MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Neighbour-based intrusion detection in wireless sensor networks

SVOČ 2010

Mgr. Lukáš Folkman

Brno, 2010

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. Andriy Stetsko

Acknowledgement

I would like to thank RNDr. Andriy Stetsko, the advisor of this work, for his valuable comments, suggestions and time he spent helping me with this work. I would like to thank Bc. Veronika Neužilová for her language concerned advice and moral support and my parents for supporting my university studies. Furthermore, I would like to thank Faculty of Informatics at Masaryk University for providing me a great study opportunity and potentiating me to take part in SVOČ 2010.

Abstract

Sensor nodes situated spatially close to each other tend to have similar behaviour. The neighbour-based detection technique is based on this principle and should provide means for anomaly intrusion detection in wireless sensor networks without prior training. Recently, this technique has been successfully applied to detect the fabricated information attack in wireless sensor networks.

This work provides the analysis of the symptoms of jamming, hello flood, selective forwarding, sinkhole, sybil, packet alteration and fabricated information attacks for the applicability of the neighbour-based technique. Furthermore, a neighbour-based intrusion detection system is designed and implemented for the operating system TinyOS. The intrusion detection system comes in two modifications – one with local knowledge of immediate neighbours only and one involving information exchanged among 1-hop or 2-hop neighbours. Collaboration is employed in order to refine information about the activity of neighbouring nodes. The accuracy of the technique was evaluated in detection of jamming, hello flood and selective forwarding attacks. Results of the simulations, namely the number of false negatives, false positives and correct warnings, are involved in this work as well.

The results presented in this SVOČ were submitted as the author's master thesis in January 2010. The topic was further researched and a publication involving the results from this work will be submitted to a conference in May 2010. Furthermore, a comprehensive technical report will be published in May 2010.

Keywords

hello flood attack, jamming, neighbour-based intrusion detection, selective forwarding, wireless sensor network

Contents

Introduction

Wireless sensor networks are composed of tiny nodes that are supposed to provide some physical measurements about their surroundings. They are left unattended in their hostile environment and are not equipped with any tamper proof mechanisms. A malicious adversary might be capable of compromising some of the nodes and even retrieve the cryptographic material from them. Intrusion detection systems are deployed in wireless sensor networks in order to secure them.

The neighbour-based intrusion detection technique is based on the principle that nodes situated spatially close to each other tend to have similar behaviour. If a node does not tend to behave similarly to its neighbouring nodes, it is considered an attacker. A neighbour-based intrusion detection system is designed and implemented in this work. It is capable of revealing selective forwarding, jamming and hello flood attacks. An effectiveness evaluation of the proposed intrusion detection system, namely the number of false negatives, false positives and correct warnings, using the simlator TOSSIM, is provided in this work as well.

The basic introduction to the topic of wireless sensor networks and their security issues is provided in the first chapter.

A description of intrusion detection systems together with their classification can be found in Chapter 2. It also discusses known implementations of the neighbour-based detection technique.

The description of jamming, hello flood, selective forwarding, sinkhole, sybil, packet alteration and fabricated information attacks is summarized in the third chapter. It describes symptoms and statistics which can be used by intrusion detection systems in order to reveal these attacks. Discussion on applicability of these statistics for implementation of a neighbour-based intrusion detection system is included.

In the fourth chapter, design and implementation issues of our system are depicted. It provides motivation for introducing the clustering method used in detections in the Collection tree protocol operating networks as well as for usage of collaboration with neighbours which can refine statistics used for detections of the attacks.

The fifth chapter describes performed simulations in the TOSSIM simulator for TinyOS networks. An effectiveness evaluation of the proposed intrusion detection system can be found there. Deployment of our system and its configuration options are described in Chapter 5 too.

We conclude this work in the last chapter and mention the possible future work.

The PowerTOSSIM plug-in for TOSSIM is used to simulate energy consumption of software running on sensor nodes. The installation and usage instructions of PowerTOSSIM as well as energy consumption estimation of our intrusion detection system are enclosed in the appendix.

Chapter 1

Wireless sensor networks

Wireless sensor networks are composed of a large number of tiny nodes that are used to measure some physical or environmental aspect of the hostile environment such as temperature, sound, vibration or motion. These nodes are resource-constrained units that communicate via a wireless medium and forward sensed data to the gateway node (depicted in Figure 1.1). The gateway node, so called base station, is the only connection with the other world apart from the network itself. Wireless sensor networks serve as a bridge from the physical world to the computer system by providing measurements of physical properties of the real world. Although wireless sensor networks were originally designed for the purpose of military application, nowadays their field of application is much wider and they are being used in civilian and industrial areas as well (healthcare applications, traffic control, home automation, industrial process monitoring or wildlife monitoring).

The basic description of wireless sensor networks, sensor nodes and gateways, software, main security aspects and depiction of security services used in wireless sensor networks is provided in this chapter. Most of the information is gained from [1, 2, 3, 4].

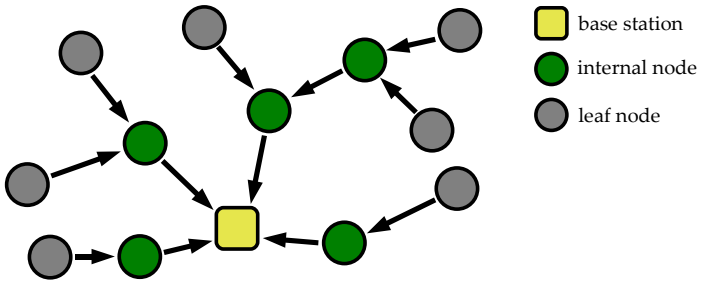


Figure 1.1: Wireless sensor network depiction

Wireless sensor networks (WSNs) are homogeneous distributed ad-hoc networks. However, there are several important differences from the classical ad-hoc networks [5]. Firstly, all of the nodes are independent and left unattended in their hostile environment without the presence of a human user. Their computing resources and batteries are more constrained, hence any of the nodes may fail to operate due to energy exhaustion. The application of a WSN is very specific depending on what kind of aspect is being monitored by the network. Furthermore, node density is much higher than in traditional ad-hoc networks. This is also a consequence of the fact that a node might disappear from the network as stated above.

Another important issue is security vulnerability of nodes deployed with no constraint of physical access to them and hence they might be captured by an attacker or might fail to operate. A WSN is highly distributed and its nodes are independent, self-configurable, capable of establishing the routing subsystem without any priorly given infrastructure and able to cooperate with each other.

A sensor network topology may be flat or hierarchical. In case of a hierarchical topology, the network is divided into clusters. Each cluster has a cluster head node which is usually a more powerful node. Cluster heads usually take some responsibilities for network maintenance as for example intrusion detection systems can be installed on these nodes. However, they become a single point of failure. In order to prevent this problem, flat sensor networks are deployed.

1.1 Sensor nodes and gateways

Sensor nodes, often referred to as motes, are basic units of WSNs and are capable of processing, collecting sensed data and communicating with their neighbours in the network. A node consists of a sensor, micro-controller, memory unit, transceiver and power source. A variety of sensor types is used depending on the application of the network. They might monitor radiation, temperature, light, movement, sound, humidity, pressure, etc. An analogue signal received by a sensor is digitalized and sent to the processing unit. There are several choices for wireless transmission. Optical communication is rarely used due to the requirement of line-of-sight among communicating nodes and inconvenient broadcasting. Sensor nodes usually make use of the ISM band for the radio wave transmission and operate with communication frequencies between 433 MHz and 2.4 GHz. For this purpose, a transceiver, which is a unit capable of both transmitting and receiving a radio signal, forms a part of every node. Different types of batteries and capacitors are used as power sources for sensor nodes. They might be classified by the type of the electrode, their size, whether they are re-chargeable or not. Some motes are able to renew their batteries using solar energy or thermo-generators.

There is a wide variety of manufactures of both sensor and gateway nodes as for example Moteiv. The intrusion detection system implemented in this work can be compiled for the Moteiv's Tmote Sky node. Tmote Sky comes with 8 MHz Texas Instruments MSP430 F1611 micro-controller, 10 kB of RAM and another 48 kB of flash memory. The micro-controller is a 16-bit RISC ultra low power processor which runs with extremely low active and sleep energy consumption. IEEE 802.15.4 compliant Chipcon Wireless Transceiver operates at the ISM 2.4 GHz frequency band with a data rate of 250 kbit/s. The mote has integrated humidity, temperature and light sensors [6].

A gateway node, usually referred to as a base station, provides a connection to the outer world, most commonly to the Internet. It is a much more powerful device than a sensor node and its energy resources are more long-lasting too. It serves as a collector of the measured values and security alerts from the motes in the network it is governing. Since the base station is connected to the Internet, it is maintained by a human user. As an example, the

current configuration of the Stargate NetBridge from Crossbow Technology, Inc. is the Intel IXP420 XScale processor with a frequency of 266 MHz, 32 MB RAM, 2 GB flash disk and one wired Ethernet port. It runs the Debian Linux operating system [7].

1.2 Software

Wireless sensor networks run specialized operating systems for which applications can be written. Contiki, Mantis, BTnut, SOS and Nano-RK are some of the operating systems that allow for writing application programmes in the C language [1]. The most commonly used operating system in wireless sensor networks is TinyOS [8]. The TinyOS operating system is an embedded, open-source, component-based operating system for wireless sensor networks. It is written in a dialect of the C language – the nesC programming language. NesC provides an event-based programming model where programmes are composed of event handlers and tasks. The TinyOS project [9] was rooted at the University of California, Berkeley. Nowadays, an international consortium, the TinyOS Alliance, supports the academic and industrial community of TinyOS developers and contributors. The current version of TinyOS is 2.1.0.

For the purpose of the WSN development and simulating TinyOS networks, a discrete event simulator, called TOSSIM, was created at the University of California, Berkeley. TOSSIM scales to thousands of motes and compiles directly from the TinyOS source code. The applications written in nesC for TinyOS are built into the compilation. TOSSIM is run on personal computers. PowerTOSSIM was created originally at the Harvard University [10]. It is used to simulate power consumption of each node in the simulation.

There comes the Dissemination protocol (DIP) and the Collection tree protocol (CTP) as a part of the TinyOS distribution. The first one is used for establishing eventual consistency on a variable shared by all the nodes in the network. The CTP is used to collect data at the base station (root of the tree) from any node. If all the nodes send data periodically, the CTP creates heavy traffic as every internal node has to forward each packet it receives up the tree. A data aggregation protocol can be used on top of or instead of the CTP in order to decrease the number of sent packets and hence prolong the battery lifetime. However, no data aggregation protocol implementation can be found in the current TinyOS distribution.

1.3 Security

Security of wireless sensor networks is an important factor of their use as in any other type of network. Information confidentiality, authentication, integrity, availability and freshness are required to be achieved. Graceful degradation is required as well which ensures that in case that a small number of nodes is compromised, the rest of the network is able to carry on its duty and continue functioning. The traditional cryptography is usually used to achieve these properties in classical networks. Research in this field is ongoing for wireless sensor networks too. However, there are several problems that make use of cryptography more complicated. In the first place, it is constrained capability of a node and requirement

for its price to be as low as possible (networks consist of thousands of nodes). The ad-hoc infrastructure-less nature of sensor networks makes the problem more challenging as well because there is no trusted central authority there. Furthermore, nodes are left unattended and because of their desired low price, they cannot have any tamper resistant or reaction mechanisms built in them. An adversary is then able to capture a node, retrieve its cryptographic material and be aware of its internal state and control its communication with the rest of the network [4].

As described in [4], there are several attacker types considered in wireless sensor networks. A *passive* attacker is only able to read ongoing communication, gather it, analyse and possibly extract cryptographic keys using cryptography analysis. Defence against a passive attacker is usually the encryption of data traffic. On the other hand, there is an *active* attacker which can also alter or inject new messages into the network communication. They are able to destroy some messages as well. An active attacker can perform *external* or *internal* attacks (we talk of a malicious outsider or insider respectively). External attacks are run by an attacker that does not compose a part of the network. Employing encryption mechanisms is not enough here and security is enhanced by authentication and synchronization mechanisms. An internal attacker is a legitimate mote in the network and has access to all of the mote's key material. That is why cryptographic techniques cannot help defending against internal attacks. An *intrusion detection system* (*IDS*) has to be used in the network. An IDS monitors behaviour of the nodes in the network and alerts the gateway node in case there is a suspect of an internal attacker. Finally, attackers may be divided based on the type of the device they use. *Mote-class* attackers misuse motes to run their attacks. *Laptop-class* attackers use more powerful devices with higher radio transceiver power and longer battery life-time.

Chapter 2

Intrusion detection systems

A wireless sensor network is deployed without a predefined infrastructure and left unattended. It is required for a WSN to be inherently autonomous. This involves being able to react on certain unusual events and reconfigure the network without human assistance or as little assistance as possible. To achieve the self-reconfigurability property, a situation awareness mechanism needs to be installed in the network so it is aware of the unusual events which the WSN should react on. A situation awareness mechanism has to be lightweight because of limited computational and battery resources of tiny nodes [11].

Intrusion detection systems are installed in order to detect internal attackers in networks. As stated in [11], "the major task of an IDS is to monitor computer networks and systems to detect these eventual intrusions in the network, alert users after specific intrusions have been detected, and finally, if possible, reconfigure the network and mark the root of the problem as malicious". A classification of different types of intrusion detection systems and description of components of an IDS for WSNs is given in this chapter. At its end, two approaches how to implement a neighbour-based intrusion detection system that is able to monitor several network properties at a time are presented.

In classical networks, intrusion detection systems are mainly situated on powerful mainframes of network segments and are able to process efficiently all data coming from the segment on which they operate. Unfortunately, there are no such devices in the case of WSNs. It must be decided how to take the advantage of redundancy in means of the number of nodes and how to deal with low-performance processing. It is essential to find optimal distribution of performance, battery saving and robustness. Moreover, an access to the mainframes in classical networks can be physically limited which makes them a reliable source of information. As opposed to this advantage, information gained from IDSs running on sensor nodes needs to be filtered because of possible presence of malicious adversaries.

Authors of [11] use a metaphor that a WSN should be able to heal itself by which they mean that a WSN should be self-reconfigurable. They liken a wireless sensor network to a living body where nodes are cells of the body and a base station is the brain. In such reasoning, network's malicious intruders, the nodes captured by an attacker, represent diseases and viruses. This metaphor is good because the way by which diseases are discovered in living bodies is the same as in intrusion detection systems for wireless sensor networks. An IDS monitors network's behaviour and looks for symptoms of diseases, possible attacks.

2.1 IDS classification

The description of different kinds of IDS classification based on [4] are summarized in this section.

There are generally two types of intrusion detection – *anomaly* detection and *signature* (sometimes denoted as *misuse*) detection. A difference can be seen in the way they discover malicious nodes. Any unusual behavioural deviations in the network opposed to its normal behaviour is announced as an anomaly in case of the anomaly detection. An IDS of such a type has to be able to learn about the normal behaviour of the network. There is usually a start-up phase, often denoted as a training phase, of an IDS for this purpose and the IDS only gathers information about normal flow for some period of time. It has to be ensured that no intruders exist in the network during this phase which might be hard to achieve. "Signature based detection techniques match the known attack profiles with suspicious behaviours" as stated in [12]. For this purpose, attack footprints have to be defined for each type of the attack that should be recognized by the IDS.

Both anomaly and signature based IDSs have their pros and cons. A signature based detection is very effective in revealing known attacks whose patterns are defined in the IDS. However, it fails completely to uncover unknown attacks. They can be recognized by an anomaly detection though. Unfortunately, such an IDS requires training to learn what a normal traffic flow looks like and if network's dynamics have changed, the IDS has to be re-trained. Employing both detection techniques should provide an effective detection mechanism for a sensor network. Additionally, a *specification* detection is sometimes introduced as the third type of IDSs. It is very similar to an anomaly detection, however, the set of rules is defined a priori and so no training phase is involved. This work deals with the *neighbour-based* intrusion detection which is a specific type of the anomaly detection. The neighbour-based detection technique is well-described later in this chapter.

