Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Dissertação de Mestrado

Mestrado em Engenharia Informática

Intrusion Tolerant Routing Protocols for Wireless Sensor Networks

André Ivo Azevedo Guerreiro (aluno nº 31209)

2º Semestre de 2010/11

23 de Setembro de 2011

Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Dissertação de Mestrado

Intrusion Tolerant Routing Protocols for Wireless Sensor Networks

André Ivo Azevedo Guerreiro (aluno nº 31209)

Orientador: Prof. Doutor Henrique João Lopes Domingos

Trabalho apresentado no âmbito da unidade curricular de Preparação de Dissertação do Mestrado em Engenharia Informática, como requisito parcial para obtenção do grau de Mestre em Engenharia Informática.

2º Semestre de 2010/11

23 de Setembro de 2011

Resumo

Esta tese é focada no estudo, desenvolvimento e avaliação experimental de soluções de segurança para redes de sensores sem fios com tolerância a intrusões.
A contribuição principal é baseada na criação de propriedades de tolerância a intrusões ao nível rede, propondo mecanismos para um sistema de encaminhamento de dados seguro com tolerância pró-activa contra intrusões. Estas propriedades visam aumentar a segurança fornecida por uma possível camada segura de comunicação, baseada em mecanismos criptográficos leves adaptados às características das redes de sensores sem fios, melhorando a resiliência da rede contra possíveis ataques por intrusão.
A solução proposta tem como referência a construção, de forma segura e eficaz, de um sistema de encaminhamento estruturado numa árvore para disseminação segura de dados, tolerando os danos causados por atacantes às comunicações, modelados de acordo com as hipóteses de Dolev-Yao ou de acordo com a tipologia de ataques enquadrados no modelo OSI X.800. Adicionalmente, a solução incorpora tolerância a intrusões que visem comprometer maliciosamente os nós da rede, com o objectivo de obter informação associada a segredos criptográficos ou que visem injectar, modificar ou bloquear o processamento de pacotes e que ponham em risco o funcionamento do encaminhamento e a manutenção segura da topologia da rede.
O sistema de encaminhamento faz uso de um protocolo de encaminhamento, utilizando mecanismos multi-rota e multi-caminho, vocacionado para redes de sensores sem fios de grande escala. Estes mecanismos podem ser combinados com protocolos de consenso distribuído probabilístico de dados, desempenhados por múltiplas estações base de captura de dados, melhorando significativamente a resiliência global da rede. Esta arquitetura afasta as tarefas computacionalmente pesadas associadas a consensos de dados dos sensores para as estações base, aproveitando o facto de estas representarem nós da rede com mais recursos e condições de confiabilidade.
Os mecanismos propostos foram avaliados experimentalmente por simulação de redes de sensores sem fios de larga escala (de centenas até dezenas de milhares de nós), para obtenção de medidas de impacto das seguintes métricas: condições de conectividade, padrões de equilíbrio e distribuição de carga, caminhos médios e implicações de latência, organização em cluster, condições de fiabilidade e avaliação de custo energético.

Palavras-chave: redes de sensores sem fios, ambientes de simulação rssf, rssf multi-hop de larga escala, mecanismos de gestão ad-hoc, tolerância a intrusões e resiliência, confiabilidade, propriedades de segurança (autenticação, confidencialidade, integridade)

Abstract

This MSc thesis is focused in the study, solution proposal and experimental evaluation of security solutions for Wireless Sensor Networks (WSNs). The objectives are centered on intrusion tolerant routing services, adapted for the characteristics and requirements of WSN nodes and operation behavior.
The main contribution addresses the establishment of pro-active intrusion tolerance properties at the network level, as security mechanisms for the proposal of a reliable and secure routing protocol. Those properties and mechanisms will augment a secure communication base layer supported by light-weigh cryptography methods, to improve the global network resilience capabilities against possible intrusion-attacks on the WSN nodes. Adapting to WSN characteristics, the design of the intended security services also pushes complexity away from resource-poor sensor nodes towards resource-rich and trustable base stations.
The devised solution will construct, securely and efficiently, a secure tree-structured routing service for data-dissemination in large scale deployed WSNs. The purpose is to tolerate the damage caused by adversaries modeled according with the Dolev-Yao threat model and ISO X.800 attack typology and framework, or intruders that can compromise maliciously the deployed sensor nodes, injecting, modifying, or blocking packets, jeopardizing the correct behavior of internal network routing processing and topology management.
The proposed enhanced mechanisms, as well as the design and implementation of a new intrusion-tolerant routing protocol for a large scale WSN are evaluated by simulation. For this purpose, the evaluation is based on a rich simulation environment, modeling networks from hundreds to tens of thousands of wireless sensors, analyzing different dimensions: connectivity conditions, degree-distribution patterns, latency and average short-paths, clustering, reliability metrics and energy cost.

Keywords: Wireless Sensor Networks, WSN simulation environments, large scale multi-hop WSNs, Secure Routing, Ad-hoc topology management, Intrusion tolerance and Network resilience, security properties (authentication, confidentiality, integrity), dependability

Contents

1. Introduction

Wireless Sensor Networks(WSNs) consist in a number of small devices[32] called sensors which possess limited capabilities with regards to power, computation, communications, sensing and storage [61] [15].

The general purpose of WSN is to serve as a pervasive sensing environment, accessible from computer systems and applications by an interface to the real world, providing information about physical phenomena, such as: temperatures, light-conditions, radiation, mobile or fixed targets-detection, humidity, etc.

Such pervasive environments, that can work as specialized monitoring "islands" integrated in a more widespread inter-networked distributed environment, differ from conventional networks in their decentralization and specialized nature. In a WSN, sensors (sometimes known as motes) collaborate towards the common goal of obtaining and deducing certain physical information from their environment. Moreover, a WSN is capable of self-organization, thus it can be deployed in certain context without human supervision and without requiring the existence of supporting infrastructure.

WSNs are becoming more and more popular given the low cost of the sensors(with prices that range from 10US$ to 1US$ in some miniaturized and low-cost implementations)[32] [34] [16]. This makes WSNs economically viable solutions to apply in a broad variety of situations, either in controlled environments such as houses, offices, warehouses, cars, or in critical uncontrolled environments like forests, disaster areas, hostile war regions or toxic areas [7] [43] [7].

Depending on the number of nodes and their spatial distribution, WSN are able to sense the environment, according to different scale conditions, density and topology and other application requirements [7] [58] [26].

WSNs can be developed, in general, as a particular case of autonomous Ad-Hoc Networks, performing without human intervention or supervision, on low cost operation. These settings can be particularly relevant for large scale deployments, in which the sensors can disseminate data or events, supported by a multi-hop routing strategy, with sensors participating as event-sources, routing nodes, data-processing or data-aggregation points, as well as temporary limited data-storage devices.

