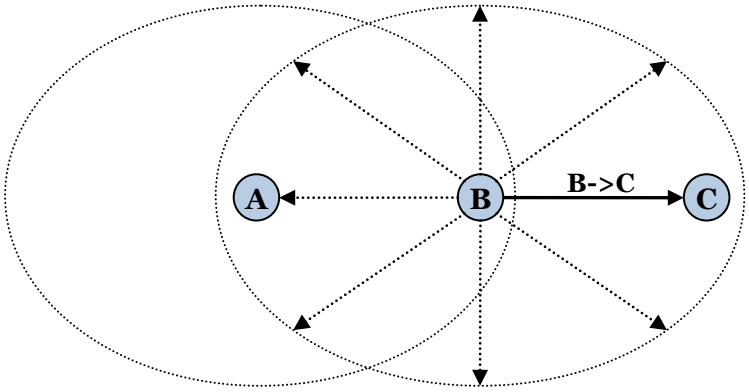

Figure 3 Node A overhearing message sent from B to C

Watchdog works as follows: it maintains a buffer of recently sent packets and compares each overheard packet with the packet in the buffer to see if there is a match. If so, the packet in the buffer is removed, since it has been forwarded on. If a packet has remained in the buffer for longer than a certain timeout, the watchdog increments a failure tally for the node responsible for forwarding on the packet. If the packet exceeds a certain threshold bandwidth, it determines that the node is misbehaving and sends a message to the source notifying of the misbehaving node. (8)

According to the specification, when no encryption is used the payload of the packet can be checked on forwarding. We therefore expect this system to be resilient not only against selfish nodes but also against malicious nodes. Reputation value of nodes is incremented in time, allowing resocialization of nodes that change their behavior.

Drawback of this approach is that it does not punish nodes that do not cooperate, but rather relieves them of the burden of forwarding for others, whereas their messages are forwarded

without complaint. This way, the misbehaving nodes are rewarded and reinforced in their behavior. Another problem is that a node trusts only accusation from one node and it is easy to subvert such accusation and innocent node might be wrongly penalized.

2.1.2 Context-Aware Detection

This proposal was first published in (9) as a reaction to the Watchdog mechanism. According to authors, there is no need to attack data forwarding phase when an attacker can more efficiently attack the route discovery phase. Their proposal therefore focuses on route discovery phase.

Each node observes behavior of all its neighbors. When it detects some misbehavior (i.e. not forwarding the route discovery packet), it sends alarm message to the source of the request. The source node executes an inference scheme based on majority voting to rate an accused. Source node can later on advertise this rating and it can be used by knowledgeable nodes to deny any future service to the attackers.

Every time after sending route requests, a node waits some amount of time and collects alarm messages. These messages are always evaluated in the context of the actual route discovery. To convict a culprit, more than three accusations are needed. If there is only one accusing node, it is itself considered to be an attacker.

The drawback of this approach is that it is more beneficial for a node not to send the alarm messages, as while sending an alarm message it risks that it will be the only accuser and it will be regarded as an attacker.

Another problem is that a selfish node can make use of this scheme to avoid forwarding foreign packets. If a node is considered to be an attacker, no one wants to route its packets via it and it therefore saves energy while still being able to send its own packets.

2.1.3 CORE

The idea behind CORE (10) is that only positive rating factors are distributed among the entities while the negative ranking factors are evaluated locally. While sending packet to a node, the sender observes behavior of its neighbor. When misbehavior is detected, the reputation value is decreased. On the other hand positive observations received from other nodes cause the reputation value to rise. There are no special packets used to spread positive observation. Receiving a packet which has before been forwarded by a node A is a positive observation of node A.

While evaluating report messages and observations, CORE considers past observations more relevant than recent ones. This way a short-time problem on communication link does not affect node's reputation. Reputation value is also decremented along time, so if a node is idle and does not want to cooperate, its reputation lowers.

In CORE the reputation is evaluated with respect to different functions (e.g. route discovery, packet forwarding). Overall reputation of a node is calculated as weighted mean of these *functional reputations*. According to the authors more relevance should be given to the packet forwarding function.

CORE punishes misbehaving nodes by not forwarding their packets. It does not use reputation information for path evaluation, so packets are still routed via misbehaving links. This way the selfish nodes do not save energy and selfish behavior does not pay off. However there is higher risk for sender that a packet will not reach its destination when a selfish node refuses to forward it.

2.1.4 SORI

The SORI proposal (11) combines node observation, information sharing between direct neighbors and probabilistic dropping of packets originating from nodes considered malicious.

The reputation value of a neighbor is calculated as a number of packets the neighbor was asked to forward divided by number of packets it really forwarded.

When the reputation value changes significantly, the new value is broadcasted to all neighbors. To calculate overall reputation value, own observation and received notifications are combined weighted by the number of requested packets and by the reporter reputation.

If a node is requested to forward a packet originating from a node with low reputation, this packed is probabilistically dropped.

Similarly to CORE, the reputation value is not used for evaluation of routing paths, so selfish nodes are still being requested to forward packets for other nodes, while not being able to send their own packets. Again this approach does not ensure better overall performance of a network with selfish nodes, but rather serves as a motivation for correct behavior.

2.1.5 OCEAN

The aim of OCEAN (12) is to show that it is possible to ensure performance similar to previous proposals without using any sophisticated and potentially vulnerable techniques of reputation propagation throughout the network.

The reputation of a neighbor is evaluated using only locally available information. For every neighbor a node counts how many times it forwarded a packet it was requested to forward (which increases reputation value) and how many times it failed to forward a packet (which decreases reputation value).

The reputation value is then used in DSR route discovery avoid lists: When initiating a route request, a node puts a list of nodes it considers misbehaving at the end of the Route Request message. Every node receiving the request first checks whether it was received from a node contained in the Route Request avoid list. If it was, the packet is dropped. If it was not, the node adds its own list of untrusted nodes and forwards the packet further on.

As this system prevents selfish nodes from both sending and forwarding packets, it is actually a low-cost way of isolation of misbehaving nodes from the network.

2.2 Comparison

In this chapter we presented five proposals which use different inputs and reputation evaluation function. We have seen that some proposals may cause higher communication overhead, while some do not use any special packets at all. We summarized our first findings about the systems in Table 1.

While all being called reputation systems, the purpose of each system is slightly different. There are two systems designed primary to ensure high network throughput even in presence of misbehaving nodes (Watchdog and Pathrater, Context Aware Detection). Two systems aim to enforce cooperation by punishing misbehaving nodes and motivating them to act correctly (CORE and SORI). The purpose of the last system (OCEAN) is to completely isolate misbehaving nodes from the network.

All systems evaluate reputation based on observation of packet forwarding and/or route discovery responses. Only one of the proposals (Watchdog and Pathrater) considers malicious nodes. In our comparison we will therefore focus more on routing functions.

There are two main ways of calculating neighbor reputation from direct observations. Watchdog and OCEAN use an approach in which a reputation value is incremented with every positive and decremented with every negative observation. In CORE and SORI the reputation value is calculated as a ratio of number of successfully forwarded packets to the number of all packets the neighbor was requested to forward.

Most of the systems give an opportunity for reintegration of a node in case it changes its behavior. It is usually ensured by some timeout period after which a node is given second

chance or by evaluation formula which considers recent observation and makes it possible for a node to gain again enough positive observations.

	Watchdog and Pathrater	Context Aware Detection	CORE	SORI	OCEAN
Considered attackers					
Selfish nodes	●	●	●	●	●
Malicious nodes	●				
Adaptability					
Possibility of re-integration	●		●	●	●
Uses					
Own experience	●		●	●	●
Observations					
Neighbor notifications			●	●	
Information from nodes along the route	●	●			
Information from route requests					●
Information spreading					
Negative observations	●	●		●	
Positive observations			●	●	
Avoid lists					●
Reaction to a misbehaving node					
Avoid routing through a misbehaving node	●	●			●
Refuse to route / drop packets from a node			●	●	●
Refuse to route / drop packets for a node					
Consequences for a misbehaving node					
Forwards packets			●	●	
Sends packets	●	●			
Receives packets	●	●	●	●	●
Saves energy	●	●			●

Table 1 Comparison of reputation systems

Several methods of reputation information sharing have been mentioned. Watchdog and Pathrater and Context Aware Detection require an observer of node misbehavior to send information to the source. While this way reputation information can be propagated to distant parts of the network, there is a risk of excessive communication overhead (especially in case of Context Aware Detection).