From another point of view, intrusion detection systems might be classified depending on their collaboration abilities into *collaborative* and *non-collaborative* (also referred to as *distributed* and *stand-alone* respectively). In case of a distributed IDS, a false information filtering system should be implemented as the IDS may collaborate with IDSs running on nodes captured by an adversary. Reputation schemes are often employed for this purpose. Stand-alone detection systems do not suffer with these problems. On the other hand, there might not be enough information gathered locally to decide some types of attacks. We designed and implemented both collaborative and non-collaborative modifications of an IDS for flat wireless sensor networks. More information is available in Chapter 4.

Collaborative IDSs are categorized as *peer-to-peer* and *hierarchical*. Peer-to-peer IDSs create high communication overhead. Hence, information should be distributed just in a small neighbourhood around a node. Hierarchical IDSs assume existence of nodes which take responsibilities of cluster heads which brings the problem of a single point of failure. An attacker who captures just a few nodes which are actually the cluster heads paralyses functioning of an IDS for the whole network. The IDS proposed in this work belongs to the peer-to-peer category of IDSs.

Moreover, there is a question on which node an IDS should be actively running at some point in time. The authors of [5] suggest to use the method of *spontaneous watchdogs*. Then, an IDS is installed on every node. When there is communication on a medium, one of the possible watchdogs (see Figure 2.1) for the communication is chosen to be active. A set of possible watchdogs is composed of all the nodes which are able to hear the communication. The selection of the active watchdog is implemented by a random choice in [5]. However, other implementations let possible watchdogs monitor the network in turn.

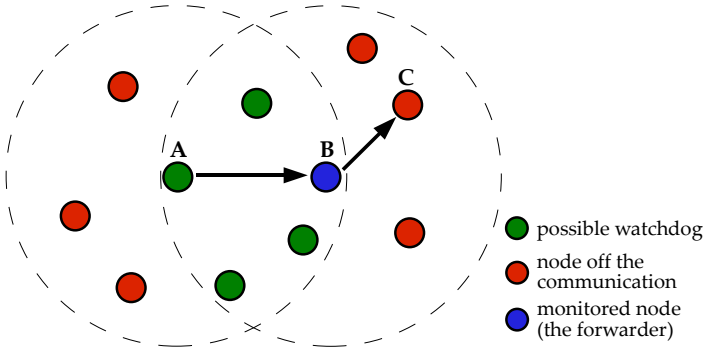


Figure 2.1: Depiction of the set of possible watchdogs for messages sent from A to C

2.2 Blueprint of an intrusion detection system architecture

There is a blueprint of an IDS architecture for WSNs described in [11]. It was adopted for design of the IDS proposed in this work. The authors claim that such a system would fulfil following properties: full network coverage, simplicity, usefulness, and extensibility. In other words, the system would cover the entire data flow in the network, be simple enough in order to run on limited motes, detect most attacks which it would be designed for and it would be possible to implement new mechanisms to detect new forms of attacks easily without the need to re-build the existing system.

The authors suggest to build the IDS as a powerful agent running on a base station and a lightweight agent running on every node. The base station agent would have access to information from all the nodes in the network gained using an appropriate collection protocol. On the other hand, agents running on nodes can operate only with the information from their neighbourhood. However, this information is very rich due to a wireless nature of communication. Every node upon receiving any message has to examine if it is destined to the node itself or some other node. Then, each node has information about all the data in its neighbourhood, not only data whose destination it is.

Furthermore, node agents are formed from *local* and *global agents*. The first one is responsible for monitoring local information on a node (measured value from its own sensor, carrier sensing time on a medium, etc.). The global agent analyses the information flow in its neighbourhood. It should be possible to turn off any of the agents in order to reduce battery

consumption.

Both global and local agents should consist of a *data acquisition* component which gathers data from the packets or sensors (see Figure 2.2). This data is processed for further analysis. The processed data is stored using a *statistics* component. A *detection* component uses the information stored by the statistics component and analyses the symptoms of attacks. The symptoms of chosen attacks which poses the highest security risks needs to be integrated into the detection component. Results of detections are held in an *alert database*. Nodes are marked as suspicious or malicious there. Finally, a *collaboration* component can be activated when communication with other parts of the system or neighbourhood is necessary.

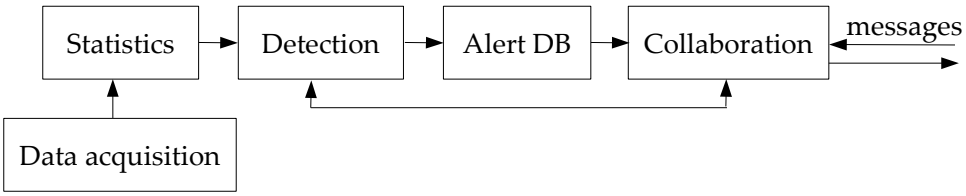


Figure 2.2: IDS component model

2.3 Neighbour-based intrusion detection techniques

2.3.1 Insider attacker detection scheme

The insider attacker detection scheme is presented in [13]. Basic ideas from the paper which were a source of inspiration for this work are summarized in this section. The authors say that the insider attacker detection scheme explores the spatial correlation in neighbourhood activities and contrary to other anomaly detection schemes it requires no prior training. The algorithm is localized which means that information is exchanged only in the limited neighbourhood. The main contribution of the paper, apart from the requirement of no prior training, to this work is that it presents a scheme that is generic. It can monitor many aspects of sensor network behaviour at one time. The way this is accomplished will be described in more detail in following paragraphs.

The basic idea is that neighbouring nodes in some area which are physically close to each other should be dealing with similar network traffic and provide similar values from their sensors. Then, it is possible to look at the set of attributes for some spatially correlated group of nodes and nominate these nodes which differ significantly in some aspect as attackers.

According to the network model from [13], the node x is able to listen to messages coming to its neighbour x_i no matter whether or not it is involved in the communication. The node x creates a model of network behaviour of the node x_i as a q-component attribute vector $f(x_i) = (f_1(x_i), f_2(x_i), \ldots, f_q(x_i))^T$ with each component describing an x_i's activity in one aspect. The component f_j represents actual monitoring results of some behavioural aspect of the node x_i for each and fixed j. For example, it might be a measured value from

the sensor, the number of dropped packets per burst period, packet delivery ratio per some period of time, etc. Behavioural aspects are chosen as appropriate and quantifiable properties which represent statistics that are used to evaluate symptoms of attacks which should be detected by the IDS. The authors assume that for any local area of normal sensor nodes x_i, all $f(x_i)$ follow the same multivariate normal distribution.

The data acquisition component of the node x gathers information from its neighbourhood and creates the set $F(x) = \{(f(x_i) = (f_1(x_i), f_2(x_i), \ldots, f_q(x_i))^T | x_i \in N(x)\}$ of attribute vectors, where $N(x)$ is the set of neighbours of the node x. This set of attributes is broadcast within the neighbourhood $N(x)$ and is taken as a source of statistics for the detection component. This approach eliminates the need of the training phase and storing its results permanently in the database of the detection component of the IDS. In each period, the normal behaviour of a node is defined as the "centre" of the set $F(x)$.

According to [13], the detection component of the insider attacker detection scheme marks malicious intruders after studying the data set $F(x)$. Actually, each node possesses the data set of sets $\{F(x_i)|x_i \in N(x)\}$. This fact is not taken into account in [13] and it is not clear what happens with the data sets $F(x_i)$ which were broadcast. The paper follows with assuming only the data set $F(x)$.

Malicious attackers are considered nodes which are further from the "centre" of the set $F(x)$ than the threshold t_0. Details on a computation of the Mahalanobis distance using Orthogonalized Gnanadesikan-Kettenring estimators can be found in [13] as well as the determination of the threshold t_0. The paper continues with a description of a voting protocol for the final decision about malicious nodes. Different nodes mark the attacker based on information from different neighbourhoods. A node has to be considered an intruder by a majority of its neighbours in order to be excluded from routing tables, reported in the alert database and announced to the base station.

The authors assume that the network on which the insider attacker detection system can be installed is operating a data aggregation protocol. This assumption is important because it ensures that traffic loads in some neighbourhood are correlated. We will try to extend our solution onto networks operating the Collection tree protocol. This will be achieved by employing clustering (see Section 4.4). Furthermore, the implemented IDS that was tested in the discussed paper is programmed to reveal a single attack – fabricated information attack (a malicious node alters the information gained from its sensor). The tests were provided using only synthetic data. We would like to find out what other attacks can be revealed by the proposed scheme (see Chapter 3) and implement an IDS which can detect most of them. Nodes programmed as malicious will be involved in these tests conducted using the TOSSIM simulator. As mentioned in the text above, there is an open question what happens with the data set of sets $\{F(x_i)|x_i \in N(x)\}$ gathered by the collaboration component. Hence, we provide results for several collaborating options and compare them (see Section 4.5).

2.3.2 Group-based intrusion detection scheme

The full description of the group-based intrusion detection can be found in [12]. The detection component of this scheme is very similar to the insider attacker detection scheme. There is even a comparison of simulation results with the insider attacker detection at the end of the paper. The group-based intrusion detection scheme is said to be more precise in marking malicious nodes as attackers and its false alarm rate is lower at the same time. Additionally, it consumes less energy. These satisfying results come from the way nodes are grouped. When the IDS agents are started, preferably after the network start-up, a grouping algorithm is initiated. It is an initial phase after which agents are ready to detect intruders.

The grouping algorithm starts with some nodes sending grouping requests after waiting a random period of time. If a node receives such a grouping request, it joins the group only if it is spatially close enough and its sensed values of the physical environment are similar to the root node of the grouping request. Nodes that are not grouped wait a random amount of time to initiate another grouping request (hence, they become the roots of these new groups). The thresholds limiting the maximal distance between nodes in some group and the group's root node and the maximal deviation of their sensed values have to be defined accordingly to the nature of the network and its application.

Each group is divided into several subgroups which monitor the entire group in turn in order to reduce battery consumption. The data acquisition components of nodes of active subgroups work the same as in the case of the previously mentioned IDS. However, a set of attribute vectors $F(x)$ is not broadcast among the group neither the subgroup. The detection mechanism processes $F(x)$ locally and makes its decision about outliers (malicious nodes) in the group (again, the same principle and mathematics functions are used as in the case of the IDS mentioned in the previous section). No majority voting follows. If an IDS agent finds an attacker, it alerts the entire group with a warning message about the attacker. If there are more such messages, the entire group wakes up and all of the nodes monitor the agitator and the proposed malicious node. If abnormal behaviour is detected in any of these two, the base station is alerted and the actual malicious node is excluded from routing tables.

The authors of the paper include the list of attacks (together with information that needs to be collected for their detection) which are suitable for their scheme. The list can be seen in Table 2.1. However, they implemented their IDS only for the fabricated information attack. Although they tested it on real data, the presence of the attacker was introduced only by noise addition to this data. As far as the list of attacks is concerned, we will point out that there is no need to use the neighbour-based detection technique to reveal packet alteration in Section 3.6. Furthermore, detection of a sinkhole attack based on a high packet receiving rate would provide inaccurate detection results as there may be sinkholes which are not malicious in a network too (see Section 3.4). As mentioned before, the authors implemented their IDS to reveal the fabricated information attack. Hence, they programmed the grouping algorithm to group according to the correlation of values measured by nodes' sensors. If a generic IDS capable of revealing several attacks at a time should be implemented, it is not clear how to group nodes effectively.

Collected information	*Attack*
sensor sensed data	fabricated information attack
packet sending rate	jamming
packet dropping rate	selective forwarding
packet mismatch rate	packet alteration
packet receiving rate	sinkhole attack
packet sending power	hello flood attack

Table 2.1: Attacks whose detection is possible with the neighbour-based detection technique and statistics needed to reveal them

Chapter 3

Attacks

There are different types of attackers and many types of attacks they are able to perform. We focus on active external and internal attackers (insiders) as they are able to run more convenient attacks and the intrusion detection system is deployed to defend against these attacks. An IDS is used to differentiate among trusted nodes and attackers as they might form a legitimate part of the network. In this chapter, the basic description and symptoms of chosen attacks are introduced.

Symptoms of attacks are very important for the study of intrusion detection systems for WSNs. An IDS may determine an internal attacker in the network based on the pre-defined symptoms of known attacks. This work deals with jamming, hello flood, selective forwarding, sinkhole, sybil, packet alteration and fabricated information attacks. The description of these attacks is outlined here based on [14, 15, 16]. This chapter also discusses whether some of the symptoms mentioned for each attack are appropriate to be used for implementation of a neighbour-based IDS as described in the previous chapter (Section 2.3.1).

3.1 Jamming

Jamming is interfering with the radio frequency used by nodes for their communication. It is performed by deliberate transmission of radio signals. It is used to conduct a denial of service attack as nodes cannot communicate at all while a jamming attack is ongoing (Figure 3.1). Nodes consider their communication media to be in use, or they believe some node is transmitting and so they remain in a receiving mode the whole time. A jamming attack is caused by a device which is usually referred to as a *jammer*. It might be a sensor node or some other device able to interfere with the radio frequency of the wireless sensor network. We may distinguish among various types of jamming attacks and jammers. Among the ones that may be the most effective are *constant*, *deceptive*, *random* and *reactive* jammers [17].

A constant jammer continually emits a radio signal without respecting any medium access protocol. In this case, other nodes never find the medium idle. A deceptive jammer uniformly injects regular packets without any gap so other nodes stay in the receiving mode most of the time. A random jammer emits or is asleep to reduce battery consumption. It switches these two states in a random manner. Random jamming may be implemented by both constant and deceptive jammers. A reactive jammer emits only when there is communication on the medium. It is harder to detect than the previous techniques and again it may be implemented by both constant and deceptive jammers.

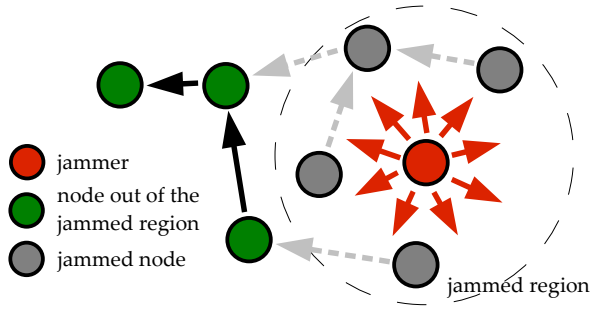


Figure 3.1: Jamming - jammed nodes cannot communicate with their neighbours

Several symptoms might be used to identify jamming. A short overview of these techniques is given in the next paragraph. However, they are not always suitable for every type of jammer. More details on identifying jamming attacks can be found in [17, 18].

A *received signal strength indicator* might be used to detect jamming because the distribution of signal strength is affected by the jammer being active. The basic approach where an average signal strength value is compared to the threshold calculated from the ambient noise level is rather limited. A more convenient technique uses signal strength spectral discrimination. Unfortunately, statistics based on signal strength magnitude are only suitable for identifying constant and deceptive jammers. Signal strength distribution is not affected in such a manner by reactive and random jammers as they alter the sleeping and emitting states which simulate the behaviour of a normal node.

A carrier sensing protocol is used to tell whether a node is allowed to transmit over the media. If a node never finds the media idle, it cannot transmit and may assume that the network is being jammed. A *carrier sensing time* metric is suitable only when the media access protocol of the sensor network tells whether a channel is idle upon a fixed threshold.

The *packet delivery ratio* drops suddenly to almost zero when a node is jammed. In the case of congestion, it does not drop so suddenly and even though the delivery ratio is very low in a congested network, it's not as close to zero as it is in the case of a jammed network. The delivery ratio may tell congestion from jamming and is useful for revealing all the jamming scenarios. However, it is prone to be inaccurate due to battery failure or other network dynamics which may suddenly lead to packet delivery inability. The delivery ratio can be estimated both as received acknowledgements per sent packets on the side of a sender or as the number of packets which passed the cyclic redundancy check per received packets on the receiver's side.