With the above characteristics in mind, WSNs can be developed in many application domains.For example, in the agriculture business, WSNs can be used to monitor the humidity of the land near the plantations so it can trigger irrigation systems. For home/office environments, they can be used to monitor the temperature of the rooms and communicate with the air conditioning system. It is also possible to detect fires or floods, or even to prevent unauthorized physical access to some areas through wireless security systems. In cars, a WSN may be devised as a Vehicular AdHoc Network to monitor the pressure of the tyres or even in a dense monitoring environment of engine and chassis parameters. In a military scenario, WSNs can be stealthily dropped from airplanes in enemy territory to monitor,detect and track movement of soldiers or vehicles. On home land, WSNs may be deployed in forests to monitor temperature

and humidity in order to prevent or make an early detection of fires.

Given their immense possibilities of use, WSNs are rapidly emerging as an important new area in mobile research. The discussion about the variety of uses and some specific settings for different applications can be found in the literature[7] [43]

1.1 Impetus

Many of the above introduced application scenarios impose security and reliability concerns. Be it on a hostile land or in a home, lives and assets may depend on the information given by the WSN.

Communications between sensors are made via wireless transmitting, which increases the risk of attacks to the communications such as eavesdropping, unauthorized access, replaying, spoofing or denial of service. On one hand, the sensors possess very limited capabilities in regards to transmitting power, storage, computation power and battery. This poses limitations in regards to the cryptographic techniques possible to use as well as the radio transmitting range possible to achieve.

At last, there is the chance of an intrusion(by means of having physical access to the sensors) on one or more sensors, rendering their cryptographic material insecure and possibly altering their behaviour to a malicious one, possibly injecting arbitrary faults in the rest of the network.

The operation in large-scale unsupervised scenarios amplifies those concerns. As such, it is of primary importance that the information provided by the network is correct, authentic, not tampered and delivered on time, according to a correct dependable system behaviour.

In the context of this thesis we consider the WSN dependable if reliance can justifiably be placed on the service it delivers. In this vision, dependability will be based on design criteria including as special cases such attributes as: reliability, availability, security and resilience. This asks for modeling and analysis methods, as well as implementation techniques from design-time prediction of dependability attributes to its experimental assessment in large-scale settings.

The experimental assessment must take in account the characteristics and limitations of WSNs and relevant performance indicators, like: connectivity and topology management and stabilization conditions, latency conditions, energy-consumption impact and balancing criteria associated to internal network data processing and data dissemination.

With the above dependability criteria and assessment conditions in mind, reliability and security requirements for WSNs must be addressed as two faces of the same coin: it is necessary to warrant the correct behavior of the network defending it against two different types of attacks; external attacks to the communications, and attacks to the nodes themselves, to establish intrusion tolerance criteria. Both pose quite a challenge given the very nature of the WSN, combining appropriate counter-measures to preserve security (to resist against different attack typologies) and reliability (to resist against failures that can be accidental or caused by those attacks).

The establishment of dependability conditions for WSNs stands to the proposition of different security services at possible different levels, presented here in a bottom-up perspective:

a) at the hardware level and local processing protection (using code-attestation techniques, integrated or not in micro-tamperproof modules, code-obfuscation[51]);

b) at the MAC- level, protecting from MAC-layer attacks, such as: DoS, incorrect MAC processing[62], anti-jamming protection [57], [50];

c) at the data-link protocol-level, protecting radio-communications with security primitives using appropriate light-weight cryptography mechanisms [41], [63] [25] [9].

d) at the network services level, through the role of secure routing protocols and topology control mechanisms [49] [21] [64].

e) at the key-establishment services level, with different techniques to prevent a secure distribution and establishment of seeds, secrets or cryptgraphic keys [70], [68], [45]

f) at specific security mechanisms for secure data-dissemination [38]

g) at the level of secure data-aggregation techniques, that can be addressed in a more generic approach or specific solutions, according to the application requirements [3]

Complementary approaches for the above levels of security for WSNs, are well surveyed and can be found in the specialized literature[4] [42]

Among all these challenges, the focus and motivation of the thesis is more closely-related to the security level approach in c) and d), which we will discuss in detail in the related work (chapter 2 of this document). The main motivation is to augment the security of WSNs by designing a safer routing protocol in regards to the intrusion tolerance aspect.

1.2 Objectives of the thesis

This Master's thesis objectives are to study, experimentally evaluate and propose solutions to secure intrusion-tolerant routing algorithms in wireless sensor networks.

The final goal will be to address the design, implementation and experimental evaluation of an optimal routing protocol for large scale wireless sensor networks, supporting optimal tree-structured secure data dissemination strategies.

The protocol must tolerate the damage caused by an intruder who has compromised deployed sensor nodes with the intention of injecting, modifying, or blocking packets, to trigger the following type of attacks: rushing[67], sinkholes[33], wormholes[66] and blackholes[29].

To address this thesis's problematic, firstly, an in-depth review will be made regarding the

security threats and counter-measures possible in the MAC and network layers (including network topology management counter-measures, ad-hoc organization counter-measures and routing protocols). In this study, our attention will be focused in different adversary conditions, which could allow the mentioned attacks. We are particularly concerned with adversary conditions related with intrusion attacks. These may allow an adversary to compromise a certain amount of network nodes.In the approach of the thesis we are not considering sybil-attacks [31], Denial-Of-Service attacks [33] or spam-attacks [54] that also have repercussions at the network level services.

Given the main focus of this thesis's subject on the intrusion tolerance at the network layer to defend from the mentioned attacks, secure proactive routing protocols with intrusion tolerance characteristics will be studied. These protocols will also be analyzed with an experimental evaluation by simulation of large scale WSNs, with regards to their security properties and other performance indicators, namely:connectivity and topology management and stabilization conditions, latency conditions and energy-consumption impact.

To accomplish this porpuse, we will use a discrete event simulator implementing the radio model support as defined by the IEEE 802.15.4 standard behavior and specifications [25]. The simulation environment includes communication settings and parameterizations for mica-motes sensors, namely mica 2 [1], telosB [2], in WSN topologies of 1000 to 50000 nodes, providing tools to inject the typology of attacks described above.

The main focus will be to support an optimal secure tree-structured routing service for data-dissemination of large scale randomly deployed WSNs, with resilience characteristics and counter-measures against Dolev-Yao[22] or X.800[30] adversarial conditions.

The devised solution will include attack-prevention and attack-detection mechanisms as preventive and pro-active ones during the process of establishment and maintenance of multi-path routes.To this matter, the Byzantine[14] failure model will be studied as well as ways to circumvent its deterministic impossibilities, particularly in the case of WSNs. As a solution to the Santoro & Widmayer impossibility result [55], probabilist Byzantine agreement protocols will be used. These techniques aim to establish distributed consensus with innovative cryptographic schemes [13] [12].We intend to apply these randomization techniques in increasingly weaker variants of attacks, until an optimal intrusion tolerant consensus protocol for an attack-detection mechanism is achieved.