CORE and SORI use observations from direct neighbors, which enables a node to have a good overview of its neighborhood. The disadvantage is that a node can base its routing decisions only on this knowledge and is not able to evaluate reliability of more distant nodes.

In OCEAN avoid lists are enlarged by each hop in the route. This way each node practically decides the following node on the route, while not having any information about possible

successive hops. This solution does not make use of the features of DSR protocol and could be possibly used with another on-demand protocol (e.g. AODV).

We will see in the fourth chapter how these differences affect network performance in different situations.

3 SensNet Simulation Framework

3.1 Base Platform

At the beginning of our research we evaluated several existing simulators and frameworks. We were not able to find any available framework covering all our requirements. Finally we decided to develop our own framework for wireless sensor network simulations. We chose OMNeT++ as a base platform and its Mobility framework as a starting point for our implementation.

3.1.1 OMNeT++

OMNeT++ is a public-source, component-based, modular and open-architecture simulation environment available online at (13).

In general OMNeT++ is a discrete event simulator which can be used for wide variety of purposes like modeling of traffic of telecommunication networks, protocols, queueing networks, multiprocessors and distributed hardware systems, validating hardware architectures, evaluating performance of complex software systems and so on.

An OMNeT++ model consists of hierarchically nested modules. Modules communicate through message passing. They can send messages either directly to their destination or along a predefined path, through gates and connections. Modules can have their own parameters, which can be used to customize module behavior and to parameterize the model's topology.

OMNeT++ simulation can make use of several user interfaces for different purposes. In our simulation we use Tcl/Tk graphical user interface for debugging and presentation and faster command-line interface for simulations of large networks.

The simulator and user interfaces and tools are portable – they work on Windows, Mac OS and several Unix-like systems, using various C++ compilers.

3.1.2 Mobility Framework

Mobility framework (14) is intended to support wireless and mobile simulations within OMNeT++. The core framework implements support for node mobility, dynamic connection management and a wireless channel model. Additionally the core framework provides basic modules that can be derived in order to implement own modules.

Mobility Framework is a good basis for further work, but unfortunately currently implemented set of protocols and functionalities is quite limited and instead of building on top of this framework we had to reuse parts of its code to develop our own framework aside from existing ones.

3.1.3 SensNet Framework

We extended functionality of Mobility Framework in these areas:

- **Energy consumption** – It is now possible to set battery capacity of each node and energy consumption of network interface card in different modes of operation (send/receive/sleep). When there is no energy in battery, node cannot send and receive packets anymore.
- **Power control** – It is now possible to set for each packet at which power level it will be sent.
- **Reputation system** – Reputation system module in each node keeps and evaluates information about other nodes in the network. It can be used by application layer, network layer or routing module.
- **Behavior control** – Every node has its own behavior control module, which makes controlling behavior of a node much easier. Having behavior control as a separate module makes complex behavior decisions possible.
- **Hidden communication** – Every module has its hidden input/output simulating hidden communication channel between nodes. It is now possible to simulate collusion attacks incorporating several nodes.
- **Simulation control** – We added simulation manager into our network. It can load an XML file with complex scenario settings and send simulation control messages to nodes via their simulation control input.
- **Simulation output** – We added new output mechanism which can be used for writing values into CSV[1] files. These files can be further processed by any text editor or imported into a spreadsheet application.

We used main ideas and parts of the source code of Mobility Framework. Our framework uses the same channel control, 802.11 network interface card and mobility implementation. We preserved Blackboard in each node for backwards compatibility. We added several modules to nodes, global scenario manager and made minor modifications in NIC code to enable simulation of energy consumption.

[1] Comma-separated values

We also simplified address model of Mobility framework by removing separate application address. Thus every node in SensNet framework has two addresses:

- **Network layer address** – Is used by application and network layer and for routing. In our implementation it has value of network layer module id.
- **MAC address** – Can be obtained for given address via ARP protocol.

We implemented set of modules that can be directly used or inherited in simulations. As a general rule modules named *Basic** provide basic functionality and are intended to be base classes for custom modules. These modules should not be used directly. For some modules we also provide *Simple** variants that implement some functionality and can be used in simulations.

3.2 Network

In the simulation all nodes are located in a fixed-size area. Each node is represented as a compound module described in the next section. Aside from regular nodes there is also a special node called *sink*, which is collecting data from hosts.

Communication between nodes is established by **Channel Control**. This module evaluates node positions and dynamically creates wireless channels between them. When a host receives a packet, its transport parameters (receive power, S/R ratio, bit errors) are evaluated and the packet is either accepted or discarded.

We extended the simulation model by introducing **Scenario Manager**. This module has direct connection with all hosts and can control their behavior during the simulation. Scenarios are given as an XML file defining events to happen in a given time. We use this functionality for scheduling output times, behavior changes and to quit the simulation after given time.

Output Control is used to output lines to CSV files. All basic data types as strings, integers and floats can be exported for further analysis in a separate application.

For evaluation of communication we use **Packet Counter** module. Every packet being sent or received is notified to this module. When requested by Scenario Manager at the end of the simulation the statistics are written to output files. Detailed results for every node as well as aggregate results for the whole network are outputted.

3.3 Node

A typical node is a compound module consisting of several modules. There are three possible means of communication with a node:

- **Network Interface Card (NIC)** – Enables wireless communication with other nodes in its wireless range.
- **Simulation Control Input** – Connected to Simulation Manager, it receives simulation control messages and forwards them to Mobility module, Traffic Generator or Behavior Control
- **Hidden Input/Output** – Can be used for hidden communication with other nodes or with a separate module managing collusion attacks.

Schema of a typical node is shown on Figure 4.

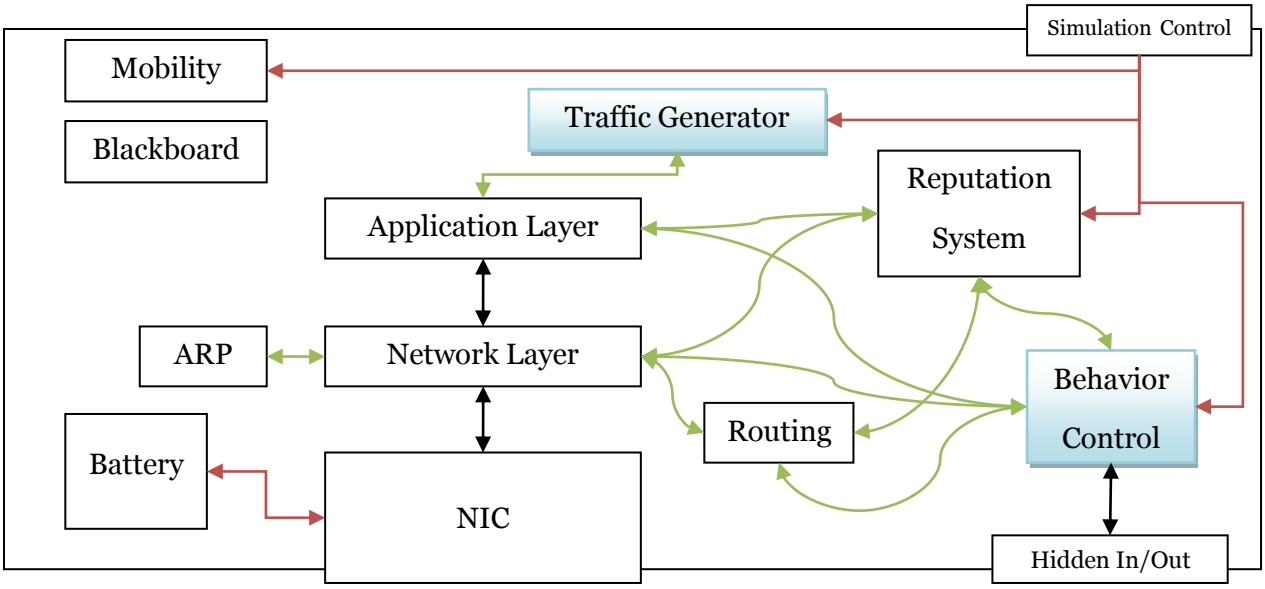

Figure 4 Schema of a typical node in SensNet framework

3.3.1 Mobility

Mobility module sets the node position in the area.