In order to eliminate falsely announced jammings, a combination of the methods described above can be used. Energy exhaustion may lead to a false alarm in the case of the packet delivery ratio symptom. If the packet delivery ratio method is combined with the signal strength consistency check, false positives are reduced. Very low packet delivery ratio and low signal strength imply that a node's neighbour is malfunctioning due to battery depletion. However, when there is a packet delivery ratio close to zero and on the other

hand signal strength is high, a jamming attack is ongoing in the wireless sensor network with highest probability.

The last method, actually the combination of two of them, should provide the most reliable way to reveal jamming. It is even suitable for all of the jamming scenarios defined in this section. Unfortunately, the delivery ratio is meant to be compared with a predefined threshold (a number close to zero). From this point of view, the neighbour-based technique would not be convenient to use (it compares the delivery ratio values among the nodes in the neighbourhood and that is not intended by the packet delivery ratio method). The neighbour-based detection scheme described in the previous chapter is determined to serve as a global agent (see Section 2.2) – it monitors the behaviour of its neighbourhood (not the node running the IDS agent itself). That is why detection based on carrier sensing time cannot also be implemented. Carrier sensing time is a statistic appropriate for monitoring by local agents. We exclude monitoring of the distribution of signal strength because of the same reason.

A *packet sending rate* is easy to monitor and simple assumption would suggest that a node which sends an abnormal amount of packets is one that produces jamming [12]. We can definitely detect deceptive jammers by monitoring the packet sending rates of the neighbouring nodes. However, it depends on the jammer's implementation and the network protocol used whether detection of random and reactive jammers is possible.

3.2 Hello flood attack

Routing protocols usually prefer the shortest or the most reliable path to the base station. Hello packets (sometimes also referred to as advertisements or beacons) are sent out by a new node in the network in order to inform other nodes that they can possibly route their messages via the new node. If a malicious node possesses a long-range antenna, it can broadcast hello packets claiming good connection to the base station. These hello packets will be received by the nodes which cannot reach the adversary back as they do not have such a strong antenna (Figure 3.2). The affected part of the network becomes paralysed as no messages are routed out of it.

Nodes which are close to the attacker may notice that *received signal strength indicator* values of messages delivered from the attacker are abnormally high. This detection method can be implemented using the neighbour-based detection technique. Nodes will keep statistics of average signal strengths of received messages from their neighbours and compare them with the averaged value of these statistics. A node having its average signal strength significantly higher will be announced as the hello flood attacker.

3.3 Selective forwarding

A compromised node (an attacker) drops packets instead of forwarding them further in a multi-hop routing system in case of a selective forwarding attack (Figure 3.3). An attacker may drop all of the incoming packets (also denoted as black hole attack) or selectively drop

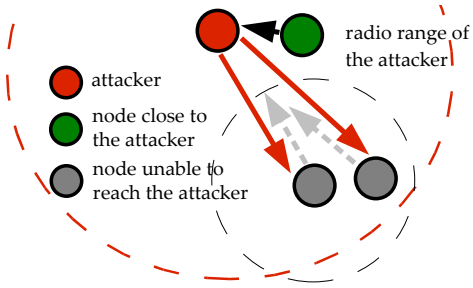


Figure 3.2: Hello flood - nodes which cannot reach the attacker will choose it for their parent most probably, hence paralysing themselves

only specific packets (coming from a specific source, having a certain destination, containing certain payload data, etc.). In the second case, it is harder to detect and several statistics have to be stored by an IDS to check.

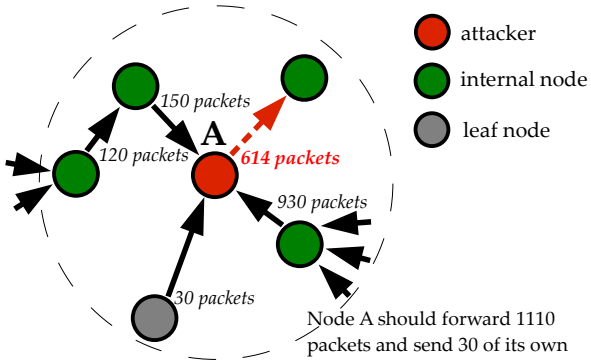


Figure 3.3: Selective forwarding

A high *packet dropping rate* can be used to identify nodes that conduct selective forwarding [19]. It is estimated as the ratio between the number of sent packets and the number of received packets over a period of time. This ratio may be kept as an overall number of transmitted packets or just for packets coming from a specific source, having a certain destination, etc.

The approach from [20] does not only take into consideration packets being dropped by a malicious node. At the same time, the intrusion detection system checks whether a packet is sent to a legitimate neighbour of the monitored node. Otherwise, it assumes that the packet was forwarded by a malicious node to a mismatched location on purpose. In order to make this technique work, the form of a hello packet (used to establish routing tables when a new node is added to the network) has to be changed so each node is able to derive its 2-hop neighbours from it. This requirement makes this mechanism harder to implement than in the case of packet dropping rate monitoring. The protocol for finding out node's 2-hop neighbours would mean another communicational and computational overhead for tiny

nodes. The described neighbour-based intrusion detection is supposed to be a lightweight agent and is not considered to provide such a functionality.

On the other hand, the packet dropping rate is an appropriate metric that can be used to monitor the network behaviour of a node's neighbourhood in the neighbour-based IDS. Nodes that are in spatial correlation (according to the insider attacker detection described in the previous chapter) should be dealing with a similar traffic load and network dynamics. The packet dropping rate should be similar as well and if a node drops packets in extreme numbers, it is probably malicious.

3.4 Sinkhole attack

A sinkhole node is one where most of the traffic is reflected to (Figure 3.4). According to a routing protocol, it is the one claiming extremely good connection to the base station in its neighbourhood. An attacker tries to create a sinkhole node from the one that is captured by them. Afterwards, more serious attacks can be run using this node. Depending on which routing algorithm is used, an attacker tries to fake routing protocol's metrics which define the best path to the gateway so most of its neighbours, preferably all, set the captured node as their parent node. An IDS may identify nodes which claim a suspiciously high-quality connection to the gateway and are the only such nodes in their neighbourhood. This technique cannot identify a sinkhole which is started at the beginning of a network's existence because the sinkhole's neighbours will claim good connection via the sinkhole node as well. The extreme difference in the apparent quality of the connection will not be noticeable.

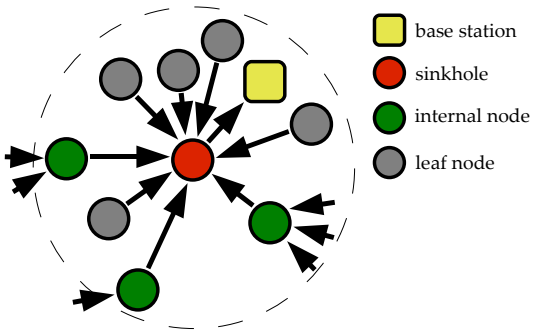


Figure 3.4: Sinkhole – all the traffic from this network region is routed via the sinkhole

If the *packet receiving rate* of some node is extremely high, it might be suspected of being an attacker [12]. However, this symptom does not differentiate natural sinkholes which may exist depending on network topology. Furthermore, the packet receiving rate of surrounding nodes will become high as well because they will gain very good connection to the base station via the sinkhole node. This approach is considered very limited and not suitable for intrusion detection techniques.

Although a malicious node may claim that its connection to the gateway is better than it actually is, nodes in its neighbourhood do not have to change their parents straight away

(this depends on a routing protocol, e.g. the CTP in TinyOS is designed this way). An attacker has to downgrade the quality of other nodes' connections in this case. A malicious node may spoof fake root update packets impersonating its neighbours. The authors of [21] suggest that this form of sinkhole attack may be revealed by an IDS which observes whether senders of root update packets are in the neighbourhood of the node running the IDS. If not or if they are even sent by the node running the IDS itself (possible when the packet's header is altered), some node is running the sinkhole attack in the network.

This technique effectively reveals attackers that try to downgrade the quality of other nodes' connections. If some node finds out that another node spoofs packets, it should alert other nodes about this immediately. It does not matter what the common behaviour of the neighbourhood is. From this point of view, this technique is not suitable for the neighbour-based intrusion detection. Unfortunately, no other technique that could be used is known.

3.5 Sybil attack

A sybil node is one that is claiming multiple identities. An attacker that owns these identities may take advantage in voting protocols or create routing paths for their own benefits. Communication with sybil nodes may be *direct* or *indirect*. In direct communication, the compromised node communicates with the other nodes in charge of all of its identities. In indirect communication, the sybil node claims that it communicates with nodes that actually do not exist. Sybil identities might be *fabricated* or *stolen*. A sybil attack can be performed *simultaneously* or *non-simultaneously* depending on whether the sybil node uses its identities at once or over time.

A sybil node may be revealed by *location testing* which is based on the principle that some number of cooperating nodes are able to estimate another node's location based on some measurements. If they find out that two nodes are located at the same position, a sybil attack is most likely being conducted by an attacker.

The technique using the *received signal strength indicator* is described in [22, 23]. Three cooperating nodes measure the signal strength upon receiving a message (Figure 3.5). After exchanging obtained values, the ratio is measured from them and stored in a database of neighbours. A unique ratio value determines the unique position of a node. A similar approach is published in [24] too. Location testing is based on measuring the *time difference of arrival* of packets among cooperating nodes instead of the received signal strength.

Both location testing methods need a group of nodes to exchange some information. This information is used for a simple computation whose result is a determination of the location of the node which they exchanged the information about. Each node has a table of such locations of the nodes in its neighbourhood. Both techniques do not look for a property of network behaviour which should not vary noticeably for a node in some group of neighbouring nodes. This is the reason why it cannot be used for the studied neighbour-based IDS. Unfortunately, we do not know any other technique that could be implemented.

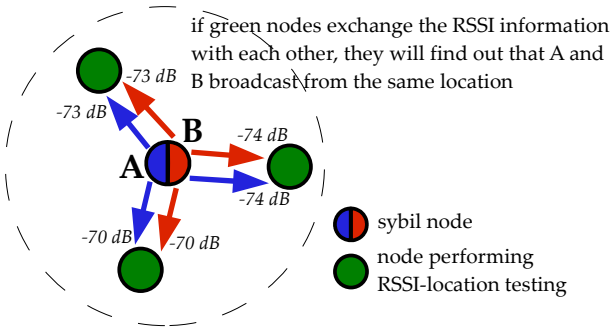


Figure 3.5: Received signal strength indicator location testing of sybil attack

3.6 Packet alteration

An attacker might be interested in spoofing or altering packets of other nodes (Figure 3.6) in order to misuse a routing algorithm, have an advantage in voting protocols or change measured values sent by sensor nodes to the base station.

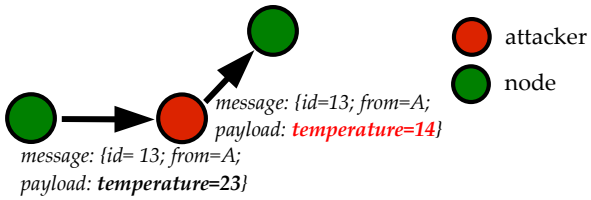


Figure 3.6: Packet alteration

Monitoring of spoofed packets can be provided in a similar way as is described in this chapter in the section dedicated to sinkhole attacks and revealing fake beacons (Section 3.4). The basic assumption is that a node should be able to hear only packets that have originated in its neighbourhood. If they have originated elsewhere, they are spoofed packets.

In order to detect alteration of data, an IDS has to store overheard packets in the buffer, wait until appropriate nodes forward them and compare whether the payloads are the same for the forwarded packets and the packets stored in the buffer. The presence of an attacker that alters or spoofs packets is noticed immediately. There is no need to use techniques such as the neighbour-based detection technique to find out if it is common to deliberately alter packets in the node's neighbourhood. Deliberate alteration should always be considered as an attack.

3.7 Fabricated information attack

A malicious node might send fallacious measured values which would not reflect the reality of its surroundings to the base station. There is an assumption that these values provided

by nodes from a close neighbourhood should usually vary just slightly (Figure 3.7). When values in node's surroundings are compared and an IDS finds out that the node provides extremely different results, it is suspected of being captured by an attacker. Simulations of both insider attacker and group-based intrusion detection schemes were run to monitor usage of this statistic. The description and results of the simulations can be found in [13, 12] respectively.

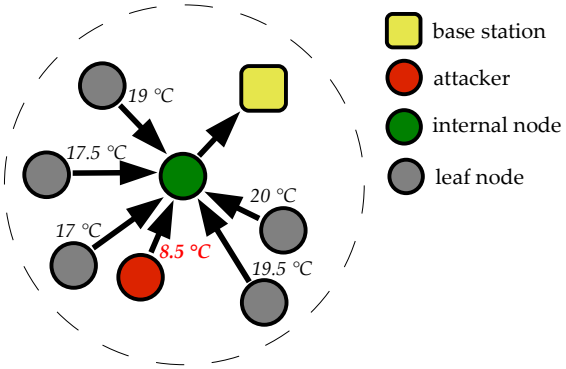


Figure 3.7: Fabricated information attack – values from trusted nodes vary just slightly

3.8 Neighbour-based detection of the attacks

Several symptoms that can be used by the detection component of an IDS to identify jamming, hello flood, selective forwarding, sinkhole, sybil, packet alteration and fabricated information attacks were presented in the previous sections. Unfortunately, not all of them are suitable for the neighbour-based detection technique. The reasons for this assumption were given above.

In order to reveal the jamming attack, the packet sending rate is an appropriate statistic for the neighbour-based IDS discussed in this work. The hello flood attack can be detected by monitoring the received signal strength indicator readings of nodes in the neighbourhood of the node running the IDS agent. The packet dropping rate may identify a malicious adversary that selectively drops packets instead of forwarding them according to the routing protocol. Regrettably, no appropriate statistics for neighbour-based detection were found for disclosing the sinkhole or sybil attack. Although the packet alteration attack is easy to recognize, the means by which this is done is not suitable for this work. Finally, the attack based on providing fallacious measured values has already been implemented by the authors of [13, 12]. The implementation of the appropriate statistics depicted here is described in detail in the next chapter where our neighbour-based IDS is presented.

Chapter 4

IDS design and implementation

This chapter describes the design of the intrusion detection system that has been implemented for the purpose of this research. The IDS uses the neighbour-based intrusion detection technique (as described in Chapter 2) and it serves as a global agent which means that it monitors its neighbouring nodes and does not monitor the behaviour of the node running the IDS agent itself. The neighbour-based detection technique as described in previous chapters is an anomaly detection technique and the normal behaviour (contrary to anomaly) is defined as actual common behaviour of the neighbourhood. It is assumed that non-malicious (referred to as trusted or fair) nodes always prevail in a neighbourhood over the malicious ones. The decomposition and design of the IDS is mostly based on [11] (see Chapter 2). The IDS comes in two modifications. The first one is built to operate over the Collection tree protocol [25, 26] and the other one to be run on networks with an aggregation protocol.

This chapter starts with the depiction of the IDS component model and the description of the technique used to process promiscuously overheard communication of the neighbouring nodes. It describes how each of the statistics is measured and what other mechanisms are needed to do so. A *clustering* technique is introduced as the consequence of unsatisfying detection results of selective forwarding and jamming in the CTP operating networks. Finally, implementation of *collaboration* among neighbouring nodes is described.

4.1 IDS component model

The component model is adopted from [11] and its depiction is summarized in Section 2.2. The IDS is composed of *data acquisition*, *statistics*, *detection*, *collaboration* and *alert database* components (Figure 4.1).

The data acquisition component is adopted from [27] and its detailed description can be found there. It introduces `AMTap` interface which provides three events: `onReceive(message_t* msg, ...)`, `onSnoop(message_t* msg, ...)`, `onSend(message_t* msg, ...)`. The events are signalled on receiving a message (in case its destination is this node), on snooping a message (this occurs if the node can hear the message, however, the message is not destined to the node itself) and on sending a message by the node. It is possible to gain information about communication involving the node itself or communication of its neighbours by writing handlers for these events.