In the first variant we will begin by restricting the number of nodes that may be the source of internal attacks in each communication step, to have a practical implementation and observation of the performance indicators above mentioned.

In this phase, the evaluation of the attack-detection mechanism must tolerate f dynamic nodes as sources of incorrect behavior in a network of $n >= 3f + 1$, with n ranging from 1000 to 50000 nodes.

In the second variant, the idea is to have no restrictions on the pattern of nodes with incorrect

[1] CrossbowTechnology; www.xbow.com . Accessed in Jan/2011

[2] CrossbowTechnology; www.xbow.com . Accessed in Jan/2011

behaviors, simulating a more aggressive setting.

At the end, with all this knowledge, the objective is to study, develop, integrate and measure the implementation of the consensus protocols subjacent to the attack-detection mechanism running atop of secure and dependable routing services [49] [21] [64].

1.3 Expected main contributions

The expected main contributions of this thesis are:

- to create a simulation environment with the possibility to inject a typology of attacks to the routing layer(rushing[67], sinkholes[33], blackholes[29] and wormholes[66]), as consequences of intrusions.This task is already partially concluded in the beggining of this thesis.
- to implement two of the evaluated protocols[1] in order to assess their practical security properties and runtime performance indicators.
- to develop an in-depth experimental assessment study of the above protocols by simulating their behavior in large scale WSN settings (up to 500 nodes) evaluating the runtime operation and performance indicators, according to the following criteria:
  - impact in the provided routing support for tree-based data dissemination models in randomized non-supervised topologies: energy cost, connectivity, effective reliability and latency conditions.
  - impact in the topology and ad-hoc organization, evaluating the following indicators (by capturing from the simulation environment the average, maximum and minimum values), correlating the observations with the following aspects:
    * fanout metrics : number of nodes selected to disseminate events at each routing step in order to retransmit information (correlating the observation in terms of trade-offs between desired reliability level and multi-path redundancy level)
    * number of maximum rounds in event retransmission to achieve a certain reliability degree
    * degree-distribution metrics: the number of node neighbors in the physical range of each node
    * average shorted path metrics, correlating the observation with the latency conditions
    * clustering coefficient metrics: for the average of the number of links (physical range) connecting nodes to neighbors divided by the number of links between those neighbors;

[1] The evaluate protocols will be Clean-Slate[49], INSENS[21] and sigf[64]

    * number of hops in the established multi-hop routes;
    * number of resilient multi-path and multi-hop routes;

- to propose and implement a new intrusion tolerant secure routing protocol, hereon called MINSENS, integrating a pro-active intrusion tolerant mechanism based on the auto-organization of nodes interconnected by disjunct multi-path routes to multiple base stations or gateways (acting as sync nodes). This establishes an ad-hoc mesh-network, in which, sync nodes allow the consensus of data-sets received from the multiple routes, establishing ad-hoc forms of consensus. In this direction the dissertation explores the adaptation of intrusion tolerant consensus strategies with randomized or probabilistic distributed light-weight consensus solutions, as an intrinsic service of WSNs for tree-based secure data dissemination. The approach favors the adoption of LCP (local coin protocol) schemes implemented with light-weight cryptography [12], as the base for probabilistic consensus mechanisms[46], given the expected complexity and inadequacy of SCP (share-coin protocol) mechanisms, requiring asymmetric cryptography[13].

- to evaluate the proposed solution under the planned attack models, to investigate the resilience conditions and the impact on the performance indicators discussed above.In this direction, the dissertation contributes with an experimental simulation environment to the study of dependability solutions provided by a possible underlaying routing layer resilient to byzantine failures or intrusions [8], [36], [48], in such a way that it is possible to evaluate experimental network connectivity conditions allowing for reliable and secure data dissemination in wireless sensor networks, as stated in [36].

The evaluation of the proposed intrusion tolerant routing protocol shows that certain experimental guarantees can be provided in WSNs to have, eventually, at least one safe path between nodes originating events and base stations, interconnecting non-compromised nodes.

From the above observations and evaluations, the dissertation finds an interesting research direction for complementary studies on byzantine consensus that could be supported over the secure routing layer proposed. For this proposal, a critical analysis of completely asynchronous consensus protocols inspired on randomized consensus strategies, performed by sync nodes (or by specially designed internal sensors for internal intrusion tolerant data-consensus at the WSN level), seems to be an interesting research direction.

1.4 Document structure

This document will be divided in six chapters. The rest of this document is structured in as follows:

Chapter two describes the related work considering the thesis objectives and expected contributions. First, relevant WSN characteristics, uses, and related security issues are explained, as well as the adversary model definition and three existent intrusion-tolerant secure routing

protocols. It also addresses the problematic of intrusion tolerance and the consensus problem. Finally, an introduction to the most know simulators and emulators is also made, stating which one will be used to develop this thesis objectives and why.

The third chapter describes the system and network model, giving formal definitions of the solutions used to accomplish this thesis's objectives.

The fourth chapter accounts for the implementation of the theoretical background on the system and WSN network model, giving an overview of the application of the referenced concepts.

The fifth chapter presents the achieved results during the experimental evaluation of the implemented solutions.

Lastly, the sixth chapter refers the main conclusions as well as possible research directions for future work on this thesis's subject, devised from its contributions.

2. Related Work

2.1 Security in Wireless Sensor Networks

2.1.1 Wireless Sensor Networks and its uses

The main reason why Wireless Sensor Networks(WSNs) are so popular nowadays is their immense possibilities of use. This chapter will give an inside view on WSNs, along with their large scale and security requisites.

Wireless Sensor Networks are formed by grouping a number of sensors in an area on which they can communicate among themselves. These normally used sensors can bee seen as small devices with very basic capabilities[1]. They have a cpu with limited computational power, limited memory, one or more sensors(specific circuits to sense the environment), a transceiver(usually operating on IEEE's 802.15.4[25] standard) and some form of battery.

A WSN is usually made of many sensor nodes and one or more base-stations, called the "sink" nodes. These base-stations are different from regular sensors, having more computational power, being usually connected to a unlimited source of energy. They also typically possess more communication's resources, being connected to a regular computer. These base-stations receive all the information gathered by the network delivering it to the WSN user.

Given that sensors have limited transceiving range, the communications are made in a multi-hop fashion, routing the information along the sensors into the base-station.

Based on this characteristics, WSN have the following properties: limited energy and operational time; ability to resist in hostile environments; resilience to node failures; coping with mobility; dynamic network topology; communication failures; large scale operation; unsupervised operation; scalable network.