For some simulation scenarios we use Random Waypoint model mobility for simulating moving nodes. In this mobility model, a node randomly chooses new destination and speed and starts moving. When reaching the target position, it waits for a random time and then chooses a new target position. (15)

3.3.2 Blackboard

The Blackboard module was preserved for compatibility with some modules from Mobility Framework. It is used for information sharing between modules in the node.

3.3.3 Traffic Generator

This module can be used for generating simulation traffic. According to its settings it periodically sends traffic generation requests to the application layer.

3.3.4 Application Layer

In some simulations application functionally might be quite complex, however in our simulation the application layer only accepts requests from the Traffic Generator and sends application packets to other nodes.

3.3.5 Network Layer

This module takes care of network communication. It receives packets from application layer and sends them down to the Network Interface Card. Using Routing module it sets routing paths of the packets. It also forwards packets from the NIC to the application layer.

We created two implementations: ***BasicNetworkLayer*** which only sends packets up or down and ***RoutingNetworkLayer*** which upon receiving a packet from application layer requests Routing module for a route to the packet destination.

3.3.6 ARP

Address Resolution Protocol translates network addresses to MAC addresses. Our simple implementation ***SimpleArp*** directly converts network address (which is in fact id of the Network Layer module of a host) to the MAC address (id of the NIC module of a host).

3.3.7 Routing

Routing module accepts RouteQuery messages from Network layer. In order to find a route to the destination, it first checks its cache. If the route is not yet known, it starts route discovery and responds with RouteQueryResponse to the network layer.

We implemented ***SimpleRouting*** which is only able to route messages to the node's direct neighbors and ***DSRRouting***, which implements DSR routing algorithm (16) extended with reputation system support.

3.3.8 NIC

Network Interface Card implementation was taken from the Mobility Framework. In our framework we retained all physical and MAC layer types, however only implementation of 802.11 was modified to work properly with our new node scheme.

It consists of four submodules:

3.3.8.1 MAC

Medium Access Control was modified so that upon each arrival of a message from network and upon each send request it checks available battery capacity. If the battery is empty, message is removed.

3.3.8.2 Decider

Based on signal-to-noise ratio and other settings the Decider decides whether a packet was received correctly or not.

3.3.8.3 snrEval

This module evaluates signal-to-noise ratio of received packets.

3.3.8.4 Radio

Radio communicates with other radios in a communication distance via Channel Control module. We simulate a simple radio with one channel and all-directional antenna.

3.3.9 Behavior Control

Other modules ask behavior control on how to behave in different situations. For instance, routing module asks if it should send a correct or false response to some request. Different kinds of node misbehavior can be implemented here.

In our simulation we use the behavior module to emulate selfish and malicious nodes.

3.3.10 Reputation System

Other modules ask reputation systems questions about actions connected with reputation. In our simulation it communicates frequently with routing module. Reputation System module can be asked to evaluate possible DSR routing path or to decide whether to react to a route request or not. All incoming and outgoing packets are notified to this module. It can also add its own information to the DSR Route Request packets.

We implemented all considered systems as Reputation System modules.

3.3.11 Battery

Battery module is given total energy capacity and energy consumption in different modes of radio operation as parameters. Every time radio is in operation, it consumes energy in battery module. When the battery runs out of energy, NIC stops to operate and discards all incoming packets.

4 Simulation of Reputation Systems

In order to make a complex objective comparison of reputation systems we implemented all systems according to descriptions in the papers. We focused on efficiency of routing and packet forwarding leaving alone additional security features like encryption or digital signatures.

Source code of the simulation is available on the attached CD as the RepSysSim project. We created a multiplatform application based on our SensNet framework. For starting simulation runs we prepared a Python script, which can be found in the Final directory.

4.1 Test Scenarios

We defined a set of 31 test scenarios and simulated all considered systems in every test scenario. We ran each test ten times and averaged the results. Simulation results were obtained during few hours on a powerful sixteen-processor server.

In order to make the simulation as similar to real deployment as possible we used a communication pattern in which every node in the network periodically sends data to the base station. Simulation time for most of the scenarios was 1000 seconds. Simulations of limited energy resources were left running for much longer time so that all nodes ran out of energy before the end and we could evaluate how energy efficient the network was. We counted total number of packets sent and delivered as well as number of packets originating from selfish and non-selfish nodes.

Common scenario settings are presented in Table 2, while specific settings for each scenario are given in Table 3. Hardware settings correspond to the referential MICA2 sensor node. (15)

Area settings		Application settings	
Dimension (m)	200 x 200	MAC	802.11
Number of nodes	100	Sending capacity (kbps)	38,4
Avg. number of neighbors	4.3	Application	CBR
Initial placement	Uniform	Application packet size (B)	24
Hardware settings			
Frequency (Mhz)	915	Receive power cons. (mW)	30
Min. transmit power (mW)	0.01	Min. transmit power cons. (mW)	30
Max transmit power (mW)	3.162	Max. transmit power cons. (mW)	81
Sensitivity (dBm)	-110		

Table 2 Common scenario settings

Performance with various numbers of selfish nodes			
Measures how the reputation system reacts to large number of selfish nodes present in the network. Good reputation system should be resilient against large portion of selfish nodes.		Mobility model	None
		Speed (m/s)	0
		Pause time (s)	0
		Sending interval (s)	100
		Sending start jitter (s)	100

Code	Number of selfish nodes	Dropping probability of selfish nodes
NSN1	0	0.5
NSN2	10	0.5
NSN3	20	0.5
NSN4	30	0.5
NSN5	40	0.5
NSN6	50	0.5

Performance under various number of connections			
Good reputation system should not generate much of excessive traffic. Overall application performance (i. e. number of application packets the network is able to deliver) should not be kept as high as possible.		Mobility model	None
		Speed (m/s)	0
		Pause time (s)	0
		Number of selfish nodes	20
		Dropping prob. of self. nodes	0.5

Code	Sending interval	Start jitter
NC1	200	200
NC2	100	100
NC3	50	50
NC4	25	25
NC5	12.5	12.5
NC6	6	6

Performance under various speed of movement			
In some applications it may be important to react fast on changes in network topology caused by node movement. Good reputation should be flexible in evaluating reputation of new neighbors.		Number of selfish nodes	20
		Dropping prob. of self. nodes	0.5
		Sending interval (s)	100
		Sending start jitter (s)	100
		Mobility model	Random waypoint

Code	Speed (m/s)	Pause time (s)
SM1	0	0
SM2	0.1	100
SM3	0.2	50
SM4	0.4	25
SM5	0.8	12
SM6	1.6	6

Performance under various dropping probability of selfish nodes			
Measures how the reputation system reacts to different behavior of selfish nodes.		Mobility model	None
		Speed (m/s)	0
		Pause time (s)	0
		Sending interval (s)	100
		Sending start jitter (s)	100
Code	Number of selfish nodes	Dropping probability of selfish nodes	
DP1	20	0.2	
DP2	20	0.4	
DP3	20	0.6	
DP4	20	0.8	
DP5	20	1.0	
Performance with nodes changing their behavior			
It is possible that a node change its behavior during the network operation. Good reputation system should be flexible and able to raise reputation of a well behaving node and lower reputation of node that changed to be selfish.		Mobility model	None
		Speed (m/s)	0
		Pause time (s)	0
		Sending interval (s)	100
		Sending start jitter (s)	100
Code	Number of selfish nodes	Dropping probability of selfish nodes	Time between change
NCB	40	0.5	100
Performance with limited energy resources			
Energy resources have great importance in wireless sensor networks. Good reputation system should protect nodes against energy wasting by sending useless packets. Selfish nodes should not have advantage over well behaving nodes		Mobility model	None
		Speed (m/s)	0
		Pause time (s)	0
		Sending interval (s)	100
		Sending start jitter (s)	100
		Battery capacity (J)	18000
Code	Number of selfish nodes	Dropping probability of selfish nodes	
LER1	20	0.5	
LER2	60	0.5	
Performance with malicious nodes			
In this scenario we introduce a portion of nodes that change content of each packet they are forwarding.		Mobility model	None
		Speed (m/s)	0
		Pause time (s)	0
		Sending interval (s)	100
		Sending start jitter (s)	100
Code	Number of malicious nodes	Probability of packet change	
MN1	10	1.0	
MN2	30	1.0	
MN3	50	1.0	
MN4	70	1.0	
MN5	90	1.0	

Table 3 Specific scenario settings

4.2 Results

As we have already seen in the second chapter the aims of the systems are slightly different, which should be taken into account in result evaluation.