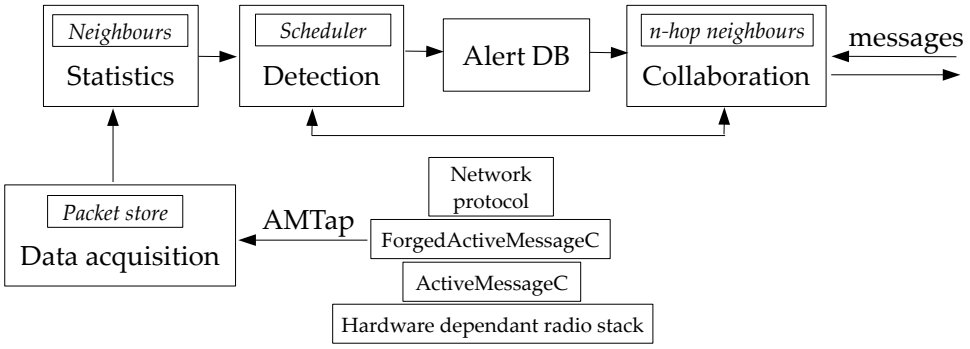


Figure 4.1: IDS component model

The database of the node's neighbours stores the information gained by the data acquisition component over some predefined period of time τ (this would usually be the detection period). It is held in the statistics component. Its size is limited due to space and computation efficiency.

We denote the node running the IDS agent as a_i ($i = \overline{1, s}$). The node a_i monitors its direct neighbours $N(a_i) = \{n_{i1}, ..., n_{im_i}\}$. The number s denotes the total number of nodes in a network and m_i denotes the number of neighbours of the node a_i. Each record about the neighbour n_{ij} ($j = \overline{1, m_i}$) in the database of a_i has these attributes (active message address serves as a unique identification):

- *active message address*
- *average received signal strength* (SS)

  $SS(n_{ij})$ is the average received signal strength of the node n_{ij} received at a_i per τ
- *packet receiving rate* (PRR)

  $PRR(n_{ij})$ is the number of packets (per τ) which were snooped or sent by a_i and destinated to n_{ij}
- *packet sending rate* (PSR)

  $PSR(n_{ij})$ is the number of packets (per τ) which were snooped or received by a_i and originated in n_{ij}
- *packet forwarding rate* (PFR)

  $PFR(n_{ij})$ is the number of packets (per τ) which were snooped or sent by a_i and destinated to n_{ij} and a_i observed n_{ij} to forward them
- *packet dropping rate* (PDR)

  $PDR(n_{ij})$ is the number of packets (per τ) which were snooped or sent by a_i and destinated to n_{ij} and a_i did not observe n_{ij} to forward them within the predefined time period φ

- *the number of packets announcing congestion* (PAC)

  $PAC(n_{ij})$ is the number of packets (per τ) which were snooped or received by a_i and originated in n_{ij} and their congestion notification bits were set to 1

Although most of this information is available from the message which comes as a parameter of the `AMTap` events, we need to place a *packet store* unit into the data acquisition component in order to keep track of which packets are being forwarded or dropped. The main functionality of the packet store unit is encapsulated as a database of minimal unique identifications of messages that have been snooped or sent by the node running the IDS and have not been forwarded by their destination nodes yet. A message is erased from the packet store when it has been forwarded. $PFR(n_{ij})$ is increased. In case the packet store is full, the oldest message is erased. $PDR(n_{ij})$ is increased at this time. The situation when the destination node has not forwarded the message after the predefined time period φ has elapsed is considered a packet dropping event as well.

The data acquisition component is running continuously as a daemon process on each node running the IDS. It acquires information that is stored in the statistics component. The detection component accesses this information later there. The detection component provides means to run the detection of malicious nodes which is started periodically. We denote this period as τ. Optionally, the detection component may use the collaboration component so the nodes share their neighbour databases (this would occur just before running the detection itself) or their detection results.

Detected malicious nodes are supposed to be announced to the neighbouring nodes and the base station and stored in the alert database. An appropriate response action should be initiated (e.g. excluding malicious nodes from the routing tables). However, this work does not deal with response mechanisms. Detected malicious nodes are only announced via the alert database. The alert database is implemented to output statements in the console when running the TOSSIM simulation. The alert statement would describe malicious nodes and ongoing attacks.

4.2 Ambiguous collisions and low gain mode monitoring

The problem that is caused by ambiguous collisions in watchdog monitoring IDSs is well-described in [27] and the solution presented there is also used in this work. The problem is especially significant when detecting selective forwarding. Ambiguous collisions may occur in several forms. The one that can be prevented is depicted in Figure 4.2. The node *D* is a watchdog for communication from the node *A* to the node *C* which is mediated via the node *B*. *D* needs to know whether *B* forwards the message to *C*. Unfortunately, just before *B* sends the message to *C*, the node *E* (which is out of the radio range of *B*) sends a message to the node *F*. Eventually, *D* is not able to receive any of the two messages and may consider that *B* dropped the message.

Although it is not possible to eliminate all of the ambiguous collisions, the number of their occurrence can be lowered by suppressing weak signals. This is achieved by lowering

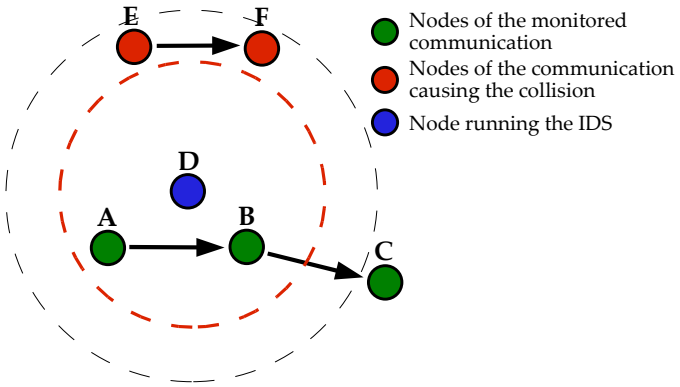


Figure 4.2: Ambiguous collision

the sensitivity of the receiver on the CC2420 chip. The implementation of this functionality for the TOSSIM simulator is adopted from [27]. Lowering receiver's sensitivity is depicted in Figure 4.2 as the red inner circle. The monitoring with suppression of weak signals enabled will be referred to as *low gain mode monitoring*. On the other hand, *high gain mode monitoring* will denote no change to the sensitivity of the receiver.

4.3 Detections

After detection period τ has elapsed since the last detection, the detection is started. At this time, statistics (the set of records about the neighbours $\{n_{i1}, ..., n_{im_i}\}$ from the neighbours database of the node a_i) are obtained from the statistics component. The statistics may be optionally exchanged in the neighbourhood (see Section 4.5). Once they are processed and stored for detection, the statistics component's database is cleared and data acquisition for the next detection round can begin. The packets waiting for forwarding in the packet store are not touched at all and wait for processing in the next detection round.

The detection component computes centre values as average SS, PSR and PRR for all nodes but the one with the maximum and the one with the minimum values (denoted as AVG^* as opposed to AVG for general average function). These average values form components of the *centre values vector* (q-component vector as defined in Section 2.3.1). Eventually, the *detection engine* is called and the statistics together with the centre values vector are passed to it for evaluation. Depending on the IDS modification, running over the CTP or an aggregation protocol, a detection rule and a threshold value (δ) for each attack are chosen. When an aggregation protocol is used, the centre values vector contains all needed information. For the case of the CTP, another average values are calculated using the *clustering* unit (called by the detection component) for each cluster.

In order to be able to evaluate efficiency of an implementation of some detection technique, terms *false positives* and *false negatives* will be used. The approach from [28] is adopted and detailed definitions of these terms can be found there. The term false posi-

tive will refer to any alert warning of a node which is not malicious, hence it is accused falsely. The number of false positives will be the number of all such alert warnings for a detection period (note that there can be several alert warnings announcing the same node). On the other hand, if a malicious node is not announced by the IDS agent monitoring it, the situation is considered false negative because the presence of the attacker was evaluated falsely negative. Again, several false negatives may refer to the same node. Furthermore, the number of *nodes falsely accused* as malicious is always lower or equal to the number of false positives and expresses how many different nodes were announced by falsely positive alerts. When the IDS is tested, the number of attackers involved in the network is known, hence the number of attackers that were not revealed by at least one IDS agent can be calculated. It is equal to the difference of the number of attackers in the network and the number of attackers revealed at least by one IDS agent. The number of *alerts about malicious nodes* will denote only to correct alert warnings when the IDS marks a malicious node as malicious. Hence, false positives are not counted in this number.

Different detection rules and thresholds are used when running the IDS in networks with an aggregation protocol and the CTP. First of all, network dynamics as PSR or PRR are very correlated in case of an aggregation protocol, however, they differ significantly for nodes situated at different levels of the CTP tree or for nodes whose number of children vary to a considerable degree. Secondly, PDR and PFR can be monitored only in the case of the CTP. In order to be able to monitor these properties in networks with an aggregation protocol, the IDS would have to know the way messages are aggregated. That is not intended in this work.

4.3.1 Detections in networks with an aggregation protocol

In case of the IDS for the aggregation protocol operating networks, a rule is evaluated for each neighbour n_{ij} in the set of statistics gathered by a_i. If n_{ij} satisfies the rule, an alert is signalled using the alert database. The rules are:

Rule 4.1: Hello flood attack

$$SS(n_{ij}) - AVG^*(SS(n_{i1}), ..., SS(n_{im_i})) > \delta_{SS}$$

Rule 4.2: Deceptive and random jamming

$$PSR(n_{ij}) - AVG^*(PSR(n_{i1}), ..., PSR(n_{im_i})) > \delta_{PSR}$$

Rule 4.3: Selective forwarding attack

$$AVG^*(PSR(n_{i1}), ..., PSR(n_{im_i})) - PSR(n_{ij}) > \delta_{PSR-MIN}$$

A hello flooder is the node which SS is significantly higher than it is common among its neighbours. We are able to detect a jammer (deceptive or random) if it emits more messages than it is common in its neighbourhood. In the case of the selective forwarding attack, a node which leaves out some of the burst periods is believed to be a dropper. This can be noticed when the PSR of some node is lower than its neighbours' $PSRs$.

The thresholds values δ_{SS}, δ_{PSR} and $\delta_{PSR-MIN}$ are identified empirically by the network administrator so the numbers of false positives and false negatives are bearable (see Chapter 5 for more information).

4.3.2 Detections in networks with the Collection tree protocol

In case of having the IDS installed on nodes of a network using the CTP, detection of the hello flood is done the same way as in networks with an aggregation protocol, Rule 4.1.

We are able to detect deceptive and random jamming. The rule for detection of random jamming will be defined later in this text (Section 4.4.2), the deceptive jamming rule is the same as the one from the previous section, Rule 4.2.

The rule for detection of selective forwarding can be based on comparing the percentage of the dropped packets. The percentage of the dropped packets will be denoted as *packet dropping ratio*. It is defined as the ratio of the PDR and receiving rate of packets that were supposed to be forwarded (denoted as PFR^*, computed as $PDR + PFR$).

Rule 4.4: Selective forwarding attack

$$\frac{PDR(n_{ij})}{PFR^*(n_{ij})} - AVG(\frac{PDR(n_{i1})}{PFR^*(n_{i1})}, ..., \frac{PDR(n_{im_i})}{PFR^*(n_{im_i})}) > \delta_{PDR}$$

However, there are several problems to face. Even though the rules' definitions are reasonable and express the nature of the attacks, finding threshold values so the trade-off of false positives and negatives is bearable might be difficult. Several reasons can be provided to explain this and a change of the way the rules are applied may make results better.

First of all, for the case of selective forwarding, there are ambiguous collisions which falsely increase $PDRs$ for internal nodes of the CTP tree. Even though we decrease the number of their occurrence by low gain mode monitoring, they cannot be omitted totally. It should be noted that ambiguous collisions are not an issue for networks with an aggregation protocol because the data traffic is much lower there.

Secondly, and this applies to both jamming and selective forwarding, neighbour-based detection is based on the principle that neighbouring nodes deal with similar sensor readings, network dynamics, etc. However, this is not true for several metrics describing network flow which a node deals with in the CTP. $PRRs$ differ significantly for nodes spatially close to each other if they lie at different levels of the CTP tree or have a different number of children (see Figure 4.3). Furthermore, PS, PFR and hence PDR are implied by the value of PRR. This assumption means that we may achieve better results if we partition neighbours

into several groups according to their PRR and detect selective forwarding in these groups. We refer to this method as to *clustering* and its detailed description can be found in the next section.

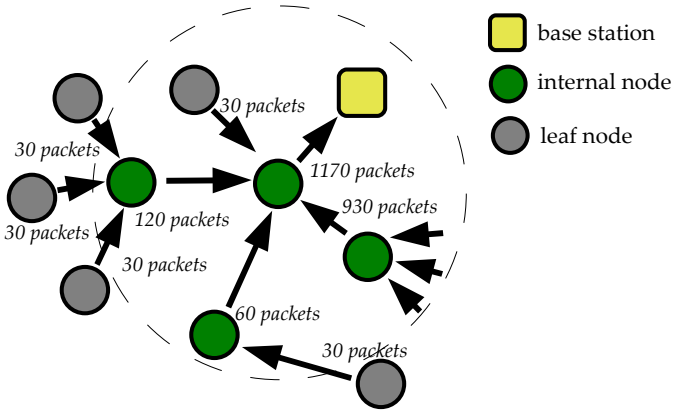


Figure 4.3: $PRRs$ may differ for nodes in a close neighbourhood

Finally, it is hard to keep track of the actual value of the PRR for some node. The known value is usually only partial. However, knowledge of the PRR is crucial for application of the clustering method and, as it will be shown, for revealing random and reactive jamming too. It can be seen in Figure 4.4 that only the subgroup of nodes monitoring the node C is able to hear the most significant amount of the traffic that is received by C and no node is able to know the actual value of its PRR (even if no collision occurs). Collaboration of nodes can be employed in order to refine the information about the PRR. Implementation of nodes collaboration in the proposed IDS is discussed later in this chapter.

4.4 Clustering

The clustering method is used for detection of both selective forwarding and jamming attacks. It customizes the neighbour-based detection technique for the CTP for which different traffic loads are typical for nodes positioned at different levels of the tree.

4.4.1 Selective forwarding

The detection of selective forwarding can be implemented using Rule 4.4 defined in the previous section. A naive approach is to apply the rule for every node in the set of the statistics. If the rule is passed by the given inputs, the node is announced as malicious. However, due to ambiguous collisions, nodes are considered to drop packets even if this is not the case and hence the number of false positives might be high.

The naive method described above was implemented and simulated on a network of 224 nodes set to low gain mode monitoring with the threshold of -88 dB. A node in this configuration had approximately 28 neighbours, however, only the subset of 20 of these neighbours

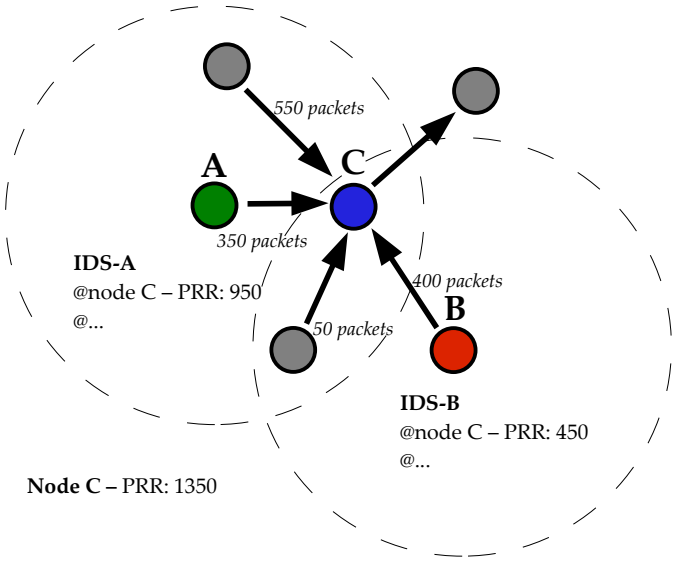


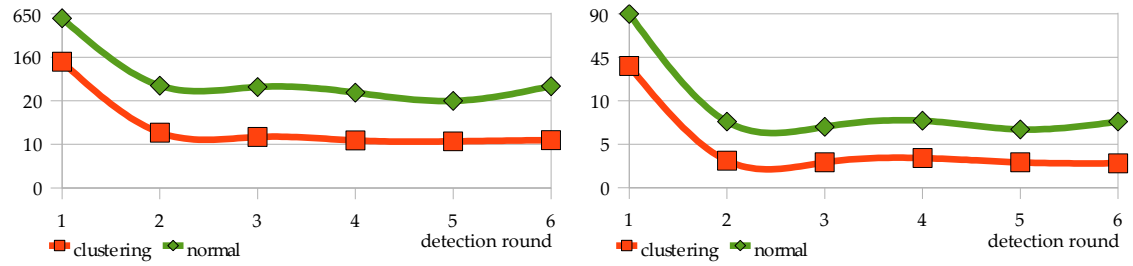
Figure 4.4: No IDS agent knows the actual PRR of the node C

with the highest $PRRs$ was monitored due to the limited memory space and computational power of a node. Nodes sensed the temperature using their sensors and sent the measured value to the base station every 16 seconds (it is the shortest sending period tested for the CTP by its authors in [25]). No attackers were involved in the network. Detections were held every 304 seconds (τ) and tested the selective forwarding attack having δ_{PDR} set at 10 %. This means that if the common dropping ratio is d % in a node's neighbourhood, the node is allowed to drop up to $d + 10$ % of packets.