2.1.2 Large scale scenarios requisites

Wireless Sensor Networks vary greatly in the number of nodes. As a reference, we consider a large scale network to be composed by tenths of thousands of sensors, thus covering a wide area.

To function on a large scale scenario, there are some needed requisites such as multi-hop routing. On a small scale network, communications can be made by one hop broadcasting. This solution clearly does not allow the network to scale. As such, routing must be made in a multi-hop fashion, where all nodes commit to route each others packets according to a routing protocol. In this context, fault tolerance and security are key aspects of WSNs functioning.

This thesis will focus on multi-hop wireless sensor networks given that is a fundamental requisite for a large scale network.

[1] CrossbowTechnology; www.xbow.com . Accessed in Julh/2011

2.1.3 Security Requirements

To understand what security requirements does a WSN have, one must first know and understand the nature of the possible attacks against WSNs. Attacks can be either passive or active. A passive attack is typically eavesdropping on communications with the intent to gain access to confidential information.

Active attacks try to disrupt or gain access to the routing by intentionally tampering, spoofing or fabricating messages. This type of attack can be divided itself into two types: internal and external.

External attacks and communications protection The very nature of WSNs wireless communication makes them more vulnerable attacks on communications such as eavesdropping, replaying, spoofing, or even trying to prevent communications by launching a Denial of Service attack.

External attacks can be categorized based on the class of the attacker. It can be a sensor-class or a laptop-class attacker[33], according to its capabilities. A sensor-class attacker as the same capabilities as a regular node, which means that he can only communicate with neighbour nodes, posing less danger. A laptop-class attacker has far greater power then the legit nodes. It has usually a laptop computer with a stronger transmission power and range, a more powerful cpu, more memory and more battery. As a result, it can jam or eavesdrop the entire network, when the sensor-class can only affect its neighbours.

Internal attacks and intrusion tolerance Internal attacks pose a big threat to WSNs because of they very nature of autonomous operation. Adding to this, the environment on which they operate is also unsupervised, which makes them prone to sensor hijacking by an attacker. In this case, an attacker physically captures a node and can not only access any cryptographic material on it but also alter its functioning making it act erroneously and supplying legit nodes with fake information.

Cryptography is usually used to accomplish the security requisites previously mentioned but is rendered useless in case of an intrusion, which makes this subject even more important.

As mentioned in the previous section, WSNs can operate in very different circumstances, some of which may involve sensitive information. Even if the data itself is not valuable, the work done by the network is only useful if it can deliver the gathered information. As such, the need for security in WSNs arises according to items described in literature such as the OSI X.800 recommendation [30]. For a WSN to be secure, its data must have the requisites; confidentiality, integrity, authenticity, freshness, availability and secure management.

It is important to note that these security properties must be guaranteed in the presence of internal attacks.

2.2 WSN security services

2.2.1 MAC layer protection

MAC layer protocols

Given the peculiar characteristics of the WSNs, specially the energy constraints, the regular Medium Access Control protocols are not suited for effective operation. Approaches based on TDMA, FDMA or CDMA imply low scalability on addition of new nodes or mobility, with an inefficient use of the channel given the strict division of access.

The solution for WSN is based on CSMA-CA[69], which presents better scalability, flexibility and latency. Although this, the standard CSMA specification is not a perfect match for WSN either, because it does not take in consideration energy restraints, making the nodes constantly listening on the channel. As such, special purpose protocols inspired on CSMA-CA[69] models and adapted from the spectrum PAN(personal area networks) protocol suites were designed with those constraints in mind such as 802.15.4 [25], ZIGBEE [9] which are the most commonly used ones in WSNs, or special purpose MAC variants that have been published and studied in the recent contributions from the WSN research community[52]

MAC layer attacks typology

As mentioned on the previous section, attacks can either be internal or external. This dichotomy applies on MAC layer attacks, which target the security services previously mentioned such as confidentiality, integrity, authenticity, freshness or availability of the network.

These attacks can be divided into two types, one on which the attacker follows the MAC protocol, and other where he does not.

On first type, the attacker acts as a legit member of the network. He can attack by flooding the network with maximum-size packets, creating a denial of service attack which drains battery on legit nodes and lowers available bandwidth. Another attack consists in configuring a node to work as if he was not running on battery. This makes the CSMA/CA [69] mechanism use a smaller backoff time in case of collisions, which monopolizes the access to the communication medium. Both these attacks lower the packet delivery rate [62] and raise the delivering delay.

On the second type of attacks, an attacker changes the node specification so that it does not comply with the standard CSMA/CA behaviour. By doing so, it can broadcast in the presence of other transmissions creating collisions and jamming all communications. It can also replay, alter or spoof packets(broadcasting false ACKs is an example)or even inject fabricated ones. By altering a node's specification, an intruder can attack the network according to the byzantine model.

Denial of service attacks will not be considered any further for the purpose of this thesis given that it is outside of its main subject.

With regards to the other mentioned attacks, their defense mechanisms are based on the use of specific cryptography techniques suited to the restraints existent on WSNs.These defense

mechanisms are of no use in the presence of internal attacks.

2.2.2 Network layer and routing protection

The network layer in WSN is closely related with routing and data dissemination models as well as network organization including ad-hoc self organized topologies.

Most of the routing protocols designed for WSN assume that all nodes behave correctly. As such, they are not adapted to communication-failure conditions as well as nodes who act maliciously in order to disrupt the network.As previously stated, attacks on the network layer can also be divided as external or internal, being passive or active. External attacks can be made by two different types of attackers, laptop-class or sensor-class ones [33].

At network layer level, attacks always target two different dimensions:

- to control network topology(by influencing self-organization or discovery processes).
- to influence routing mechanism.

The second type of attacks can be triggered on three typical phases of a routing process: route discovery, route selection or after establishment of the routing path. [33]

2.2.2.1 Attacks topology on Route Discovery

- Fake routing information: A simple attack on the route discovery phase is to advertise fake routing information, invalidating the forwarding tables of the other nodes. This attack can be made on table-driven and on-demand routing protocols but its easier on on-demand ones given that the malicious node is informed of all route-request messages and can fabricate a malicious response or even drop the requests, preventing other nodes from participating. This attack both needed and the first step to execute sinkhole/blackhole attacks.
- Rushing attacks: A rush attack [67] in mainly targeted at on-demand routing protocols. When a node wants to communicate, it broadcasts a route-request message(RREQ). To limit flooding, each node only forwards the first copy of the RREQ it receives. When the attacker receives the original RREQ message, it fabricates a new one and tries to deliver it to the other nodes before they receive the first copy of the original ones. This will make them discard the correct RREQ messages, making route discovery impossible or putting the attacker on the route.
- Route Request Flood attacks: Also targeted to on-demand routing protocols, it aims to do a DoS attack by flooding the network with fake RREQ messages, consuming energy and memory on other nodes and preventing valid RREQ from being processed given the overflow of RREQ messages.