4.2.1 Number of Selfish Nodes

In the first set of tests we focused on performance under various numbers of selfish nodes with the same probability of packet dropping.

	None	Watchdog	Context Aware	CORE	SORI	OCEAN
NSN1	910,9	911,5	908,0	911,2	911,2	911,2
NSN2	689,8	707,7	673,1	692,7	692,7	694,7
NSN3	443,8	454,7	437,2	460,4	460,4	461,0
NSN4	291,9	274,2	265,8	279,6	279,6	280,6
NSN5	191,0	191,1	179,8	197,2	197,2	196,9
NSN6	123,2	118,5	119,3	119,3	119,3	119,3

Table 4 Total number of packets delivered to the base station with various numbers of selfish nodes

As we can see in Table 4 when there are no selfish nodes in the network (NSN1) about 91 % of packets is delivered. The packet loss is due to simulated errors in wireless transfer. We did not implement any mechanism of end-to-end control of packet delivery and resending as we believe that in the moment of eventual packet resending the information is outdated and it is more efficient to wait and send a new and actual value in next turn. This way also energy resources are saved.

When the number of selfish nodes gets higher, the reputation systems start to be able to identify misbehaving nodes and to find alternative routes around them. This way Watchdog and Pathrater can increase network efficiency by 2.5 % in the NSN3 scenario. In the same scenario CORE and SORI, i.e. the systems that let selfish nodes forward packets instead of isolating them, prove to be more efficient than Watchdog and Pathrater, increasing network efficiency by 4 %. In the same line OCEAN also seems to be better able to avoid routing through selfish nodes and also causes 4 % increase in number of delivered packets.

When the number of selfish nodes is too high (50 % selfish nodes in NSN6), it becomes impossible to find routes which do not contain selfish nodes and even reputation systems cannot solve the problem.

4.2.2 Dropping Probability

In the second set of tests we measured how the number of delivered packets depends on probability of packet dropping by selfish nodes.

	None	Watchdog	Context Aware	CORE	SORI	OCEAN
DP1	786,6	799,1	784,7	790,1	790,1	790,1
DP2	711,9	684,0	696,0	672,9	672,9	672,5
DP3	672,7	651,4	652,3	650,0	650,0	650,8
DP4	682,6	660,4	657,7	644,0	644,0	646,7
DP5	739,5	740,6	737,9	740,4	740,4	740,4

Table 5 Total number of packets delivered to the base station with various dropping probability of selfish nodes

As we can see in Table 5 possibilities for reputation systems are quite limited in extreme cases. When dropping probability is low (20 % in DP1), it is difficult for a system to correctly distinguish between node misbehavior and transport error. When dropping probability is high (1.0 in DP5), selfish nodes drop all routing packets and thus do not take part in communication. Therefore results are again very similar.

Interestingly in this test performance of networks protected by reputation systems is worse than performance of unprotected network. This might be caused by the fact that reputation systems are not able to react to changing behavior of the nodes, i.e. when a node forwards one packet correctly they increase its reputation too much and when it behaves incorrectly only in a portion of cases, the reputation is not sufficiently decreased.

4.2.3 Number of Connections and Speed of Movement

Increasing number of connections and speed of movement are a challenge for routing protocol design. Good routing protocol should ensure high delivery efficiency in high network traffic and should be flexible enough to work in a network with changing topology. In these two sets of tests we measured how introducing reputation systems affect efficiency of DSR routing.

	None	Watchdog	Context Aware	CORE	SORI	OCEAN
NC1	329,5	333,4	326,0	342,1	342,1	339,3
NC2	689,8	707,7	673,1	692,7	692,7	694,7
NC3	1 458,6	1 424,8	1 340,6	1 444,4	1 444,4	1 433,5
NC4	2 789,6	2 724,1	2 657,1	2 646,0	2 646,0	2 609,3
NC5	5 534,1	5 457,1	5 353,5	5 604,8	5 604,8	5 543,4
NC6	11 666,2	11 715,9	11 263,3	11 537,0	11 537,0	11 461,9
SM1	689,8	707,7	673,1	692,7	692,7	694,7
SM2	687,8	643,2	666,4	664,5	664,5	651,2
SM3	573,1	601,7	583,6	573,0	573,0	571,7
SM4	476,8	496,8	511,2	515,3	515,3	499,5
SM5	504,5	501,1	508,7	495,9	495,9	495,9
SM6	498,3	496,5	502,0	508,7	508,7	508,7

Table 6 Total number of packets delivered to the base station with increasing number of connections and speed of movement

Table 6 shows that all systems are able to handle increasing number of connections quite well. There is only a little decrease in number of delivered packets in Context Aware Detection protected network, however it is in maximum just 4 %.

4.2.4 Limited Energy Resources

One of the most important questions we asked at the beginning of our research was to find out how reputation systems affect energy consumption of the network. We let every network in operation until all nodes ran out of energy and counted packets successfully delivered to the base station.

	None	Watchdog	Context Aware	CORE	SORI	OCEAN
LER1	413,6	420,2	400,1	409,5	409,5	408,7
LER2	173,9	166,6	160,5	170,1	170,1	170,0

Table 7 Total number of packets delivered to the base station with limited energy resources

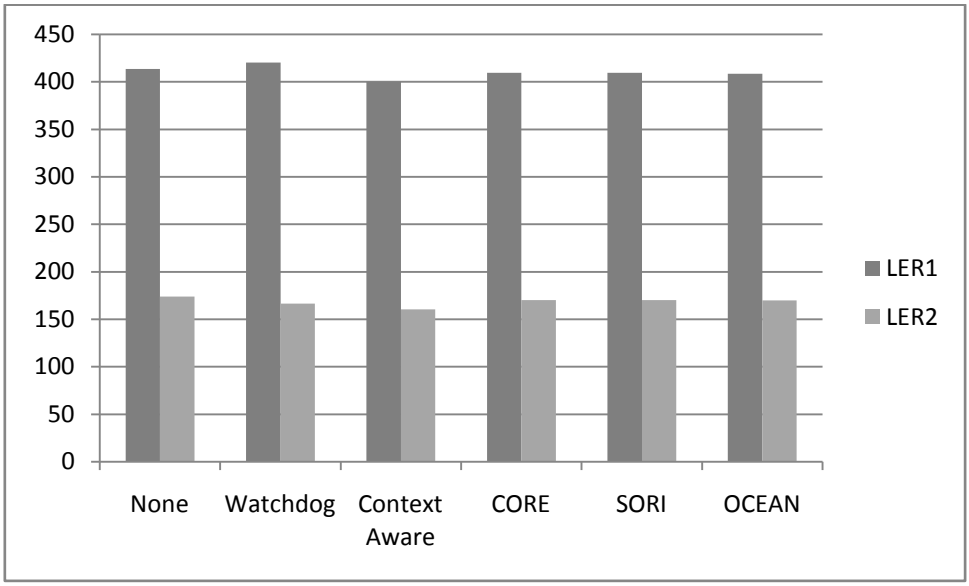

Figure 5 Total number of packets delivered to the base station with limited energy resources

Table 7 and Figure 5 show that communication overhead caused by reputation systems does not significantly affect energy consumption of the network. Using Watchdog and Pathrater in a network with relatively well behaving nodes (only 20 selfish nodes in LER1) it is possible to deliver slightly more packets than without any protection. We can see that in all cases nodes naturally run out of battery after forwarding constant amount of packets.

4.2.5 Malicious Nodes

As we have already mentioned in the previous chapter, protection against malicious nodes was not the main motivation for systems considered in our comparison. However we decided

to include a set of tests measuring protection against presence of nodes intentionally changing contents of packets they were requested to forward.

	None	Watchdog	Context Aware	CORE	SORI	OCEAN
MN1	661,3	654	614,4	631,9	631,9	631,9
MN2	381,2	352,6	325,5	376,7	376,7	376,7
MN3	171,4	156	146,9	165,2	165,2	165,2
MN4	97,2	81,3	93,6	98,6	98,6	98,6
MN5	58,6	41,8	56,1	60,9	60,9	60,9

Table 8 Total number of packets correctly delivered to the base station in presence of malicious nodes

Averaged results in Table 8 show that considered reputation systems do not provide protection against misbehaving nodes. On the contrary, tendency to route all packets via few more trusted nodes cause more packets to be compromised when there is a misbehaving node in such a route.