Six detection rounds were evaluated in 10 simulations. The first detection round always provided significantly worse results than the other detection rounds due to the inconsistency of a network flow which is caused by the creation and stabilization of the CTP tree. If only the results of the other five detection rounds are averaged, the detection technique outputs 22.4 false positives accusing 7.32 different nodes as malicious per detection round. The averaged results of the 10 test rounds can be seen in Figure 4.5 and are referred to as normal detection.

The results are quite unsatisfying. In order to decrease the number of false positives, δ_{PDR} could be increased. However, this would have negative effect on the number of false negatives (especially in the case of attackers that drop only a minor part of the traffic routed through them).

We should take into account that for the nodes lying at different levels of the CTP tree, $PRRs$ differ significantly even if they are spatially correlated. Furthermore, PSR, PFR and hence PDR are implied by the value of PRR. After analysing traffic flowing via nodes that are falsely announced as malicious and the traffic in their neighbourhood, we may identify that there are usually three groups of nodes in some neighbourhood. There is a group of nodes receiving the highest amount of traffic (approximately 1300 packets per node) hav-



(a) false positives

(b) nodes falsely accused as malicious

Figure 4.5: Normal and clustering detection of selective forwarding in a CTP network comparison

ing a lot of messages forwarded and some being dropped (we refer to packet dropping according to the IDS observation which does not necessarily mean the actual dropping of the packet only that the IDS did not observe the packet being forwarded). Then, there is a group of nodes which receives just a fraction of the number of messages (approximately 280 packets per node) received by the first group. These nodes do not tend to drop so many packets as the previous group. The last group could be defined as nodes that are not forwarders or forward just a very small amount of messages (approximately 100 packets per node). It is important to note that if the IDS notices such a node to drop just a couple of packets, the percentage of the dropped packets goes quite high.

We conclude that it might be convenient to apply the detection rule and compute the centre values over the first and the second group separately (as the nodes within each group are more correlated) instead of doing so over the whole set of neighbours. We do not need to evaluate nodes from the third group as they are not supposed to forward significant amounts of traffic so they are not of any interest to the attacker. We implement this idea using the k-means clustering method [29] with k set to 3, hence we refer to it as clustering. For the case of selective forwarding, clustering is done according to the nodes' $PRRs$.

We define a_i's clusters as the subsets C_{ik} ($k = \overline{1,3}$) of the set of neighbours $N(a_i) = \{n_{i1}, ..., n_{im_i}\}$. At the same time, $N(a_i) = \bigcup C_{ik}$ and $\emptyset = \bigcap C_{ik}$. The size of the cluster C_{ik} is referred to as $m_{C_{ik}}$ and the members of C_{ik} can be addressed by $C_{ik}(n)$ where $n = \overline{1, m_{C_{ik}}}$. Then, the detection rule for selective forwarding is as follows:

Rule 4.5: Selective forwarding attack (clustering)

$$\frac{PDR(n_{ij})}{PFR^*(n_{ij})} - AVG(\frac{PDR(C_{ik}(1))}{PFR^*(C_{ik}(1))}, ..., \frac{PDR(C_{ik}(m_{C_{ik}}))}{PFR^*(C_{ik}(m_{C_{ik}}))}) > \delta_{PDR} \wedge n_{ij} \in C_{ik} \wedge k > 1$$

The sets of statistics of all the detection rounds gained from the simulations described in the paragraphs above are used to obtain results of detections that employ the described clustering method (different results would be gained if the simulations were run again). The average number of nodes in the first and the second cluster was 4. The cluster containing

the nodes which forward the least part of the traffic had an approximate size of 10 nodes. As predicted, we achieved the lowering of the number of false positives (from 22.4 to 11.4) and the number of nodes falsely announced as malicious is lower (3.02 nodes opposed to 7.32) when clustering was used for the detection (again, averaged values are computed from all the detection rounds but the first one). The comparison of results can be seen in Figure 4.4. Furthermore, the results did not fluctuate and the detection with clustering achieved better result in every single detection round of every simulation.

Detection using clustering becomes a part of a default configuration of the proposed IDS when installed on networks operating the CTP. It is possible to turn off this feature when needed. There is no need to use clustering for the detection in networks with an aggregation protocol because the nodes in these networks should receive and send approximately the same amount of traffic.

4.4.2 Jamming

The motivation for searching for more advanced techniques for revealing jamming is the same as in the previous section for the case of selective forwarding. Nodes at different levels of the CTP tree send different amounts of traffic even though they lie close to each other. If we consider the network from the simulations described in the previous section, it is common that the $PSRs$ of the 20 neighbouring nodes vary from 1,000 to 20 and for some nodes it even goes up to 3,000 or 4,000 (still having leaf nodes with 20 packets sent per the detection period in the neighbourhood).

4.4.2.1 Deceptive jamming

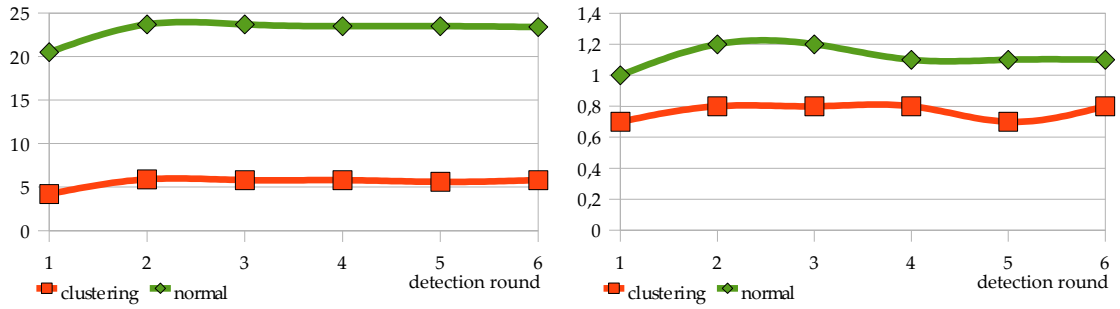
Firstly, we consider deceptive jamming where the jammer is sending packets one after another without considering the medium access (MAC) protocol. In this scenario, nodes that are jammed are the ones within the radio range of the jammer. However, their effort to reach the idle medium grows exponentially and hence they may paralyse their neighbours as well. Furthermore, they are not able to hear any other messages from other nodes because of the jamming. If the average value of all the $PSRs$ in the set of the statistics gathered by the IDS were computed, the number of false positives would be high in the case there were no jammers in the network (nodes which forward 4,000 packets per detection period would be announced as the jammers). Clustering will be used to differentiate the group of nodes which detection will be held on. This will be the union of the two clusters of the nodes having the highest $PSRs$. The clusters (created according to the nodes' $PSRs$) are joined together because their average sizes are quite small (2.5 and 2.7 nodes). The other cluster of the nodes whose $PSRs$ are low will not go through the detection (approximately 13 nodes in this cluster). The detection rule for the deceptive jamming, Rule 4.2 stated in Section 4.3.2., can be reformulated as stated below:

Rule 4.6: Deceptive jamming attack (clustering)

$$PSR(n_{ij}) - AVG(PSR(C_{i1}(1)), ..., PSR(C_{i1}(m_{C_{i1}})), PSR(C_{i2}(1)), ..., PSR(C_{i2}(m_{C_{i2}}))) > \delta_{PSR} \wedge n_{ij} \in C_{i1} \cup C_{i2}$$

10 simulation tests were run to identify the number of false positives for the case of revealing deceptive jamming using clustering disabled and clustering enabled detections. The configuration of the network and the IDS was the same as in the previous section. δ_{PSR} was set to 2,000 packets. It should be noted that a deceptive jammer in used simulation configuration produces up to 60,000 messages per detection round.

The achieved results (the average of all detection rounds but the first one) were 5.78 false positives in contrast to 23.56 (when clustering was not used). Approximately less than one node (0.78) per test round was announced as malicious when clustering was used (it was 1.14 in the other case). The results per detection round can be seen in Figure 4.6.



(a) false positives

(b) nodes falsely accused as malicious

Figure 4.6: Normal and clustering detection of deceptive jamming in a CTP network comparison

4.4.2.2 Random jamming

We would like to be able to detect another type of jammer as well. It is said to be more convenient as it injects packets into the traffic only at the time when other nodes are transmitting - a reactive jammer. Random jamming is quite similar, a jammer transmits randomly over time (not obeying MAC protocol) and so it happens to collide at least with some of the traffic. We focus on random jamming in our simulations. However, the method proposed here should be able to reveal reactive jamming too.

Although a random jammer does not create a heavy amount of redundant traffic, we may assume that the messages which are sent by this jammer are not the forwarded messages based on some messages received by the attacker from its CTP children. Hence, if a percentage ratio of received to sent messages is computed, it should be significantly lower in case of a malicious node. A new rule is defined for revealing random jamming based on so called *received to sent ratio*:

Rule 4.7: Random jamming

$$\frac{PRR(n_{ij})}{PSR(n_{ij})} - AVG(\frac{PRR(n_{i1})}{PSR(n_{i1})}, ..., \frac{PRR(n_{im_i})}{PSR(n_{im_i})}) > \delta_{\frac{PRR}{PSR}}$$

Rule 4.7 could be used for detecting deceptive jamming too. However, as it will be shown later, this rule introduces much more false positives than the one defined for revealing deceptive jammers (hence, it is better to have a separate and very effective rule (Rule 4.6) for this case).

When revealing random jamming, clustering is employed again. We search for nodes producing some amount of messages which causes collisions. Hence, we cluster the set of the statistics according to the $PSRs$ of its members and exclude the nodes sending the least amount of packets from detection (approximately 13 nodes). Furthermore, the two clusters with the neighbours with the highest $PSRs$ are joined together (the sizes of these clusters are 2.5 and 2.7 nodes on average). Otherwise separate clusters would be too small and an attacker having its received to sent ratio low would influence the common received to sent ratio heavily. In such a case, the attacker might be evaluated falsely negative because it would be common to have low received to sent ratio in its cluster. Simulations were performed again the same way and in the same configuration as described in Section 4.4.1 which was dedicated to the selective forwarding attack. $\delta_{\frac{PRR}{PSR}}$ was set to 65 %. No attacker was involved in the network. Elimination of the false positives and nodes falsely accused as malicious by using clustering is confirmed again as shown in Figure 4.7. Rule 4.7 is adapted to clustering as follows:

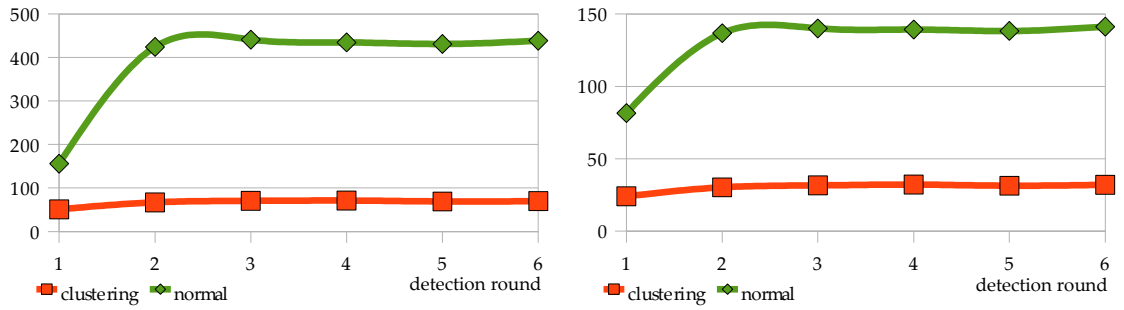
Rule 4.8: Random jamming attack (clustering)

$$\frac{PRR(n_{ij})}{PSR(n_{ij})} - AVG(\frac{PRR(C_{i1}(1))}{PSR(C_{i1}(1))}, ..., \frac{PRR(C_{i1}(m_{C_{i1}}))}{PSR(C_{i1}(m_{C_{i1}}))}, \frac{PRR(C_{i2}(1))}{PSR(C_{i2}(1))}, ..., \frac{PRR(C_{i2}(m_{C_{i2}}))}{PSR(C_{i2}(m_{C_{i2}}))}) > \delta_{\frac{PRR}{PSR}}$$
$$\wedge\, n_{ij} \in C_{i1} \cup C_{i2}$$

As mentioned before, the IDS does not know the exact $PRRs$ about its neighbours. Unfortunately, the known values are often not even close to it. Knowledge of the $PRRs$ of node's neighbours is a precondition for the application of the method described in the previous paragraph. The collaboration component is described in the next section. It is implemented in order to gain more accurate information of the node's neighbourhood.

4.5 Collaboration

Collaboration is used for the information exchange between the IDS agents running on different nodes. It is implemented in order to extend the number of neighbours passed for



(a) false positives

(b) nodes falsely accused as malicious

Figure 4.7: Normal and clustering detection of random jamming in a CTP network comparison

detection. This is needed when there is not enough nodes monitored by a single IDS agent (the network density is low) and it helps to reveal node's other neighbours which cannot be monitored directly as they are two hops away from it. Hence, it collaborates with its 1-hop neighbours – neighbouring nodes exchange the statistics they gathered. Alternatively, this can be done with nodes that are two or more hops away from the node running the IDS agent. This means of collaboration may be used for detection in sparse networks operating the CTP as well as an aggregation protocol. However, collaboration might be needed even in dense networks, as mentioned in the previous section, for example when the IDS needs to know the most exact information about the $PRRs$ of its neighbouring nodes. In this case, it is implemented in order to refine actual statistics of the IDS agent.

Collaboration has both its pros and cons. If we consider a network with attackers, information received from neighbours might be influenced by their presence. A reputation scheme should be build in order to filter such information. We will not implement any reputation scheme protocol in this work and will assume that received information is correct. Unfortunately, even then, collaboration may have negative effects. Presence of inaccurate statistics is increased after exchanging such statistics. On the other hand, statistics might be refined based on the knowledge of node's neighbours. According to our aim, we need to decide if the IDS should prefer information coming from its neighbouring nodes (which might be richer in some context) or its own knowledge. Furthermore, received statistics can be joined together with node's own statistics or can be used only to refine them. In the second case, the number of neighbours would not grow.