2.2.2.2 Attacks topology on Route Selection

- HELLO Flood attack: Some routing protocols depend on HELLO messages so that nodes may discover their neighbors. By receiving a HELLO message, a node concludes that the sender is at a one-hop distance. This may not be truth if there is a laptop-class attacker. He can transmit to a greater distance, and with that make a large number of nodes think that their are their neighbors. A HELLO Flood attack [33] exploits this possibility to make himself on the majority of routes and to disrupt communications on some nodes given that they have a shorter radio range and as such their messages will not reach the attacker.This attack is also needed and the first step to execute sinkhole/blackhole attacks.

- Sinkhole attack: A sinkhole attack [33] consists on manipulating the routing mechanism so that the attacker advertises shorter or high quality routes to fool its neighbors to route all their traffic through him. Contrary to the HELLO flood attack, this one is made using a regular antenna and only affects the attacker's direct neighbors.

- Wormhole attack: This attack is made by two or more malicious nodes that collude to attack the network. its made using a out-of-range communication channel to exchange information at higher bandwidth and lower latency, making the malicious nodes at one-hop distance. This makes their neighbors choose them to route information. A well placed WormHole attack [66] can even change the topology of the network, if for example one of the malicious nodes is next to the sink node and the other very far out. The neighbors of the far out one will think they are close to the sink node, changing their routing premisses. On the extreme, a Wormhole attack can attract the majority of the network traffic, allowing the malicious nodes to act on it.

- Sybil attack: The Sybil attack [31] is made by making a node announce many identities to their neighbors. This not only raises the chance of the malicious node being included in a route path, as it disturbs the effectiveness of fault-tolerant multi-path routing because the fake virtual nodes are treated as different by the routing mechanism.

2.2.2.3 Attacks topology after establishing of routing paths

- BlackHole attack: A Blackhole attack [29] aims to disrupt communication by not forwarding messages that are intended to other nodes. This attack is easily spotted and defended if the attacker drops all messages, because it will appear to its neighbors as if the node runned out of battery. If on the other hand the attacker selectively forwards the messages, then the attack will be much harder to detect and to defend against.

- Spam attack: A Spam attack [54] consists on a flood of unwanted and useless messages to the network. These messages drain energy from the forwarding nodes, as well as consume available bandwidth. This type of attack is specially dangerous because usually a WSN works by gathering data and sending it to the sink node. If an attacker plays a Spam attack

on the nodes near the sink node and wastes their batteries, the whole network will loose connection to the sink node and as such become useless.

2.2.2.4 Protection against external attacks

According to the previously stated attacks, there are many possible counter-measures to defend them. RREQ Floods and fake routing information can be dealt with by using authentication on RREQ packets.

Rushing attacks on the other hand, can be defended by randomizing the forwarding of RREQ packets[67]. Waiting for all RREQ packets to arrive and forwarding a random one makes the rush attack pointless, given that the neighbours will still wait for all the packets and forward a random one.

Hello floods are possible to defend by simply testing bi-directionality of the communication. A node will only accept other as its neighbour if his messages arrive at the neighbour.

To handle sinkhole attacks there are three typical techniques. The first one is to analyze the sequence numbers on the packets and suspect those who are either unusually large or not strictly increasing. This will give away attackers who whose very large sequence numbers to force the other nodes to update their routing tables. The second method is to verify if the source address on the RREQs matches the sender's identity. This can me made by the sender's neighbours. In this case, a lower ratio of verified RREQ packets in the overall network indicates the presence of an attacker. Last, the third technique is based on having each node verifying their routing caches for a node that is part of the majority of the routes. If this happens, it means that this node is a potential sinkhole attacker.

A Wormhole attack is made using a out-of-range link between two or more malicious nodes. This link has usually a wider range and more bandwidth, but cannot outrun the laws of physics. As such, defense against wormholes comes in the form of a leash[66] on the packets. On a network with tightly time synchronization, a node can insert its current time on the transmission of the packet and therefore enable the receiver to verify within a certain margin of error the transmission time. Given that wormholes are usually wide range links, the latency of the transmission enables the legit nodes to detect the wormhole .

Sybil attacks can be dealt with using one of two methods; radio resource testing and random key pre-distribution [31]. Radio resource testing is based works by assigning different radio channels to each neighbours. Because each physical node can only transmit on a single channel at a time, a node can assign each neighbour with a channel to broadcast a message and then randomly choose one channel to listen. If it receives a message, the neighbour is real. Otherwise is treated like a fake node.

Blackhole attacks [53] can bi mitigated by the combined use of a watchdog and path-rater schemes. A watch-dog scheme is based on the idea that a node can hear its next-hop neighbour broadcast to the following hop. As such, the watchdog can maintain a counter to record the misbehavior of its neighbours and use a threshold to consider a neighbour as a malicious node. The path-rater scheme uses this information to help the source-node select the best route by

assigning a non-negative rating to every normal node and a highly negative rating to each malicious node. By calculating the average rating of each route, it will return the highest rated one, excluding the ones with possible malicious nodes. Besides this technique, it is also possible to defend blackhole attacks using a mechanism of acknowledgements and fault announcements together with timeouts on each message. The idea is that the source node sets the course a message must traverse, and the destination must send and ACK to the reverse path. Each node from the source to the destination sets a timeout during which it expects to receive the ACK. If the timeout expires, it sends a fault announcement back to the source. All data, ACKs and fault announcements are authenticated. This mechanism allows the source to have feedback on the delivery and exclude routes that contain blackhole attackers.

In regards to SPAM attacks, they can be defended by a detect and defend spam(DADS) scheme [54]. Control is made by the sink node, who must supervise the content, frequency of arrival and generation rate of the received messages. If the sink node suspects of a spam attack, it broadcasts a warning message to the network. By receiving this message, each node sets a timer before which it will not relay unauthenticated messages. By doing so, spam messages will not be forwarded by the attacker's neighbours preventing the attack.

There are other counter-measures available, but those rely mostly on cryptography. Inherently, if the MAC layer is already protected then there is no need to use cryptography on the network layer.

Nevertheless, internal attacks must be considered and as a consequence, network layer counter-measures must provide methods for intrusion tolerance mechanisms.

2.2.2.5 Protection against internal attacks

At this level an internal attack means an intrusion that can induce a malicious behavior at sensor's process level. It is important to mention that an internal attack can also cause the above mentioned attack topology in a global behavior on which the mentioned protection mechanisms are ineffective. The counter-measures to defend against this typology of attacks are based either on preventive actions (preventive intrusion tolerance) or pro-active actions(pro-active intrusion resilience).