However in the next chapter we will see that it is possible to efficiently use reputation systems to protect packets against intentional changes if this aim is taken into account when designing a reputation system.

5 Protection against Malicious Nodes

5.1 New Reputation System Design

While analyzing results from the previous chapter we made several observations about reputation systems:

1. Evaluating reputation based on observations of forwarding of routing packets is not very reliable as none of the considered systems was able to correctly distinguish between communication errors and intentional packet dropping (Section 4.2.2)
2. There is no real difference between systems punishing misbehaving nodes by dropping their packets and other systems. (Section 4.2.1)
3. Existing reputation systems did not prove that they could significantly improve routing efficiency.

We came to conclusion that even though the idea behind reputation systems is promising, their impact on routing functions is quite limited. However we decided to apply the reputation approach on another issue: We tried to improve integrity in a network where no digital signatures or encryption are used and data are transferred in plain text form. Using effective reputation system might be a low energy and low cost solution for such networks.

We can summarize the main principles of our new reputation system design as follows:

- Only integrity of forwarded packets is observed by neighboring nodes as it is the only observation which can be reliable enough in real deployment.
- Instead of separate alarm packets or Route Request packets we use Route Reply packets to distribute routing information throughout the network. This idea arises from a characteristic communication pattern of wireless sensor networks, where a node A is sending packets to the base station BS. In order to find a route it sends Route Request towards the base station and receives reply which is forwarded by nodes between the base station and the node A. As these nodes have the best knowledge about behavior of nodes between A and BS, appending reputation information at the end of Route Reply packet is the fastest way of spreading the information towards potential users of the route.
- In order to ensure data integrity it is vital to avoid routing via malicious nodes. While evaluating possible paths at the end of route discovery, the rating of a single path is not the average value of node reputation, but the lowest value.

5.2 Protocol Specification

We were experimenting with several versions of protocol, especially with the parameters of the reputation value and with selection of reputation information which should be spread around the network. Finally we came to reputation system described in this section.

Watchdog: Every node observes communication of its neighbors. When it sends a packet forward request to some neighbor or overhears some neighbor asking another neighbor to send a packet, it stores a copy of the packet into its internal buffer. When it overhears the packet being forwarded it checks its content. If the content corresponds to the packet in the buffer, it is a *positive observation* (value 10.0). If the content is different, it is a *negative observation* (value 1.0). If the node does not hear the packet being forward in a given timeout it is removed from the buffer.

Every node preserves its own *reputation table*. For every node it knows about in the network it stores its reputation value and a confidence value of this record. Reputation of a node it encounters for the first time is 5.0 with confidence 1.0.

When a node receives or is requested to forward a Route Reply packet, it checks its contents for *notifications* from other nodes. Such notification consists of address of the reporting node, address of the observed node, reputation value and confidence value. A node stores only the most actual notification of each node about each other particular node.

Whenever a node makes new observation or receives new notification it updates reputation value of the node. Reputation value is evaluated using following function:

$$\boldsymbol{N_S} = \sum (\boldsymbol{Values\ of\ node's\ own\ observations})$$

$$\boldsymbol{D_S} = (\boldsymbol{Number\ of\ own\ observations})$$

$$\boldsymbol{N_N} = \sum_{\boldsymbol{notifications}} (\boldsymbol{Reputation\ value}(\boldsymbol{reporter}) * \boldsymbol{Confidence\ of\ notification} * \boldsymbol{Notified\ reputation\ value})$$

$$\boldsymbol{D_N} = \sum_{\boldsymbol{notifications}} (\boldsymbol{Reputation\ value}(\boldsymbol{reporter}) * \boldsymbol{Confidence\ of\ notification})$$

$$\boldsymbol{Reputation\ value} = \frac{\boldsymbol{10.0 * N_S + N_N}}{\boldsymbol{10.0 * D_S + D_N}}$$

In other words, the new reputation value is a weighted average of all observations and notifications. Values are weighted their corresponding confidence values and reputation of reporter. Node's own reputation is set to 10.0 and confidence value is a number of own observations.

When a node sends or forwards Route Reply packet, it appends reputation record of the node it has the latest observation of. This way the most actual reputation information is spread by the node with the best knowledge of the neighbor.

When the reputation system is asked by Routing module to choose the best path for a packet, it calculates rating of every path as:

$$\boldsymbol{Path\ rating = min(Reputation\ values\ of\ the\ nodes\ in\ the\ path)}$$

Path with the highest rating is chosen as the packet route.

5.3 Simulation Results

We simulated our new reputation system using tests MN1 – MN5 from our former simulation.

	Unprotected Network	New Reputation System
MN1	661,3	666,3
MN2	381,2	385
MN3	171,4	183,7
MN4	97,2	101,9
MN5	58,6	61,9

Table 9 Number of correctly received packets in unprotected network and in network protected with our new reputation system

As we can see in Table 9 our reputation system can improve data integrity in a network with malicious nodes by up to 7 % in the MN3 scenario (50% of malicious nodes). This result is better than the results of reputation systems focused on routing functions and shows that the reputation system can be used as a low cost way of improving data integrity in the network. However the improvement is not as significant as to ensure sufficient integrity on its own.

6 Conclusion

The aim of our research was to evaluate capabilities of reputation systems in wireless sensor networks. We selected several systems and made a theoretical comparison.

In the next step we prepared a complex wireless sensor network simulation framework SensNet, which can also be used for a large variety of simulations in the future. We implemented selected systems and carried out a series of tests of our systems.

While analyzing the results of the simulations we found that reputation systems can slightly improve performance of wireless sensor networks. We made several observations on differences between selected systems. We stated that the idea of punishing nodes considered misbehaving by dropping their packets actually deteriorates network performance and we found it more efficient for a network to forward packets via such nodes even if there is a risk that the packets will get lost.

We identified the main obstacle of reputation system operation, which is the unreliability of packet overhearing. In a simulation which is closer to real conditions, with background noise and transmission errors, it is difficult for a reputation system to distinguish between intentional packet dropping and communication error. Reputation systems which rely mainly on observations of broadcasted Route Request packets are not very effective. In general we observed the highest increase of network performance by only 4 %.

Because of this unreliability we stated that the only observation a node can make is observation of forwarded packet integrity. Thus we tried to focus on a slightly different issue and explored possibility of using reputation systems to ensure higher integrity of data forwarded in otherwise unprotected network. Using observations from previous experiments we decided on basic principles of a new reputation system. We introduced a new way of spreading reputation information and defined a function for evaluating reputation values. We showed in simulation that using our reputation system in a network with 50 % percent of nodes malicious it is possible to increase number of correctly delivered packets by 7 %.

In general reputation systems proved to be a possibly working way of improving security in wireless sensor networks. While the increase of network performance in presence of selfish nodes was not very significant, we believe that after further research and improvements reputation systems could be used as a low-cost way of improving integrity of data in networks with lower security requirements.

Bibliography

1. **Palafox, Luis E. and Garcia-Macias, J. Antonio.** Security in Wireless Sensor Networks. [book auth.] Yan Zhang, Jun Zheng and Miao Ma. *Handbook of Research on Wireless Security*. s.l. : Information Science Reference, 2008.

2. **Romer, Kay and Mattern, Friedemann.** The design space of wireless sensor networks. *Wireless Communications, IEEE*. 12 2004, Vol. 11, 6, pp. 54-61.

3. **Johnson, David B.** Routing in Ad Hoc Networks of Mobile Hosts. *Proceedings of the Workshop on Mobile Computing Systems and Applications*. 12 1994, pp. 158-163.

4. **Dolev, D. and Yao, A.** On the security of public key protocols. *IEEE Transactions on Information Theory*. 1983, Vol. 29, 2, pp. 198 - 208.

5. **Sobeslavsky, Petr.** *Key Management in Wireless Sensor Networks*. Faculty of Informatics, Masaryk University. Brno : s.n., 2008. Term Project.

6. **Buchegger, Sonja and Le Boudec, Jean-Yves.** Nodes bearing grudges: towards routing security, fairness, and robustness in mobile ad hoc networks. *10th Euromicro Workshop on Parallel, Distributed and Network-Based Processing: Proceedings*. Canary Islands : Inst Elect & Electronic Engineers, 2002, pp. 403-410.