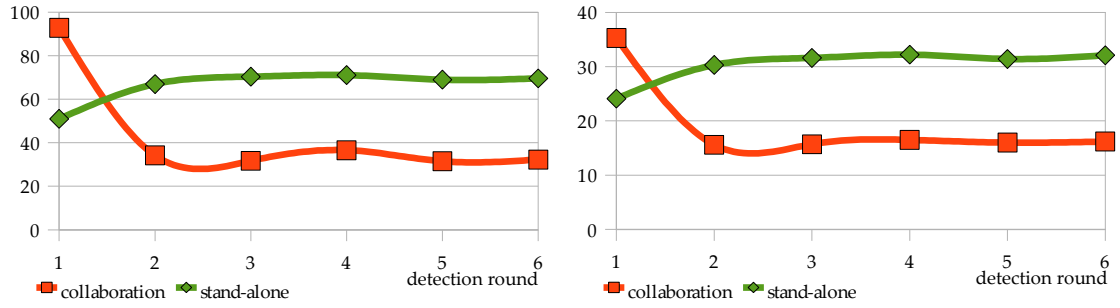
The collaboration component, as designed in this work, starts exchanging messages at the moment when detection is supposed to occur (detection is rescheduled after collaboration has finished). The local statistics are stored in the collaboration database and so the statistics component can be restarted to gather new statistics. Collaboration messages are broadcast to 1-hop neighbours in several packets, each describing two monitored nodes. On receiving such a message, the IDS agent stores the records the message is carrying into the collaboration database. The exchange message period ends after predefined time. After this

time, nodes discard any collaboration messages until the actual detection round finishes. All the received statistics are joined together and with the IDS agent's own ones.

It is common that several IDS agents running on neighbouring nodes monitor the same node. Several records describing the same node can be found in the collaboration database then. The one with the highest PRR is kept in our IDS. It is presumed that a node usually knows only about part of the messages received by its neighbour (see Figure 4.4). However, it cannot know about messages that were not sent to its neighbour. Hence, the node which heard the highest amount of the packets is the one possessing the most accurate information about the neighbour's PRR. The resulting set is passed to the detection engine for detection.

There is 2-hop neighbours collaboration implemented too. It is done in two rounds. The first one is the same as described in the previous paragraph. When having all the statistics joined together, they are said to be the statistics gathered by the actual IDS agent and sent out again the same way. The proposed IDS comes in three modifications – the one without collaboration and two with it – namely the 1-hop and 2-hop collaboration.

We would like to know whether the collaboration component eliminates occurrence of false positives when revealing random jamming. Assumption is that it may do so due to more precise information about the $PRRs$ of the monitored nodes. However, the number of false positives may be multiplied by accepting falsely alert positive statistics from other nodes at the same time. Simulations of 224 nodes as described in Section 4.4.1 were executed having $\delta_{\frac{PRR}{PSR}}$ set to 60 % (hence a node is allowed to have the received to sent ratio of 60 % less then it is common in its neighbourhood). It was found out that the number of false positives decreased to 32.66 (from 69.42 when collaboration was not used). It may look like the threshold is too vague, however, it is chosen empirically based on observations about received to sent ratios of non-malicious nodes. The number of nodes announced was 16 instead of 31.52 for the stand-alone detection. The comparison of the two approaches per each detection round can be seen in Figure 4.8.



(a) false positives (b) nodes falsely accused as malicious

Figure 4.8: Stand-alone and 1-hop collaboration detection of random jamming in a CTP network comparison

Unfortunately, even when the proposed collaboration component was used, the number of false positives is quite high and hence detection results are inaccurate. The aim of

the collaboration component was to gain richer information about the number of received packets by the node's neighbours. This was achieved just partially. If additional information about packets' origins is stored for each packet, the statistics received from collaborating neighbours could be joined together in better fashion and detection results might get better too. The actual PRR of a node would be composed of all the $PRRs$ gained using the collaboration component instead of taking the maximum value (current approach). Unfortunately, this solution is too expensive as the node's unique address is 16 bits long and there are twenty neighbours being monitored. Memory requirements of storing the $PRRs$ of the neighbourhood of the node running the IDS would increase from 0.4375 kB to 140 kB which is not bearable. Even if a hash table is used, memory of size of 28 kB would be needed.

Chapter 5

Simulations and results

This chapter starts with brief instructions how to deploy our intrusion detection system on Tmote Sky hardware or TOSSIM simulator. Description of performed simulations in networks involving malicious nodes and the results from these simulations follow. Depiction of several optional IDS' settings and their influence on performance of the IDS can be found in this chapter as well.

5.1 Deployment of the IDS

The proposed IDS can be incorporated into any wireless sensor network application. However, its performance depends on the configuration of the IDS and the actual network. The IDS can be configured before compiling by defining macros in the application's `Makefile`. Description of these configuration options can be found in the `Makefile.template` (disabling and enabling of clustering, collaboration or low gain monitoring). Furthermore, the detection period, low gain monitoring threshold, maximum number of monitored neighbours and thresholds used in detections can be set in `Ids.h`.

We only mention here that disabling hardware address recognition is necessary for the IDS to function. Otherwise the transceiver would reject packets that are not destined to it, hence the IDS agent would not be able to monitor its neighbourhood. Disabling hardware address recognition can be done by adding the following rule to the application's `Makefile`.

```
CFLAGS += -DCC2420_NO_ADDRESS_RECOGNITION
```

The path to the IDS source code has to be added there too:

```
CFLAGS += -I$(IDS_DIR)
```

Finally, the application can be compiled for Tmote Sky mote or for simulating using TOSSIM. After a node is booted, the IDS starts up – data acquisition of network traffic begins and the first detection is scheduled. `Detection` and `AlertDb` channels can be added in order to see detection outputs and alerts while simulation is running in TOSSIM. This is done by calling `addChannel(channel, output)` on the `Tossim` object in C++ or Python source code which manages the simulation. File pointer or standard output can be provided as the output parameter.

C++
```
Tossim* t = new Tossim(NULL);
t->addChannel("Detection", stdout);
```

Python
```
>>> t = Tossim([])
>>> t.addChannel("Detection", sys.stdout)
```

5.2 Simulating intrusion detection in networks with attackers

Results from simulations of networks including malicious nodes are summarized in this section. Firstly, simulations of a network with an aggregation protocol are described. CTP operating network simulations follow.

5.2.1 Networks with an aggregation protocol

The way data is aggregated is not matter to the functionality of our IDS. Hence, it was sufficient to alter the CTP for the purpose of simulating an aggregation protocol. The forwarding engine was disabled and so each node sent only its own values up the network tree. As the number of packets sent in the network was lowered, less retransmissions were needed and less collisions occurred while monitoring.

A network of 224 nodes set to low gain mode monitoring with the threshold of -88 dB was used for simulations. A node in this configuration had approximately 28 neighbours, however, only the subset of 20 of these neighbours with the highest $PRRs$ was monitored. Nodes sent the measured values from their sensors to the gateway every 8 seconds. The detection period τ was set to 304 seconds. The collaboration component was not used.

Six detection rounds were evaluated in 5 simulations for each attack. We wanted to know the number of false positives, nodes falsely accused as malicious, false negatives and number of alerts about malicious nodes. Five attackers were involved in each simulation. We tested hello flood, jamming and selective forwarding attacks. The threshold values differentiating malicious and trusted nodes were defined empirically.

The terms false positives, false negatives, nodes falsely accused as malicious and the number of alerts about malicious nodes have already been explained in Section 4.3.

5.2.1.1 Hello flood attack

A hello flood attacker was implemented as a node whose signal strength is 40 dB stronger than it would be if a normal node was used situated at the same place in the network. δ_{SS} was set to 30 dB which means that a node having its SS higher by more than 30 dB than the average value for its neighbourhood will be announced as an attacker.

The IDS was able to recognize all of the attackers and produced 92 to 98.2 alerts on average per detection round having 0 false positives. 824.6 to 966.2 false negatives were recorded on average per detection round. This is understandable. Messages from an attacker are re-

ceived by its neighbourhood with an extremely strong signal and nodes in this neighbourhood are able to reveal the attacker. However, there are a lot of nodes which are spatially far away from the attacker still receiving its messages with the SS value not so high any more. These nodes cannot point out the malicious node as the attacker. Furthermore, the radio range of the attacker is big and so there are plenty of such nodes.

5.2.1.2 Jamming

Five jammers were sending messages randomly at least every 1.5 seconds but not more often than every second. All of them were successfully revealed by their neighbours when δ_{PSR} was set to 100 messages. This means that the PSR of a node cannot be higher by more than 100 messages from the average PSR of the node's neighbourhood. The attackers were announced in 71 to 78 alerts on average in every detection round. No false positives and no false negatives occurred.

5.2.1.3 Selective forwarding

In our scenario, a selective forwarder is a node which omits its every other burst period. The IDS revealed all of the five selective forwarders having produced no false positives and false negatives. Malicious nodes were announced using 80 to 83 messages on average per detection round. $\delta_{PSR-MIN}$ was set to 10 packets.

5.2.2 Networks with the Collection tree protocol

The simulation environment for the CTP operating network is very similar to the one described in the section dedicated to networks with an aggregation protocol. The only differences are that nodes send messages containing values from their sensors every 16 seconds (the shortest sending period tested for the CTP by its authors in [25]) and, of course, the CTP is used to deliver these messages to the gateway. Clustering as it was described in the previous chapter is used for detection of deceptive and random jamming and selective forwarding attacks. Ten tests will be provided for each attack this time as the results tend to fluctuate more. Results from simulations of networks with hello flood attackers, deceptive and random jammers and selective forwarders will be shown in this section. The results are stated as average values per detection round excluding the very first one which usually provides quite bad detection results as the CTP network tree is not stable at this time yet.

First of all, we tried to estimate appropriate threshold values by running simulations without attackers. We focused on eliminating the number of false positives when setting the thresholds. Ten simulations in the described configuration were run with no attackers. The output of these simulations was used to find out the occurrence of errors introduced by individual attack-revealing methods. For the case of hello flood attack, the error is the number by which node's SS differs from the signal strength centre value. We define the error as the difference in node's PSR and sending rate centre value in case of deceptive jamming. The difference in the node's packet dropping ratio and common dropping ratio

and the difference in the node's received to sent ratio and the common received to sent ratio are the errors in case of selective forwarding and random jamming respectively. The results can be seen in Figure 5.1. According to these results we define individual thresholds (δ_{SS}, δ_{PDR} and $\delta_{\frac{PRR}{PSR}}$) for each of the relevant type of an attack.

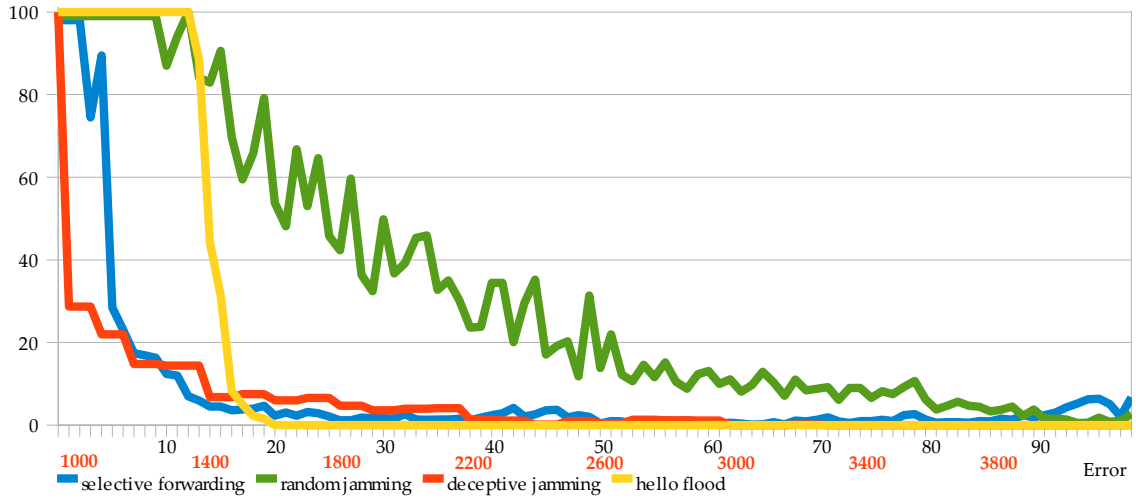


Figure 5.1: Occurrences of errors when detecting attacks (Error-axis gauge is in red for the case of deceptive jamming, in black for the other attacks)

5.2.2.1 Hello flood attack

The implementation of the hello flood attacker is the same as in the case for the networks with an aggregation protocol. δ_{SS} was set to 22 dB this time (see Figure 5.1). The IDS was supposed to reveal 5 attackers and succeeded in detecting all of them having no false positives at all. Attackers were announced in 244.25 warnings on average per detection round. The number of false negatives went up extremely (754.7 on average per detection round) again as in the case of the hello flood attack in the network with an aggregation protocol. The explanation of this behaviour can be found in Section 5.2.1.1. The results are pictured in Figure 5.2.

5.2.2.2 Deceptive jamming

The deceptive jammer is implemented as a node that after booting starts emitting packets without obeying the MAC protocol of the network. After a packet's sending procedure is finished another packet is sent immediately. Hence no other node in its neighbourhood is able to send a packet (other nodes either remain in the receiving mode or they are not able to reach the medium being idle). Such a jammer creates traffic of approximately 60,000 packets per detection period (hence it sends message approximately every 5 ms). δ_{PSR} was set to 2200 packets (see Figure 5.1).

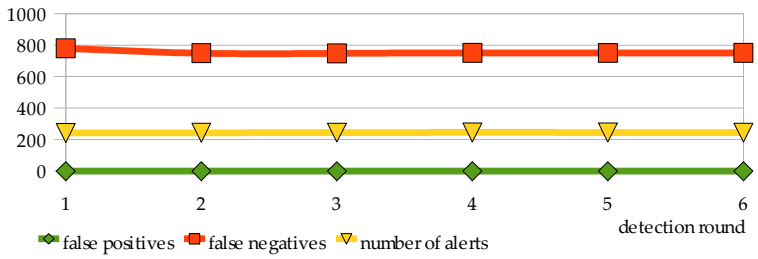


Figure 5.2: Detection of hello flood attack in a CTP network

Only one jammer is deployed in the network at a time because the presence of such a jammer heavily influences the whole network. The IDS was able to identify the jammer in approximately 10 alert messages per detection round. It produced no false positive and approximately 1 false negative per detection round (see Figure 5.3). The existence of the false negatives is introduced due to nodes which monitor the jammers although they are able to hear only a small subset of packets emitted by them (e.g. they are spatially far away from the attacker).

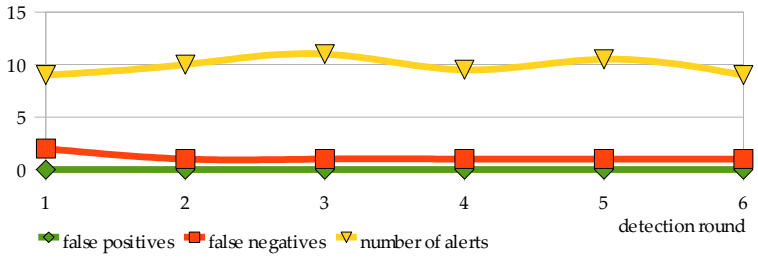


Figure 5.3: Detection of deceptive jamming in a CTP network

5.2.2.3 Random jamming

The way the random jammer is implemented is the same as described in Section 5.2.1.2. The only difference is the frequency of packet sending. The number of packets in the CTP operating network is higher than in the one using an aggregation protocol and hence the jammer should produce more messages too. It sends messages at least every 0.25 seconds but not more often than every 0.1 seconds.

Five jammers were involved in the network. The IDS considers a node being an attacker if its received to sent ratio deviates by more than 67 % ($\delta_{\frac{PRR}{PSR}}$) from the common received to sent ratio of the cluster. Stand-alone IDS configuration was not tested at all because of an extremely high number of false positives observed in simulations with no attackers (see the previous chapter, Figure 4.8-a). Collaboration among the IDS agents was enabled and statistics were exchanged with the node's 1-hop neighbours. Two collaborating scenarios were simulated.