On the first type emerge solutions based on multi-path routing, which work by having multiple disjoint routes for every given destination, hoping that the at least on is not affected by the attacker.

On the second type of solutions we have Byzantine fault-tolerant routing mechanisms. When a node acts maliciously it can induce a Byzantine attack model. This subject is analyzed in depth in chapter 2.5.3

2.2.3 WSN Security overview

On the above sections in this chapter we made an analyses on WSNs as well as their security services.

On the first subject there were explained WSNs main characteristics, uses as well as large scale scenarios and security requirements. On the security requirements it was pointed out the difference between external and internal attacks.

The second subject referenced the security services at two different levels; MAC layer and network layer. On this topics we introduced attacks topology as well as their counter-measures. This subject also mentioned other security components such as the key establishment schemes and security associations as well as secure data aggregations.

This thesis's objectives are to augment the security at network layer by introducing a fault-tolerant consensus mechanism capable of dealing with intrusions.

This objective is closely tied to the security services at network layer discussed at section 2.2.

2.3 Communication Security and typical approaches

There are several protocols which aim to ensure protection against the attacks at MAC layer mentioned in the previous chapter .

SNEP [5] was one of the first secure link-layer protocols to be developed as part of the SPINS protocol suite. It manages to get a low energy consumption by keeping a consistent counter between the sender and the receiver, sparing the transmission of the IV on each packet. If there is packet loss, SNEP needs to resynchronize the counters which is a slow a energy expensive protocol.

TinySec [18] is another secure link-layer protocol, which accomplishes low energy consumption by reusing part of the packet header to send the IV. The drawbacks are that TinySec uses a single network-wide key (making him vulnerable to a single node exposure) and does not provide protection against replay attacks.

ZigBee [6] is another secure link-layer protocol, very similar to SNEP. The big difference is that it passes the 8 byte counter on the clear instead of keeping the state on the two principals.

MiniSec[44] is an evolution of the SNEP protocol with various improvements to lower energy consumption, such as mechanism to try to guess the IV in case of packet loss, lowering the probability of a resynchronization.

From all the these mentioned secure link-layer protocols, MiniSec is considered to be the best given that it provides authentication, data secrecy as well as replay protection, and it is the most energy efficient one.

Multi-hop communications are protected by the use of this types of protocols as long as there are no intrusions, given that all this security services rely on cryptography.

All the above solutions for WSN communication's security relate with secure link-layer protocols above described considering a Dolev and Yao[22] adversary model, adopting as base security mechanisms low cost cryptographic systems(symmetric cryptography, MACMessage Authentication Codes) and secure hash functions). The Dolev and Yao[22] model states that an attacker can only aim to attack communications, restricting it to external attacks. Therefore, the

above solutions do not consider external attacks such as those previously described in 2.2.2.4.

2.4 Routing Security

2.4.1 Intrusion tolerance routing

2.4.1.1 INSENS

INSENS [21] is a secure routing protocol that exploits redundancy to tolerate intrusions without needing to detect them. One of the main design concerns was to prevent a single node to disrupt a big part of the network. As such, INSENS aims to provide protection against two types of attack: DoS that flood data packets to the entire network and routing attacks that propagate erroneous data.

This protocol takes advantage of the asymmetric architecture common on WSNs. The base station node is usually more powerful and less resource constrained, and in virtue of this it performs all heavy-duty computation in order to free the other nodes from building routing tables or dealing with intrusions. It shares a symmetric key with every sensor so that it can securely communicate with him.

To prevent a single malicious node from affecting the entire network, communications are constrained. Broadcasts can only be made by the base station and are authenticated to prevent tampering [5]. Unicast communications are made through the base station so that it can filter possible DoS attacks towards a single node.

Because this protocol is based on surviving in the presence of intrusions rather then detecting and trying to eliminate them, INSENS uses redundant multi-path routing to bypass compromised nodes. The paths used to accomplish this are as disjoint as possible, typically sharing only sender and destination nodes.

The protocol works in three rounds:

- Route Request: the base station floods a request message to all nodes. Each node discovers its neighbors by receiving a message from them.

- Route Feedback: the nodes reply to the base-station their local topology information using a feedback message that traverses the reverse-path on which the request message arrived to the node.

- Routing Table Propagation: the base station computes all the forwarding tables for each sensor ad sends them to the respective nodes using a routing update message.

The key idea behind INSENS is that although a malicious node may be able to affect its one-hop neighbors, it cannot cause a wide-spread damage to the network.

2.4.1.2 Clean-Slate

The Clean-Slate routing protocol [49] was designed baring in mind three main directions for secure routing protocols: prevention, detection/recovery and resilience.

The protocols works by having a network authority that gives to each node a certified id along with a set of randomly chosen challenges. In the beginning, it takes place a secure neighbour discovery protocol on which each node announces himself to is one-hop neighbours. This mechanism prevents the insertion of new malicious nodes at later phases of the protocol.

After the establishment of neighbours, it starts a recursive grouping algorithm that uses a reliable broadcast service to deliver its messages. This algorithm is deterministic for any given network topology, and forms the routing tables as well as assigns a unique network address to each node. By using this recursive grouping algorithm, the Clean-Slate protocol ensures dynamically established routing tables and network addresses.

In terms of security, this protocol uses resilient routing techniques to transmit messages(based on multi-path routing) as well as mechanisms to detect and eliminate malicious nodes. This later mechanisms are based on Group verification trees to prevent multiple ids from a node as well as a honeybee attack to eliminate malicious nodes.Specifically, the protocols protects against the following type of attacks:

- fake routing information by secure neighbor discovery along with recursive grouping information.
- Sybil attacks by the use of secure neighbor discovery.
- Blackhole attacks because routes are not based on advertised distances.
- Wormholes and localized jamming/DoS by being able to route around possible attackers given the multi-path routing.

The Clean-Slate protocol was designed for low or inexistent mobility.

2.4.1.3 SIGF

The Secure Implicit Geographic Forwarding(SIGF)[64] are a family of three protocols, SIGF0, SIGF1 and SIGF2, based on the IGF[11] routing protocol, a stateless non deterministic network/MAC layer hybrid routing protocol.

The idea behind the SIGF is to adequate the security mechanisms to the security requirements of the environment. As such, each of the protocols adds more security than the previous, each making the basis for the next. SIGF-0 keeps o state, and offers only probabilistic defenses. SIGF-1 add to the SIGF-0 the use of local history and neighbor reputation. SIGF-2, the most secure one, has everything that SIGF-1 has, and it adds neighborhood-shared state.

The protocols provide a explicit tradeoff between security and state maintenance, allowing configurability that can be adapted at runtime.