7. —. Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes? Fairness In Dynamic Ad-hoc Networks. *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*. 2002, pp. 226-236.

8. **Marti, S., et al.** Mitigating routing misbehavior in mobile ad hoc networks. *International Conference on Mobile Computing and Networking: Proceedings of the 6 th annual international conference on Mobile computing and networking*. 2000, Vol. 6, 11, pp. 255-265.

9. **Krishna, Paul.** Context aware detection of selfish nodes in DSR based ad-hoc networks. *Proceedings of IEEE GLOBECOM*. 2002, pp. 178--182.

10. **Michiardi, P. and Molva, R.** CORE: A Collaborative Reputation Mechanism to enforce node cooperation in Mobile Ad hoc Networks. *Advanced Communications and Multimedia Security: IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*. 9 2002.

11. **He, Q., Wu, D. and Khosla, P.** SORI: A Secure and Objective Reputation-based Incentive Scheme for Ad-hoc Networks. *IEEE Wireless Communications and Networking Conference (WCNC 2004)*. 2004.

12. **Bansal, S. and Baker, M.** Observation-based Cooperation Enforcement in Ad Hoc Networks. *Arxiv preprint cs.NI/0307012*. 2003.

13. *OMNeT++ Discrete Event Simulation System*. [Online] [Cited: 11 1, 2008.] http://www.omnetpp.org/.

14. *Mobility Framework for OMNeT++*. [Online] [Cited: 11 1, 2008.] http://mobility-fw.sourceforge.net/.

15. **Shnayder, V., et al.** Simulating the power consumption of large-scale sensor network applications. *Proceedings of the 2nd international conference on Embedded networked sensor systems*. NY : ACM New York, 2004, pp. 188-200.

16. **Johnson, David B., Maltz, David A. and Hu, Yih-Chun.** The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. [Online] 2 2007. [Cited: 11 20, 2008.] http://tools.ietf.org/html/rfc4728.

17. **Wikipedia contributors.** Wireless sensor network. *Wikipedia*. [Online] [Cited: 11 1, 2008.] http://en.wikipedia.org/wiki/Wireless_sensor_network.

18. **Varga, Andras.** OMNeT++ 3.2 User Manual. *OMNeT++*. [Online] [Cited: 11 1, 2008.] http://www.omnetpp.org/doc/manual/usman.html.

19. **Camp, T., Boleng, J. and Davies, V.** A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*. 2002, Vol. 2, 5, pp. 483-502.

Appendix A – Simulation Results

This appendix consists of several tables of averaged test results for every reputation system in the network. The numbers were used as input for analysis in the fourth, fifth and sixth chapter of this paper.

These tables as well as detailed results from every single test run can be found in original CSV files and Excel files on the attached CD.

Average Results of All Test Runs in Unprotected Network

Test scenario	All packets sent to sink			Packets sent by selfish nodes			Packets sent by non-selfish nodes		
	Packets sent	Packets received	Percentage of received packets	Packets sent	Packets received	Percentage of received packets	Packets sent	Packets received	Percentage of received packets
NSN1	1 000,0	910,9	91,1	0,0	0,0	0,0	1 000,0	910,9	91,1
NSN2	1 000,0	689,8	69,0	200,0	131,4	65,7	800,0	558,4	69,8
NSN3	1 000,0	443,8	44,4	400,0	170,9	42,7	600,0	272,9	45,5
NSN4	1 000,0	291,9	29,2	600,0	161,6	26,9	400,0	130,3	32,6
NSN5	1 000,0	191,0	19,1	800,0	148,4	18,6	200,0	42,6	21,3
NSN6	1 000,0	123,2	12,3	1 000,0	123,2	12,3	0,0	0,0	0,0
NC1	500,0	329,5	65,9	100,0	65,4	65,4	400,0	264,1	66,0
NC2	1 000,0	689,8	69,0	200,0	131,4	65,7	800,0	558,4	69,8
NC3	2 000,0	1 458,6	72,9	400,0	283,0	70,8	1 600,0	1 175,6	73,5
NC4	4 000,0	2 789,6	69,7	800,0	531,5	66,4	3 200,0	2 258,1	70,6
NC5	8 000,0	5 534,1	69,2	1 600,0	1 072,6	67,0	6 400,0	4 461,5	69,7
NC6	16 671,0	11 666,2	70,0	3 333,3	2 275,2	68,3	13 333,4	9 391,0	70,4
SM1	1 000,0	689,8	69,0	200,0	131,4	65,7	800,0	558,4	69,8
SM2	1 000,0	687,8	68,8	200,0	144,6	72,3	800,0	543,2	67,9
SM3	1 000,0	573,1	57,3	200,0	119,2	59,6	800,0	453,9	56,7
SM4	1 000,0	476,8	47,7	200,0	100,9	50,5	800,0	375,9	47,0
SM5	1 000,0	504,5	50,5	200,0	99,3	49,7	800,0	405,2	50,7
SM6	1 000,0	498,3	49,8	200,0	100,8	50,4	800,0	397,5	49,7
DP1	1 000,0	786,6	78,7	200,0	151,3	75,7	800,0	635,3	79,4
DP2	1 000,0	711,9	71,2	200,0	137,3	68,7	800,0	574,6	71,8
DP3	1 000,0	672,7	67,3	200,0	131,7	65,9	800,0	541,0	67,6
DP4	1 000,0	682,6	68,3	200,0	133,7	66,9	800,0	548,9	68,6
DP5	1 000,0	739,5	74,0	200,0	142,5	71,3	800,0	597,0	74,6
NCB	1 000,0	282,5	28,3	768,7	144,5	18,8	231,3	138,0	59,7

Test scenario	All packets sent to sink			Percentage of received packets	Percentage of packets received correctly	Number of packet changes
	Packets sent	Packets received	Packets received correctly			
MN1	1000	911,2	661,3	91,1	66,1	313,8
MN2	1000	911,2	381,2	91,1	38,1	1 274,6
MN3	1000	911,2	171,4	91,1	17,1	2 418,7
MN4	1000	911,2	97,2	91,1	9,7	3 471,2
MN5	1000	911,2	58,6	91,1	5,9	4 768,7

		All packets	Sent by selfish nodes	Sent by non-selfish
LER1	Packets received:	413,6	78,7	334,9
LER2	Packets received:	173,9	94,2	79,7

Average Results of All Test Runs with Watchdog and Pathrater

Test scenario	All packets sent to sink			Packets sent by selfish nodes			Packets sent by non-selfish nodes		
	Packets sent	Packets received	Percentage of received packets	Packets sent	Packets received	Percentage of received packets	Packets sent	Packets received	Percentage of received packets
NSN1	1 000,0	911,5	911,5	0,0	0,0	0,0	1 000,0	911,5	911,5
NSN2	1 000,0	707,7	707,7	200,0	136,8	136,8	800,0	570,9	570,9
NSN3	1 000,0	454,7	454,7	400,0	172,0	172,0	600,0	282,7	282,7
NSN4	1 000,0	274,2	274,2	600,0	147,3	147,3	400,0	126,9	126,9
NSN5	1 000,0	191,1	191,1	800,0	149,1	149,1	200,0	42,0	42,0
NSN6	1 000,0	118,5	118,5	1 000,0	118,5	118,5	0,0	0,0	0,0
NC1	500,0	333,4	333,4	100,0	64,0	64,0	400,0	269,4	269,4
NC2	1 000,0	707,7	707,7	200,0	136,8	136,8	800,0	570,9	570,9
NC3	2 000,0	1 424,8	1 424,8	400,0	273,5	273,5	1 600,0	1 151,3	1 151,3
NC4	4 000,0	2 724,1	2 724,1	800,0	520,6	520,6	3 200,0	2 203,5	2 203,5
NC5	8 000,0	5 457,1	5 457,1	1 600,0	1 056,4	1 056,4	6 400,0	4 400,7	4 400,7
NC6	16 671,0	11 715,9	11 715,9	3 333,3	2 249,6	2 249,6	13 333,4	9 466,3	9 466,3
SM1	1 000,0	707,7	707,7	200,0	136,8	136,8	800,0	570,9	570,9
SM2	1 000,0	643,2	643,2	200,0	132,1	132,1	800,0	511,1	511,1
SM3	1 000,0	601,7	601,7	200,0	124,4	124,4	800,0	477,3	477,3
SM4	1 000,0	496,8	496,8	200,0	105,4	105,4	800,0	391,4	391,4
SM5	1 000,0	501,1	501,1	200,0	102,7	102,7	800,0	398,4	398,4
SM6	1 000,0	496,5	496,5	200,0	99,8	99,8	800,0	396,7	396,7
DP1	1 000,0	799,1	799,1	200,0	156,1	156,1	800,0	643,0	643,0
DP2	1 000,0	684,0	684,0	200,0	132,4	132,4	800,0	551,6	551,6
DP3	1 000,0	651,4	651,4	200,0	122,7	122,7	800,0	528,7	528,7
DP4	1 000,0	660,4	660,4	200,0	125,8	125,8	800,0	534,6	534,6
DP5	1 000,0	740,6	740,6	200,0	142,7	142,7	800,0	597,9	597,9
NCB	1 000,0	268,5	26,9	771,6	134,6	17,4	228,4	133,9	58,6