In the first one, the IDS agent used information from its neighbours only to make its own statistics more precise. It did not adopt statistics about nodes which it did not monitor itself. The IDS was able to recognize only 2.65 of the 5 jammers on average in this scenario and produced 19.52 alert messages on average about them in each detection round. The IDS output 10.09 false positives of 6.41 nodes on average per detection round. It was not able to detect a jammer 54.3 times per detection round on average. The results are pictured in Figure 5.4. Unfortunately, they cannot be considered satisfying. As mentioned before, this is mostly due to inaccurate knowledge of the neighbours' $PRRs$. If the IDS does not know the accurate PRR of a trusted node, it may easily consider the node as a malicious one. Furthermore, if there is a malicious node in the neighbourhood of nodes for which the IDS does not know their $PRRs$, the common received to sent ratio of this neighbourhood will be low. In case the common received to sent ratio is lower than $\delta_{\frac{PRR}{PSR}}$ (set to 67 % in our simulations), the attacker will not be identified.

In the case of the other simulation scenario, the IDS agent joined together all of the received statistics. Hence the number of nodes passed to the detection was higher – approximately 45. We will see that the number of false positives, falsely accused nodes and false negatives went higher (false positives – 36.63 instead of 10.09, falsely accused nodes – 9.5 instead of 6.41, false negatives – 97.09 as opposed to 54.3). However, the number of alerts and the number of revealed attackers were also higher. The IDS was able to reveal 3.24 attackers on average per detection round in 43.44 alert messages (2.65 attackers in 19.52 alerts in the first scenario). The results of the second scenario per detection period can be seen in Figure 5.5.

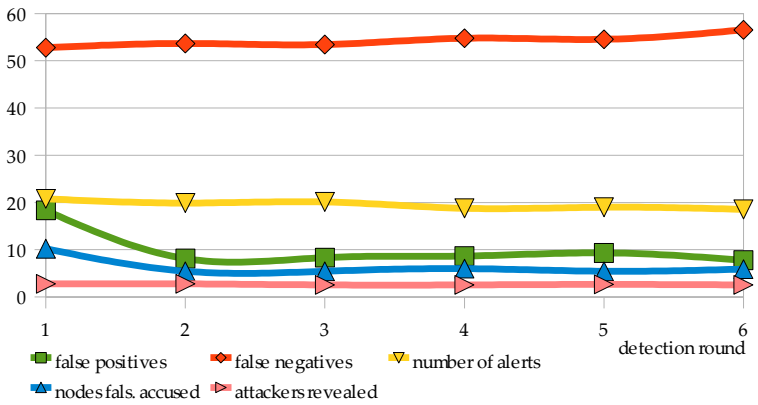


Figure 5.4: 1-hop collaboration detection of random jamming in a CTP network - intersection of the exchanged statistics

5.2.2.4 Selective forwarding

Only nodes which serve as forwarders (internal nodes of the CTP tree) are able to conduct a selective forwarding attack. The network tree may change in time and its form cannot

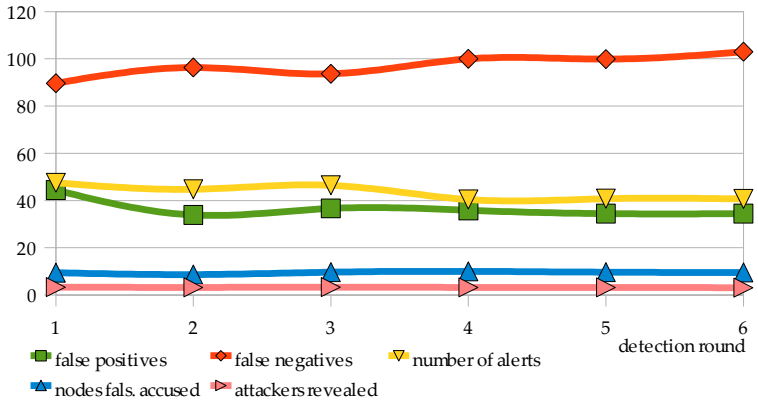


Figure 5.5: 1-hop collaboration detection of random jamming in a CTP network - union of the exchanged statistics

be predicted. In order to simulate selective forwarders in our simulation, they are chosen randomly from subgroups of nodes that forwarded at least 300 messages during the first detection round. The required number of forwarded messages was chosen empirically and it ensures that there are enough nodes to choose from (21 nodes on average). And most importantly, these nodes can be considered steady forwarders which will not tend to become leaf nodes later during the simulation. If any of these nodes became a leaf node, simulation would have to be restarted.

4.4 attackers on average were involved in each simulation and the IDS tried to detect them with δ_{PDR} set to 16 %. All of the attackers were identified in each detection period and announced by 60.6 alerts on average. 1.52 nodes were falsely announced in 8.4 false positive alerts per detection round. The number of false negatives was 42 on average per detection round. The stated results are an average of all the detection rounds but the first one during which attackers were not active throughout the whole round (just after forwarding the first 300 packets). Results involving the very first detection can be seen in Figure 5.6.

Another ten simulations were performed in the same configuration but using the collaboration component. The number of alerts increased to 291.64. The number of false negatives remained more or less the same (36.6). However, 1.2 trusted nodes on average per detection round were evaluated as attackers by mistake in 41.48 warnings. Results are depicted in Figure 5.7.

5.3 Configuration options of the IDS

Several advanced options of the IDS configuration are briefly presented here. The results from simulations using these options are pictured in this section as well.

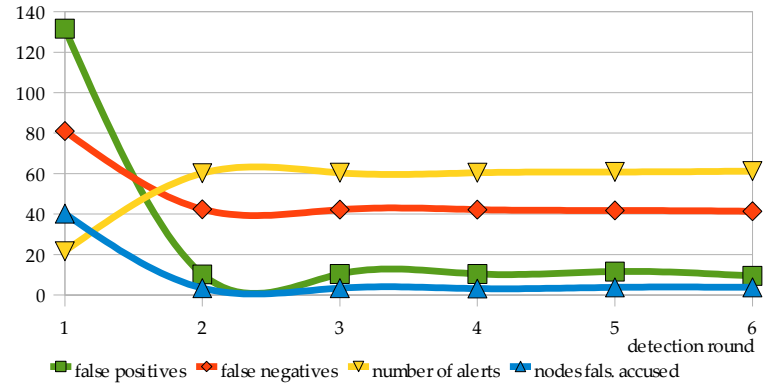


Figure 5.6: Stand-alone detection of selective forwarding in a CTP network

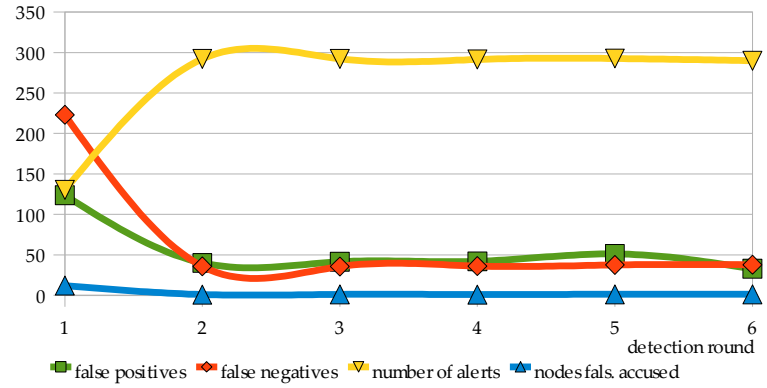


Figure 5.7: 1-hop collaboration detection of selective forwarding in a CTP network

5.3.1 Sparse networks

If the IDS is deployed in a sparse network, several problems have to be overcome. First of all, each node has less neighbours and it may happen that the set of nodes that the IDS agent is able to monitor is very small. Then, it cannot serve as a good statistical sample for determining the centre values vector or for computing the clusters. The solution proposed here is to let the nodes exchange their gathered statistics data and use the unions of them to perform detections.

We simulated this scenario in the configuration described before but different network topology was used. A node had approximately 7 neighbours in this topology (24 nodes had only 3 or less neighbours). After one hop exchange every node had a set of statistics of approximately 24 nodes on average to be passed to the detection component. The results of selective forwarding detection using 1-hop collaboration in a CTP operating network can be seen in Figure 5.8. The IDS was able to reveal all of the attackers in every detection round.

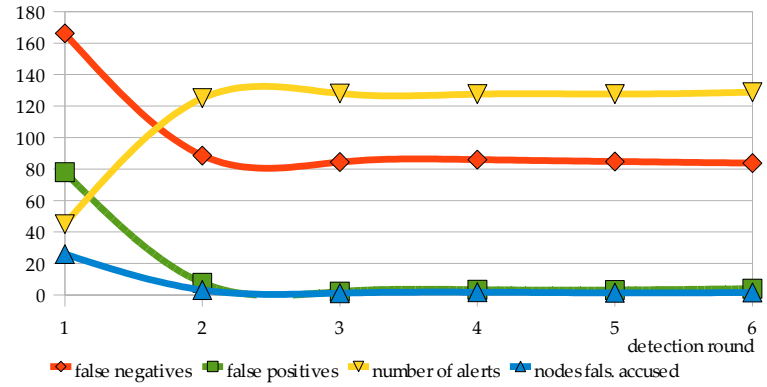


Figure 5.8: 1-hop collaboration detection of selective forwarding in a sparse CTP network

5.3.2 2-hop collaboration

So far, only simulation results of 1-hop collaboration have been summarized in this chapter. However, the IDS also offers 2-hop collaboration. It should be noted that transmitting over the radio is the most costly operation that motes perform. However, the number of packets transmitted in a network with the CTP is not significatnly increased by employing 2-hop collaboration. The description how the 2-hop collaboration is implemented is described in the previous chapter.

Ten simulations of the network operating the CTP were averaged per detection round and results of selective forwarding attackers detection can be seen in Figure 5.9. The description of the network and the IDS configuration can be found in Section 5.2.2. In comparison to the 1-hop collaboration results described in Section 5.2.2.4, the number of false positives is higher (62.28 in contrast to 41.48) and the number of false negatives is lower (26.4 instead of 36.6). The number of nodes falsely accused remains approximately 1. The number of alerts increased significantly to 467.48 (291.64 for the case of 1-hop collaboration).

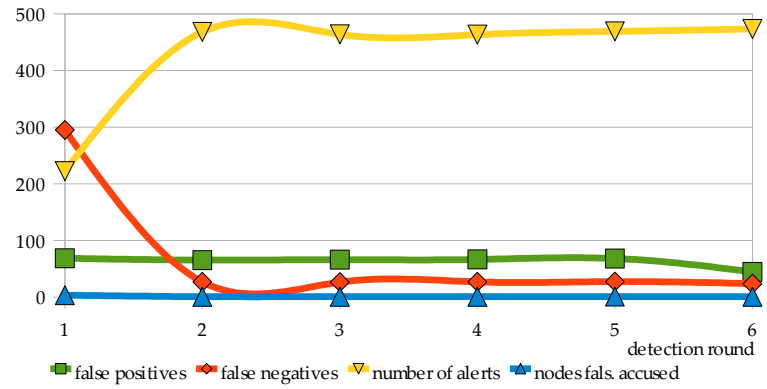


Figure 5.9: 2-hop collaboration detection of selective forwarding in a CTP network

Results from revealing attacks using stand-alone detection in data aggregation protocol operating networks can be considered very good for all types of attacks. Hello flood and deceptive jamming attacks conducted in networks with the CTP can be revealed reliably by stand-alone detection too (as the results from Section 5.2.2 shows). Hence, only results from the 2-hop collaboration of random jamming detection will be depicted in this section (Figure 5.10 and 5.11). Again (see Section 5.2.2.3), two means of treating the received statistics were simulated. If the results are compared with the 1-hop collaboration detection, no better performance can be seen in using the 2-hop collaboration when the local statistics are only refined by the received ones. However, the 2-hop collaboration and union of statistics brought results which are better in every fashion than any other simulated results. Most importantly, the number of revealed attackers increased to 4.13 (there were 5 attackers included in each simulation).

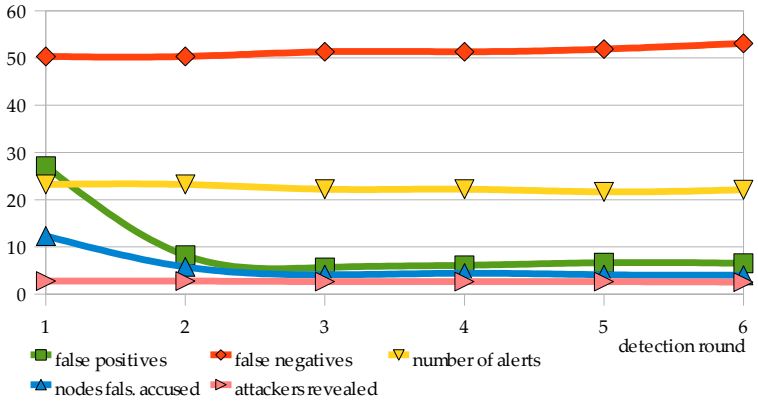


Figure 5.10: 2-hop collaboration detection of random jamming in a CTP network - intersection of the exchanged statistics

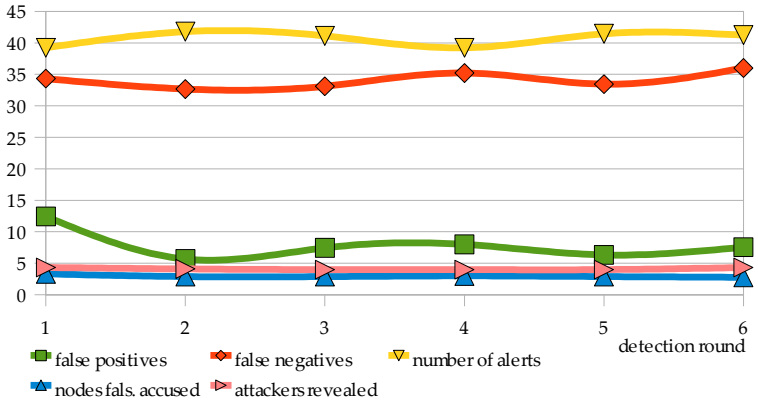
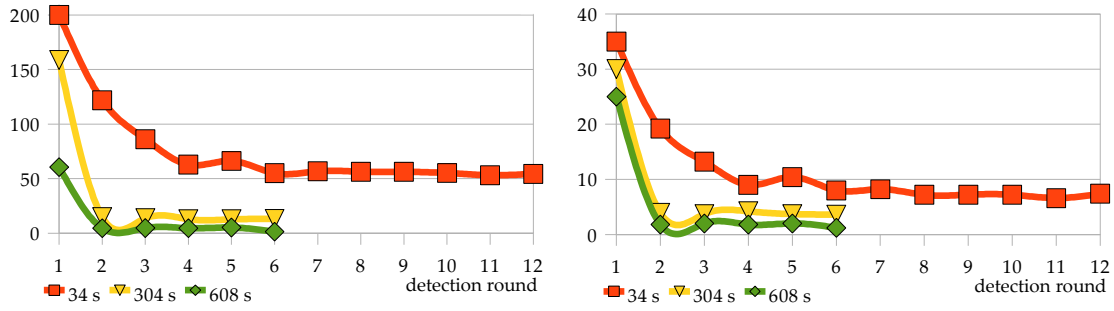


Figure 5.11: 2-hop collaboration detection of random jamming in a CTP network - union of the exchanged statistics

5.3.3 Detection interval length influence

One of the possible options in the IDS configuration is to set the length of the detection interval τ. It may be the case that the interval is needed to be short so any intrusion is announced as soon as possible. However, this may have negative influence on the detection results. We illustrate this in Figure 5.12 where averaged results of 10 simulations of the CTP operating network with no attacker included are depicted. As the graphs show, the number of false positives (Figure 5.12-a) and falsely accused nodes (Figure 5.12-b) when detecting selective forwarding grows as τ is set shorter. The same network model was used as described in the previous sections. τ was set to 34, 304 and 608 seconds.



(a) false positives

(b) nodes falsely accused as malicious

Figure 5.12: Detection interval length influence on detection of selective forwarding in a CTP network

Chapter 6

Conclusion

We have dealt with the neighbour-based intrusion detection technique in this work. In order to design a generic IDS capable of revealing several types of attacks at a time, symptoms and statistics have to be defined for these attacks. This work provides a study of several attacks which can be conducted in WSNs and describes possible statistics which can be used to reveal them. It is pointed out that only some of them are appropriate for the neighbour-based detection technique as the technique was defined in [13, 12].