The main purpose is to have minimal active secure protection, obtained by choosing the correct SIGF protocol according to the security requirements of the deployment.

2.5 Intrusion tolerance and distributed consensus

2.5.1 Consensus problem in distributed systems

Consensus is a classical problem in distributed systems. A consensus protocol enables a set of processes to agree on a value. Each one starts with a value that proposes to other, and the consensus problem consists in designing a protocol where all correct processes unanimously decide on a common output value that is one of the inputed ones. The consensus problem comprises four properties:

- termination: every correct process eventually decides on a value.
- integrity: a process decides at most once
- agreement: no two correct processes decide differently
- validity: a process can only decide a value that was previously proposed.

The solution to this problem in the absence of failures is quite simple and exists in literature[19]. However, in the presence of failures, the solutions to this problem depend on the assumptions about the system model[23]. There are two types of faults to consider:

- fail-stop failures, where a process operates correctly and suddenly crashes, ceasing all actions.
- byzantine[40] failures where no assumptions are made about the behavior of the process. It can send messages when it is not supposed to, send only part of the messages, send messages with erroneous data, or even not respond for a period of time and respond again soon after.

A key aspect is whether or not the failure of a node to send an expected message can be detected by other nodes. If so, the receiver gains the important piece of knowledge that the sender is faulty. This can only be either in a system where there are clocks and time-bounds[39] or in a synchronous model where the processes run in lock step and messages sent in one step are received in the next. However, this detection is impossible in a fully asynchronous system model where there is no way to tell the difference between a crashed node and a node operating very slowly. Such system models are bound to an impossibility result, the FLP impossibility result[24].This impossibility states that consensus is impossible to resolve by deterministic protocols in asynchronous systems if even one node can fail.

2.5.2 Santoro & Widmayer impossibility

As stated above, the asynchronous systems are bound to the FLP[24] impossibility. There is a similar impossibility for synchronous systems where communications are unreliable. The Santoro & Widmayer impossibility[55] states that even with strong synchrony assumptions, there is no deterministic solution capable to solve any non-trivial form of agreement if n-1 or more messages can be lost per communication round in a system with n processes.

2.5.3 Consensus problem on WSN

The thesis will explore probabilistic consensus techniques to address the design and implementation of intrusion-detection mechanisms to be used as a complementary strategy to other preventive intrusion mechanisms. These mechanisms will allow for the reconfiguration of the network topology, routing reorganization and establishment and maintenance of new routes, when an intruder is detected in the path or current established routes.

In this direction, efficient and optimal consensus protocols must be devised, adapted for the characteristics of multi-hop WSNs. The problem must be addressed from both a theoretical and practical perspective, and in the context of the thesis will be centered in the adoption of randomized or probabilistic forms of distributed consensus.

When surveying existent approaches, we looked for three main characteristics of possible approaches: first, we looked for intrusion-tolerant protocols and mechanisms, second, we considered protocols to be executed in decentralized (or leader-free) scenarios and third, we focused our attention to protocols designed for asynchronous computation models. Given the inherent unreliability of WSNs, the independence of any kind of timing assumptions would ensure correctness properties, considering the unpredictable timing behavior.

Of course that any existent protocol is subject to the aforementioned FLP impossibility result, and limited by the Santoro and Widemayer deterministic consensus hypothesis. This means that the inspiration from previous approaches must be adapted in order to circumvent that impossibility. In randomization protocols there are certain steps in which a current value proposed by a principal can take a random value, chosen according with a probability distribution. This means that an adversary scheduler cannot determine the outcome of any disruptive strategy, because the outcome is only affected by possible random elements out of control. At the same time these solutions are completely asynchronous. To achieve this form of consensus, one has to judiciously weaken the problem statement for usual consensus properties, allowing a probabilistic termination property as an alternative to the deterministic termination condition. Thus, the properties for probabilistic consensus inherently approached for the objectives of the thesis can be stated by the conjugation of the following properties:

Validity: if every sensor node proposes on a same value v, then all the correct nodes that decide, will decide the value v;

Agreement: no two correct sensors decide differently;

Termination: all the correct sensors eventually decide, with probability p=1.

The touch-stone in the devised consensus technique to be applied to implement the intrusion-detection mechanism is based in these properties, particularly by exploring optimized solutions associated to the termination property, evaluating the impact in an implementation addressed for WSNs.

2.5.4 Probabilistic consensus solutions

The randomized consensus schemes devised in the thesis are based on so-called coin tossing cryptographic schemes. Depending on the implementation, these schemes are based on local node computations and a distributed protocol. The idea is that the local component delivers "0" or "1" (or a vector of such values). In general is possible to deliver a set of values chosen in a larger domain. The protocols are primarily categorized in one of two classifications [56]: local coin tossing operation (or LCP) and shared coin operation (or SCP).

On LCP schemes the operation is performed independently by each node, with the final decision based on autonomous conditions and ligh-weight cryptography, after a round of messages interchanged between the participants in the consensus. SCP schemes involve a distributed coordination process, using in general heavy-weigh threshold asymmetric cryptographic primitives (or m,n threshold asymmetric methods), warranting that the consensus is achieved if only a subset m of the n participants finalize the protocol. minimize the interactions.

Both classes of protocols were introduced independently in 1983 by Ben-Orr [10] (for the case of LCP schemes) and Rabin for the case of SCP. Ben Orr presented two algorithms, tolerating byzantine process failures, tolerating respectively crash-failures and arbitrary failures. The crash scheme tolerates f faulty processes out of n>= 2f +1, when n participants want to establish the consensus. The byzantine scheme tolerated f faulty processes out of $n >= 5f + 1$.

Both protocols terminate in an expected exponential number of rounds. Rabin proposed a scheme tolerating f faulty processes out of n >= 10f +1, with the advantage that the agreement is terminated in a constant number of rounds. In the years that followed, other randomized protocols were proposed and implemented, following the base approach of Ben-Orr[12] and Rabin [13]. A detailed survey on these techniques is available in [17].

Among the several goals pursued we emphasize the quests for achieving optimal resilience in the number of byzantine processes involved, which is directly related with the possible application of these schemes to large scale and dense WSNs. Protocols with optimal resilience (tolerating f faulty nodes in networks with n>= 3f +1 nodes) were presented quite early, both using LCP schemes [12] and SCP models. Constant expected round complexity with optimal resilience was, for long, an objective in the research., with the primary approach provided by [13]. The ABBA protocol [13] also terminates in a small number of rounds and sends a much lower number of messages than the initial solutions. It does so, however, at the cost of heavy-weight asymmetric threshold cryptographic primitives. Compared with this approach,

the BRACHA protocol [12] requires pre-distribution of data among the involved processes for each coin-tossing agreement, something that can be quite difficult in certain applications, but i tis not necessarily the case in the devised solution in the context of the thesis. Anyway, this requirement can be removed, taking more recent techniques that don't need the previous distribution of data, seeds or secrets [13].