Test scenario	All packets sent to sink			Percentage of received packets	Percentage of packets received correctly	Number of packet changes
	Packets sent	Packets received	Packets received correctly			
MN1	1000	911,7	654	91,2	65,4	320,8
MN2	1000	910	352,6	91,0	35,3	1 356,6
MN3	1000	907,5	156	90,8	15,6	2 732,5
MN4	1000	908,9	81,3	90,9	8,1	4 030,9
MN5	1000	908,9	41,8	90,9	4,2	5 558,7

		All packets	Sent by selfish nodes	Sent by non-selfish
LER1	Packets received:	420,2	82	338,2
LER2	Packets received:	166,6	88,2	78,4

Average Results of All Test Runs with Context Aware Detection

Test scenario	All packets sent to sink			Packets sent by selfish nodes			Packets sent by non-selfish nodes		
	Packets sent	Packets received	Percentage of received packets	Packets sent	Packets received	Percentage of received packets	Packets sent	Packets received	Percentage of received packets
NSN1	1 000,0	908,0	90,8	0,0	0,0	0,0	1 000,0	908,0	90,8
NSN2	1 000,0	673,1	67,3	200,0	130,9	65,5	800,0	542,2	67,8
NSN3	1 000,0	437,2	43,7	400,0	165,4	41,4	600,0	271,8	45,3
NSN4	1 000,0	265,8	26,6	600,0	147,5	24,6	400,0	118,3	29,6
NSN5	1 000,0	179,8	18,0	800,0	140,2	17,5	200,0	39,6	19,8
NSN6	1 000,0	119,3	11,9	1 000,0	119,3	11,9	0,0	0,0	0,0
NC1	500,0	326,0	65,2	100,0	63,0	63,0	400,0	263,0	65,8
NC2	1 000,0	673,1	67,3	200,0	130,9	65,5	800,0	542,2	67,8
NC3	2 000,0	1 340,6	67,0	400,0	259,0	64,8	1 600,0	1 081,6	67,6
NC4	4 000,0	2 657,1	66,4	800,0	517,3	64,7	3 200,0	2 139,8	66,9
NC5	8 000,0	5 353,5	66,9	1 600,0	1 035,3	64,7	6 400,0	4 318,2	67,5
NC6	16 668,0	11 263,3	67,6	3 333,3	2 197,0	65,9	13 333,4	9 066,3	68,0
SM1	1 000,0	673,1	67,3	200,0	130,9	65,5	800,0	542,2	67,8
SM2	1 000,0	666,4	66,6	200,0	138,8	69,4	800,0	527,6	66,0
SM3	1 000,0	583,6	58,4	200,0	120,5	60,3	800,0	463,1	57,9
SM4	1 000,0	511,2	51,1	200,0	102,7	51,4	800,0	408,5	51,1
SM5	1 000,0	508,7	50,9	200,0	101,3	50,7	800,0	407,4	50,9
SM6	1 000,0	502,0	50,2	200,0	102,7	51,4	800,0	399,3	49,9
DP1	1 000,0	784,7	78,5	200,0	153,9	77,0	800,0	630,8	78,9
DP2	1 000,0	696,0	69,6	200,0	135,4	67,7	800,0	560,6	70,1
DP3	1 000,0	652,3	65,2	200,0	127,0	63,5	800,0	525,3	65,7
DP4	1 000,0	657,7	65,8	200,0	126,3	63,2	800,0	531,4	66,4
DP5	1 000,0	737,9	73,8	200,0	142,3	71,2	800,0	595,6	74,5
NCB	1 000,0	271,7	27,2	771,5	139,6	18,1	228,5	132,1	57,8

Test scenario	All packets sent to sink			Percentage of received packets	Percentage of packets received correctly	Number of packet changes
	Packets sent	Packets received	Packets received correctly			
MN1	1000	910,3	614,4	91,0	61,4	391,1
MN2	1000	910,3	325,5	91,0	32,6	1 394,1
MN3	1000	910,3	146,9	91,0	14,7	2 564,4
MN4	1000	910,3	93,6	91,0	9,4	3 662,3
MN5	1000	910,3	56,1	91,0	5,6	4 921,5

		All packets	Sent by selfish nodes	Sent by non-selfish
LER1	Packets received:	400,1	77,7	322,4
LER2	Packets received:	160,5	89,1	71,4

Average Results of All Test Runs with CORE

Test scenario	All packets sent to sink			Packets sent by selfish nodes			Packets sent by non-selfish nodes		
	Packets sent	Packets received	Percentage of received packets	Packets sent	Packets received	Percentage of received packets	Packets sent	Packets received	Percentage of received packets
NSN1	1 000,0	911,2	91,1	0,0	0,0	0,0	1 000,0	911,2	91,1
NSN2	1 000,0	692,7	69,3	200,0	130,6	65,3	800,0	562,1	70,3
NSN3	1 000,0	460,4	46,0	400,0	180,2	45,1	600,0	280,2	46,7
NSN4	1 000,0	279,6	28,0	600,0	154,2	25,7	400,0	125,4	31,4
NSN5	1 000,0	197,2	19,7	800,0	153,3	19,2	200,0	43,9	22,0
NSN6	1 000,0	119,3	11,9	1 000,0	119,3	11,9	0,0	0,0	0,0
NC1	500,0	342,1	68,4	100,0	64,9	64,9	400,0	277,2	69,3
NC2	1 000,0	692,7	69,3	200,0	130,6	65,3	800,0	562,1	70,3
NC3	2 000,0	1 444,4	72,2	400,0	277,6	69,4	1 600,0	1 166,8	72,9
NC4	4 000,0	2 646,0	66,2	800,0	512,0	64,0	3 200,0	2 134,0	66,7
NC5	8 000,0	5 604,8	70,1	1 600,0	1 097,0	68,6	6 400,0	4 507,8	70,4
NC6	16 671,0	11 537,0	69,2	3 333,3	2 225,8	66,8	13 333,4	9 311,2	69,8
SM1	1 000,0	692,7	69,3	200,0	130,6	65,3	800,0	562,1	70,3
SM2	1 000,0	664,5	66,5	200,0	138,1	69,1	800,0	526,4	65,8
SM3	1 000,0	573,0	57,3	200,0	117,5	58,8	800,0	455,5	56,9
SM4	1 000,0	515,3	51,5	200,0	106,7	53,4	800,0	408,6	51,1
SM5	1 000,0	495,9	49,6	200,0	100,3	50,2	800,0	395,6	49,5
SM6	1 000,0	508,7	50,9	200,0	104,9	52,5	800,0	403,8	50,5
DP1	1 000,0	790,1	79,0	200,0	153,6	76,8	800,0	636,5	79,6
DP2	1 000,0	672,9	67,3	200,0	130,1	65,1	800,0	542,8	67,9
DP3	1 000,0	650,0	65,0	200,0	121,8	60,9	800,0	528,2	66,0
DP4	1 000,0	644,0	64,4	200,0	120,9	60,5	800,0	523,1	65,4
DP5	1 000,0	740,4	74,0	200,0	142,7	71,4	800,0	597,7	74,7
NCB	1 000,0	269,8	27,0	771,2	135,4	17,6	228,8	134,4	58,7