A neighbour-based IDS was designed, implemented and evaluated in this work. It was programmed to detect different types of jammers, hello flood and selective forwarding attackers. It was shown that detection in networks with an aggregation protocol provides very good and stable detection results. The IDS was able to reveal all of the attackers in every single simulation producing no false positives or false negatives in most cases.

Furthermore, we were able to enhance the IDS to be able to detect mentioned attackers in networks with the CTP as well. We presented the limitations of basic detection methods and described how clustering can be used in order to achieve better performence. By employing clustering method, we decreased the number of false positives and false negatives almost to 0 in case of detection of deceptive jamming. The IDS was never able to reveal all of the random jammers deployed in the network. The best detection results when the number of false positives was below 10 and at the same time approximately 4 random jammers out of 5 deployed were revealed was achieved only by enabling 2-hop collaboration. Detection of selective forwarding in CTP networks provided approximately 60 accurate alerts revealing all the malicious nodes (5 of them) and accused falsely only less than 2 nodes on average. Even though the number of false negatives was quite high (approximately 42), it was below the number of the accurate alerts.

The IDS comes in three modifications - stand-alone, 1-hop and 2-hop collaboration. We described several ways how to deal with information coming from the node's neighbours. Collaboration can be used in order to refine the statistics gathered by the IDS agent. However, it can also be used when the IDS is deployed in a sparse network where the number of monitored nodes by a single IDS agent is too low. IDS agents share their knowledge of their neighbourhoods and gain information about their 2-hop or 3-hop neighbours in such a case.

It should be noted that an important advantage of the neighbour-based detection technique is that it requires no prior training. However, the IDS has to be configured before it is deployed in a specific network. The automation of these configuration settings could improve the usability of such IDS in the future. Study of advanced clustering techniques could

enhance the current solution and provide better detection results in networks without an aggregation protocol. However, this is left for future work in this field.

Bibliography

[1] Wikipedia: *Wireless sensor network*, document available (January 2010) at URL, <http://en.wikipedia.org/wiki/Wireless_sensor_network> . 1, 1.2

[2] Wikipedia: *Sensor node*, document available (January 2010) at URL, <http://en.wikipedia.org/wiki/Sensor_node> . 1

[3] Wikipedia: *TinyOS*, document available (January 2010) at URL, <http://en.wikipedia.org/wiki/TinyOS> . 1

[4] Stetsko, A.: *Intrusion detection for wireless sensor networks*, [Dissertation thesis topic], Masaryk University, Faculty of Informatics, Brno, Czech Republic. Document available (January 2010) at URL, <http://is.muni.cz/th/184905/fi_r/thesis_topic.pdf> . 1, 1.3, 2.1

[5] Roman, R. and Zhou, J. and Lopez, J.: *Applying intrusion detection systems to wireless sensor networks*, In CCNC 2006: 3rd IEEE Consumer Communications and Networking Conference, Washington, USA, IEEE Computer Society 2006, 640–644. 1, 2.1

[6] Moteiv: *Tmote Sky low power sensor module*, document available (January 2010) at URL, <http://www.bandwavetech.com/download/tmote-sky-datasheet.pdf> . 1.1

[7] Crossbow Technology Inc.: *Stargate NetBridge embedded sensor network gateway*, document available (January 2010) at URL, <http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Stargate_NetBridge_Datasheet.pdf> . 1.1

[8] Stankovic, J.: *How things work, Wireless sensor networks*, document available (January 2010) at URL, <http://www.cs.virginia.edu/~stankovic/psfiles/r10how.pdf> . 1.2

[9] TinyOS Alliance: *TinyOs*, document available (January 2010) at URL, <http://www.tinyos.net/> . 1.2

[10] Shnayder, V. and Hempstead, M. and Chen, B. and Allen, G. and Welsh, M.: *Simulating the power consumption of large-scale sensor network applications*, In SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems, New York, USA, ACM 2004, 188–200. 1.2, A

[11] Roman, R. and Lopez, J. and Gritzalis, S.: *Situation awareness mechanisms for wireless sensor networks*, Communications Magazine, IEEE, 46, 2008, 4, 102–107. 2, 2.2, 4, 4.1

[12] Li, G. and He, J. and Fu, Y.: *A Group-Based Intrusion Detection Scheme in Wireless Sensor Networks*, In GPC-WORKSHOPS '08: Proceedings of the 3rd International

Conference on Grid and Pervasive Computing - Workshops, Washington, USA, IEEE Computer Society 2008, 286–291. 2.1, 2.3.2, 3.1, 3.4, 3.7, 3.8, 6

[13] Liu, F. and Cheng, X. and Chen, D.: *Insider Attacker Detection in Wireless Sensor Networks*, In INFOCOM 2007: 26th IEEE International Conference on Computer Communications, Washington, USA, IEEE Computer Society 2007, 1937–1945. 2.3.1, 3.7, 3.8, 6

[14] Roosta, T. and Shieh, S. and Sastry, S.: *Taxonomy of Security Attacks in Sensor Networks and Countermeasures*, In The First IEEE International Conference on System Integration and Reliability Improvements, Hanoi, 2006, 13–15. 3

[15] Kumar, H. and Sarma, D. and Kar, A.: *Security Threats in Wireless Sensor Networks*, In 40th IEEE International Carnahan Conference on Security Technology, Washington, USA, IEEE Computer Society 2006, 243–251. 3

[16] Datema, S.: *A Case Study of Wireless Sensor Network Attacks*, [Master thesis], Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft, Netherlands. Document available (January 2010) at URL, <http://www.pds.twi.tudelft.nl/education/masters/theses/MSc-thesis-Datema.pdf> . 3

[17] Xu, W. and Trappe, W. and Zhang, Y. and Wood, T.: *The feasibility of launching and detecting jamming attacks in wireless networks*, In MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, New York, USA, ACM 2005, 46–57. 3.1, 3.1

[18] Xu, W. and Ma, K. and Trappe, W. and Zhang, Y.: *Jamming sensor networks: attack and defense strategies*, IEEE Network, 20, 2006, 3, 41–47. 3.1

[19] Freiling, F. and Krontiris, I. and Dimitriou, T.: *Towards Intrusion Detection in Wireless Sensor Networks*, [Technical paper], Paris, France, 13th European Wireless Conference, 2007. Document available (January 2010) at URL, <http://pi1.informatik.uni-mannheim.de/filepool/publications/krontiris-ew2007.pdf> . 3.3

[20] Hai, T. and Huh, E.: *Detecting Selective Forwarding Attacks in Wireless Sensor Networks Using Two-hops Neighbor Knowledge*, In NCA '08: Proceedings of the 2008 Seventh IEEE International Symposium on Network Computing and Applications, Washington, USA, IEEE Computer Society 2008, 325–331. 3.3

[21] Krontiris, I. and Dimitriou, T. and Giannetsos, T. and Mpasoukos, M.: *Intrusion Detection of Sinkhole Attacks in Wireless Sensor Networks*, In ALGOSENSORS 2007: 3rd International Workshop on Algorithmic Aspects of Wireless Sensor Networks, Heidelberg, Germany, Springer-Verlag 2007, 150–161. 3.4

[22] Demirbas, M. and Song, Y.: *An RSSI-based Scheme for Sybil Attack Detection in Wireless Sensor Networks*, In WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks, Washington DC, USA, IEEE Computer Society 2006, 564–570. 3.5

[23] Wang, J. and Yang, G. and Sun, Y. and Chen, S.: *Sybil Attack Detection Based on RSSI for Wireless Sensor Network*, In WiCom 2007: International Conference on Wireless Communications, Networking and Mobile Computing, Washington, USA, IEEE Computer Society 2007, 2684–2687. 3.5

[24] Wen, M. and Li, H. and Zheng, Y. and Chen, K.: *TDOA-based Sybil attack detection scheme for wireless sensor networks*, Journal of Shanghai University (English Edition), 12, 2008, 66–70. 3.5

[25] Gnawali, O. and Fonseca, R. and Jamieson, K. and Moss, D. and Levis, P.: *Collection tree protocol*, In SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, 7, New York, USA, ACM 2009, 1–14. 4, 4.4.1, 5.2.2

[26] TinyOS Alliance: *The Collection Tree Protocol (CTP), TEP-123*, document available (January 2010) at URL, <http://tinyos.net/tinyos-2.x/doc/html/tep123.html> . 4

[27] Honus, L.: *Návrh, implementace a simulace systému detekce průniku v bezdrátových senzorových sítích*, [Master thesis], Masaryk University, Faculty of Informatics, Brno, Czech Republic. Document available (January 2010) at URL, <http://is.muni.cz/th/139910/fi_m/master.pdf> . 4.1, 4.2, 4.2

[28] Stetsko, A. and Matyas, V.: *Effectiveness Metrics for Intrusion Detection in Wireless Sensor Networks*, In EC2ND 2009: Proceedings of European Conference on Computer Network Defense, Milan, Italy, 2009. 4.3

[29] Wikipedia: *k-means clustering*, document available (January 2010) at URL, <http://en.wikipedia.org/wiki/K-means_clustering> . 4.4.1

[30] Prabhakar, T. and Venkatesh, S. and Sujay, M. and Kuri, J. and Praveen, K.: *Simulation blocks for TOSSIM-T2*, In COMSWARE 2008: 3rd International Conference on Communication Systems Software and Middleware and Workshops, Washington, USA, IEEE Computer Society 2008, 17–23. A

[31] Perla, E. and Cathain, A. and Carbajo, R. and Huggard, M. and Mc Goldrick, C.: *PowerTOSSIM z: realistic energy modelling for wireless sensor network environments*, In PM2Hw2N '08: Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks, New York, USA, ACM 2008, 35–42. A

[32] Distributed Systems Group, Trinity College Dublin: *PowerTOSSIM-Z: Realistic Energy Model in TOSSIM for the MicaZ Mote*, document available (January 2010) at URL, <https://www.cs.tcd.ie/~carbajor/powertossimz/index.html> . A.1

[33] Krontiris, I. and Giannetsos, T. and Dimitriou, T.: *LIDeA: a distributed lightweight intrusion detection architecture for sensor networks*, In SecureComm '08: Proceedings of the 4th international conference on Security and privacy in communication netowrks, New York, USA, ACM 2008, 1–10.

[34] da Silva, A. and Martins, M. and Rocha, B. and Loureiro, A. and Ruiz, L. and Wong, H.: *Decentralized intrusion detection in wireless sensor networks*, In Q2SWinet '05: Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks, New York, USA, ACM 2005, 16–23.

[35] Zhang, Y. and Lee, W.: *Intrusion detection in wireless ad-hoc networks*, In MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking, New York, USA, ACM 2000, 275–283.

[36] TinyOS Alliance: *Packet Protocols, TEP-116*, document available (January 2010) at URL, <http://tinyos.net/tinyos-2.x/doc/html/tep116.html> .

[37] TinyOS Alliance: *message_t, TEP-111*, document available (January 2010) at URL, <http://tinyos.net/tinyos-2.x/doc/html/tep111.html> .

[38] TinyOS Alliance: *Hardware Abstraction Architecture, TEP-2*, document available (January 2010) at URL, <http://tinyos.net/tinyos-2.x/doc/html/tep2.html> .

[39] TinyOS Alliance: *TinyOS 2.x Boot Sequence, TEP-107*, document available (January 2010) at URL, <http://tinyos.net/tinyos-2.x/doc/html/tep107.html> .

[40] Levis, P.: *TinyOS Programming*, [Technical manual], California, USA, TinyOS Alliance, 2006. Document available (January 2010) at URL, <http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf> .

[41] Gay, D. and Levis, P. and Culler, D. and Brewer, E.: *nesC 1.1 Language Reference Manual*, [Technical manual], California, USA, TinyOS Alliance, 2003. Document available (January 2010) at URL, <http://nescc.sourceforge.net/papers/nesc-ref.pdf> .

Appendix A

Energy consumption estimation using PowerTOSSIM

Sensor nodes run on batteries and so the evaluation of software's energy consumption is important. PowerTOSSIM was originally developed at Harvard University [10] and was dedicated for TinyOS 1.x networks. It was based on the energy model of mica2 mote which uses the CC1000 radio chip. Later, it was redesigned for TinyOS 2.x [30]. There is a version of PowerTOSSIM for TinyOS 2.x available from Trinity College Dublin researches which used the micaZ mote with the CC2420 radio chip [31]. We used this version of PowerTOSSIM to estimate power consumption of our intrusion detection system (the Tmote Sky node for which our IDS can be compiled uses the CC2420 radio chip).

A.1 Deployment in PowerTOSSIM

PowerTOSSIM downloaded from [32] is dedicated for TinyOS 2.x simulations. Our IDS is written for TinyOS 2.1.0. Even though these versions are compatible, some of the features may not be available.

After uncompressing the downloaded archive of PowerTOSSIM, there are three directories obtained – `tos`, `postprocessZ` and `powercurses`. The first one is a replica of the directories and files that were modified inside the TinyOS 2.x tree. They can be hard-copied into TinyOS' installation or included at the compile time by defining appropriate rules in the application's `Makefile`. After accomplishing this part, PowerTOSSIM can be considered as installed. In order to record power consumption of a simulation, `ENERGY_HANDLER` channel needs to be added to the simulation source code. Once the output of this channel is obtained in a file, a post-processor can be used to summarize the energy consumption results. It is located in `postprocessZ` directory. In order to see all of its available options, the `python postprocessZ.py` command can be run in the shell. PowerCurses is a software for graphical representation of nodes' energy consumption and can be found in the last directory.

A.2 Energy consumption simulations

We would like to know the energy consumption introduced by running our intrusion detection system on every node of a network. The most expensive operation that nodes perform is communication using the radio. Therefore, we will be interested in comparing energy consumption of the IDS with and without the collaboration.

We ran simulations of stand-alone modification of the IDS, 1-hop collaborating modification of the IDS and a simulation of a network without the IDS installed. The simulated networks were composed of 224 nodes that sensed the temperature every 16 seconds and sent the measured values using the CTP to the gateway. Nodes were set to the low gain monitoring mode with the threshold of -88 dB. It should be noted that this should positively influence the energy consumption even in the case when the IDS agent is not running on a node. The detections (when the IDS was installed) were held every 304 seconds and 6 such detections were simulated during 2001 seconds. We only mention that running the post-processor on the output of such a simulation takes several hours (Intel Core Duo 1.7 GHz, 1 GB RAM). The energy consumptions results can be seen in Table A.1.

	CPU (mJ)	*Radio chip (mJ)*	*Messages sent*
no IDS installed	117.43	117776.26	670
stand-alone IDS	117.43	117774.23	769
collaborative IDS	117.43	117774.75	731

Table A.1: Average sensor node energy consumption

The simulated energy consumption of a node on average seems to be the same for all the tested scenarios. This was not expected. The IDS is believed to consume a lot of energy because of its requirement to disable early dropping of packets which are not destined to the node running the IDS agent (this is done by disabling address recognition in the application's `Makefile`). Unfortunately, the TOSSIM network model used for the simulations of our IDS cannot be set to early packet dropping mode. Hence, we programmed this feature for TOSSIM. It may be the case that PowerTOSSIM is not compatible with this change and hence the power consumption of the radio chips remains the same even when the IDS is turned off.

Almost no difference in energy consumption was found when comparing the stand-alone and 1-hop collaborating modifications of the IDS. This might be due to a huge number of messages sent when the CTP is used to deliver packets to the base station. Furthermore, the number of messages differs for every simulation.

From another point of view, we may estimate power consumption of the collaboration component in means of what part of the total radio chip's consumption it represents. Five simulations of the IDS having its collaboration component enabled were performed. Until the beginning of the sixth detection, there were approximately 170,000 packets sent in total. Only 6,700 messages (less than 4 %) were produced by the collaboration component. Hence, the power consumption of the collaboration component is 4 % (the CPU energy consumption is not taken into account because the total CPU consumption in the simulations performed with PowerTOSSIM was only 0.01 % of the total node's energy consumption).