Test scenario	All packets sent to sink			Percentage of received packets	Percentage of packets received correctly	Number of packet changes
	Packets sent	Packets received	Packets received correctly			
MN1	1000	911,1	631,9	91,1	63,2	341,5
MN2	1000	911,1	376,7	91,1	37,7	1 321,3
MN3	1000	911,1	165,2	91,1	16,5	2 507,0
MN4	1000	911,1	98,6	91,1	9,9	3 558,2
MN5	1000	911,1	60,9	91,1	6,1	4 812,7

		All packets	Sent by selfish nodes	Sent by non-selfish
LER1	Packets received:	409,5	76,9	332,6
LER2	Packets received:	170,1	94,1	76,0

Average Results of All Test Runs with SORI

Test scenario	All packets sent to sink			Packets sent by selfish nodes			Packets sent by non-selfish nodes		
	Packets sent	Packets received	Percentage of received packets	Packets sent	Packets received	Percentage of received packets	Packets sent	Packets received	Percentage of received packets
NSN1	1 000,0	911,2	91,1	0,0	0,0	0,0	1 000,0	911,2	91,1
NSN2	1 000,0	692,7	69,3	200,0	130,6	65,3	800,0	562,1	70,3
NSN3	1 000,0	460,4	46,0	400,0	180,2	45,1	600,0	280,2	46,7
NSN4	1 000,0	279,6	28,0	600,0	154,2	25,7	400,0	125,4	31,4
NSN5	1 000,0	197,2	19,7	800,0	153,3	19,2	200,0	43,9	22,0
NSN6	1 000,0	119,3	11,9	1 000,0	119,3	11,9	0,0	0,0	0,0
NC1	500,0	342,1	68,4	100,0	64,9	64,9	400,0	277,2	69,3
NC2	1 000,0	692,7	69,3	200,0	130,6	65,3	800,0	562,1	70,3
NC3	2 000,0	1 444,4	72,2	400,0	277,6	69,4	1 600,0	1 166,8	72,9
NC4	4 000,0	2 646,0	66,2	800,0	512,0	64,0	3 200,0	2 134,0	66,7
NC5	8 000,0	5 604,8	70,1	1 600,0	1 097,0	68,6	6 400,0	4 507,8	70,4
NC6	16 671,0	11 537,0	69,2	3 333,3	2 225,8	66,8	13 333,4	9 311,2	69,8
SM1	1 000,0	692,7	69,3	200,0	130,6	65,3	800,0	562,1	70,3
SM2	1 000,0	664,5	66,5	200,0	138,1	69,1	800,0	526,4	65,8
SM3	1 000,0	573,0	57,3	200,0	117,5	58,8	800,0	455,5	56,9
SM4	1 000,0	515,3	51,5	200,0	106,7	53,4	800,0	408,6	51,1
SM5	1 000,0	495,9	49,6	200,0	100,3	50,2	800,0	395,6	49,5
SM6	1 000,0	508,7	50,9	200,0	104,9	52,5	800,0	403,8	50,5
DP1	1 000,0	790,1	79,0	200,0	153,6	76,8	800,0	636,5	79,6
DP2	1 000,0	672,9	67,3	200,0	130,1	65,1	800,0	542,8	67,9
DP3	1 000,0	650,0	65,0	200,0	121,8	60,9	800,0	528,2	66,0
DP4	1 000,0	644,0	64,4	200,0	120,9	60,5	800,0	523,1	65,4
DP5	1 000,0	740,4	74,0	200,0	142,7	71,4	800,0	597,7	74,7
NCB	1 000,0	269,8	27,0	771,2	135,4	17,6	228,8	134,4	58,7

Test scenario	All packets sent to sink			Percentage of received packets	Percentage of packets received correctly	Number of packet changes
	Packets sent	Packets received	Packets received correctly			
MN1	1000	911,1	631,9	91,1	63,2	341,5
MN2	1000	911,1	376,7	91,1	37,7	1 321,3
MN3	1000	911,1	165,2	91,1	16,5	2 507,0
MN4	1000	911,1	98,6	91,1	9,9	3 558,2
MN5	1000	911,1	60,9	91,1	6,1	4 812,7

		All packets	Sent by selfish nodes	Sent by non-selfish
LER1	Packets received:	409,5	76,9	332,6
LER2	Packets received:	170,1	94,1	76,0

Average Results of All Test Runs with OCEAN

Test scenario	All packets sent to sink			Packets sent by selfish nodes			Packets sent by non-selfish nodes		
	Packets sent	Packets received	Percentage of received packets	Packets sent	Packets received	Percentage of received packets	Packets sent	Packets received	Percentage of received packets
NSN1	1 000,0	911,2	91,1	0,0	0,0	0,0	1 000,0	911,2	91,1
NSN2	1 000,0	694,7	69,5	200,0	133,6	66,8	800,0	561,1	70,1
NSN3	1 000,0	461,0	46,1	400,0	180,2	45,1	600,0	280,8	46,8
NSN4	1 000,0	280,6	28,1	600,0	152,9	25,5	400,0	127,7	31,9
NSN5	1 000,0	196,9	19,7	800,0	152,8	19,1	200,0	44,1	22,1
NSN6	1 000,0	119,3	11,9	1 000,0	119,3	11,9	0,0	0,0	0,0
NC1	500,0	339,3	67,9	100,0	64,6	64,6	400,0	274,7	68,7
NC2	1 000,0	694,7	69,5	200,0	133,6	66,8	800,0	561,1	70,1
NC3	2 000,0	1 433,5	71,7	400,0	274,1	68,5	1 600,0	1 159,4	72,5
NC4	4 000,0	2 609,3	65,2	800,0	503,1	62,9	3 200,0	2 106,2	65,8
NC5	8 000,0	5 543,4	69,3	1 600,0	1 074,7	67,2	6 400,0	4 468,7	69,8
NC6	16 671,0	11 461,9	68,8	3 333,3	2 219,0	66,6	13 333,4	9 242,9	69,3
SM1	1 000,0	694,7	69,5	200,0	133,6	66,8	800,0	561,1	70,1
SM2	1 000,0	651,2	65,1	200,0	135,0	67,5	800,0	516,2	64,5
SM3	1 000,0	571,7	57,2	200,0	117,3	58,7	800,0	454,4	56,8
SM4	1 000,0	499,5	50,0	200,0	102,9	51,5	800,0	396,6	49,6
SM5	1 000,0	495,9	49,6	200,0	100,3	50,2	800,0	395,6	49,5
SM6	1 000,0	508,7	50,9	200,0	104,9	52,5	800,0	403,8	50,5
DP1	1 000,0	790,1	79,0	200,0	153,6	76,8	800,0	636,5	79,6
DP2	1 000,0	672,5	67,3	200,0	130,2	65,1	800,0	542,3	67,8
DP3	1 000,0	650,8	65,1	200,0	124,1	62,1	800,0	526,7	65,8
DP4	1 000,0	646,7	64,7	200,0	120,4	60,2	800,0	526,3	65,8
DP5	1 000,0	740,4	74,0	200,0	142,7	71,4	800,0	597,7	74,7
NCB	1 000,0	269,0	26,9	771,1	134,3	17,4	228,9	134,7	58,8

Test scenario	All packets sent to sink			Percentage of received packets	Percentage of packets received correctly	Number of packet changes
	Packets sent	Packets received	Packets received correctly			
MN1	1000	911,1	631,9	91,1	63,2	341,5
MN2	1000	911,1	376,7	91,1	37,7	1 321,3
MN3	1000	911,1	165,2	91,1	16,5	2 507,0
MN4	1000	911,1	98,6	91,1	9,9	3 558,2
MN5	1000	911,1	60,9	91,1	6,1	4 812,7

		All packets	Sent by selfish nodes	Sent by non-selfish
LER1	Packets received:	408,7	77,2	331,5
LER2	Packets received:	170,0	93,7	76,3

Appendix B – Contents of the Attached CD

```
Final               - test scenario settings, reputation system settings
-- runRepSysSim.py  - starting script for simulation runs

RepSysSim           - source code of the reputation system simulation (Chapter 3)

Results             - simulation results
-- csv              - set of original .csv output files
-- Analysis.xlsx    - simulation results data and graphs (Chapter 5)
-- Output data.xlsx - results of all simulation runs
-- Results.xlsx     - averaged results, tables from Appendix A

SensNet             - source code of the SensNet framework (Chapter 4)

omnetpp-3.3-src.tgz - source code of the OMNeT++ 3.3 simulator

install.txt         - brief installation instructions for Linux and Windows

thesis.pdf          - this thesis
```