2.5.5 Local Coin Protocol

The Bracha's local coin protocol [12] exchanges $O(n^3)$ point to point messages per round, with an $2^(n-f)$ expected number of rounds until termination assuming a strong adversary model[2].

This algorithm does not use any type of cryptographic operations but assumes reliable communication channels, which inherently use lightweight cryptographic hash functions.

2.5.6 Shared Coin Protocol

The ABBA shared coin protocol[13] exchanges $O(n^2)$ point to point messages per round and reaches a decision in one or two rounds with high probability.

It makes extensive use of asymmetric cryptography to ensure correct execution and it assures integrity of messages through the use of public-key signatures. It also assumes the use of reliable channels and two cryptographic primitives: dual-threshold signature and dual-threshold coin-tossing scheme.

An (n,k,f) dual-threshold signature scheme is a technique where a set of shares of signatures are generated for n processes, and k of such shares and both necessary and sufficient to assemble a valid signature even if up to f processes are corrupt. This scheme can be implemented using a vector of RSA signatures[13]

An (n,k,f) dual-threshold coin-tossing scheme is similar, but each process holds shares of an unpredictable function F that maps the coin name C to a binary value F(C) $\in 0, 1$. Similarly, k of those shares are both sufficient and necessary to assemble the function F. An implementation is the Diffie-Hellman based solution of Cachin et al[13].

2.5.7 Critical analysis: LCP vs SCP

As seen in the above description of the local and shared coin-tossing mechanisms, local coin ones use lightweight or no cryptography thus being lighter in the computational aspect, but use more rounds of message requiring more communications. Shared coin mechanisms use public key cryptography and require less message rounds. As such, theorycally, choosing one of this types of mechanisms is a trade-off between computation cost and transmission cost. The

[2] A strong adversary model assumes that the attacker controls the network scheduling

results experimental results on regular computers in a LAN setting found in [46] show some interesting results:

- The LCP is significantly faster than SCP on the tested environment
- The SCP showed bigger scalability on the number of involved processes
- The SCP behaves better in the presence of malicious faults comparing to the LCP.
- The average number of rounds was quite small in both protocols, with f malicious processes the SCP performed similar to the failure-free scenario, and the LCP showed a small degradation.
- When raising the number of simultaneous consensus, the bottleneck in LCP was the network because of the high number of exchanged messages. In the SCP the bottleneck was the CPU given the use of expensive asymmetric cryptography.

Considering that in WSN transmissions are much more costly in terms of energy than processing, SCP would seem the best choice. Unfortunately,the use of asymmetric cryptography is impossible on the sensors given their computational restraints, which makes the LCPs the ones to use on the development of this thesis.

2.6 Simulation environments and platforms

Wireless Sensor networks requirements are getting more and more demanding. As such, the need for reliable and dependable implementations grows making network simulation tools more important.

Application software can be previously tested by running in emulated PC-environments before being used the physical nodes. This facilitates testing and debugging of the programs.

Sensor network emulator and simulation can be defined as:

- Emulator: can usually run the same code that runs in a sensor node and emulate the operation of hardware in the sensor node, e.g., processor, sensors, and radio.
- Simulation: : A technique where the properties of an existing, planned and/or non-ideal network are simulated in order to assess performance, predict the impact of change, or otherwise optimize technology decision-making.

Next, we will make a review on the most common simulation and emulation tools, analyzing their properties and suitability as the simulation platform for implementing and experimentally accessing of the contributions planned for this thesis.

2.6.1 TOSSIM/TinyOS

TinyOS is an open-source operating system designed for wireless embedded sensor networks [3]. TOSSIM developed by the University of Berkeley, is targeted to emulate/simulate TinyOS equipped sensor nodes (especially Berkeley MICAz motes) [4]. The emulation tool is currently included in TinyOS release. TOSSIM compiles the same source code for simulation that runs in a sensor node and emulates a limited type of hardware like ADC-converters, sensors and IO.

Some drawbacks in the simulator considering emulation are:

- Every sensor node must run same code in simulation.
- Sensor node application code execute time is assumed zero in simulations.
- Overlapped interrupts are not registered
- Radio modeling is very simple (probabilistic bit error)
- Compiles TinyOS applications for TOSSIM platform, not for motes
- Some emulated code might not behave equally in sensor node because of the simplification assumptions in the emulator.

2.6.2 PowerTOSSIM

PowerTOSSIM is an extension to TOSSIM developed by Harvard University, which adds power estimation feature[5] . It estimates the power consumption of target nodes through logging the operations both in TOSSIM modules and in the application code. In PowerTOSSIM, the electric current of each type of hardware peripheral, such as the CPU, radio, sensor, EEPROM, and LEDs, is measured separately. Subsequently, PowerTOSSIM uses the measured electric currents as its experimental power model.

2.6.3 Omnet++

The "Objective Modular Network Testbed in C++" is an object-oriented modular discrete event simulator. Some extensions for WSN exist, e.g. NesCT converter that allows limited TinyOS (Berkeley motes) simulation/emulation by converting NesCT source to simulator supported C++ classes. NesC code interchange between the simulator and a sensor platform compiler is restricted, because the protocol and the hardware implementation in the simulator are

[3]TinyOS homepage: www.tinyos.net, accessed Jan/2011

[4]TOSSIM homepage: www.eecs.berkeley.edu/ pal/research/TOSSIM.html, accessed Jan/2011

[5]PowerTOSSIM: Efficient Power Simulation for TinyOS Applications: www.eecs.harvard.edu/ shnayder/p-TOSSIM/, access Jan/2011

simplified and not all hardware is supported [35].

Comparison with TOSSIM [6]

- Simulation is faster than in TOSSIM
- better debugging capabilities than in TOSSIM
- NesCT enables coding in NesC with TinyOS components and still makes use of functionality from OMNeT++.
- Coding possibility for the actual hardware.
- Software testing with real hardware is possible with small modifications.

There are some drawbacks:

- NesCT compiler is compatible only with TinyOs 1.1x (current release 2.0)[7]
- Due to simplifying assumptions, simulated code requires modifications before it produces same results with a real sensor node.

2.6.4 Atemu

Atemu is a software emulator for AVR-microcontroller based systems [8]. Atemu takes a compiled binary file as an input and includes complete emulation of AVR instruction set. The emulator is able to run TinyOS based binary code and it supports partially all MICA2 board components including the radio. Also different sensors nodes can run different programs. Some features of Atemu are:

- complete emulation of the AVR instruction set
- partial support for all MICA2 board components
- loading of ELF executables and Motorola SREC images
- support for multiple sensor nodes in a sensor network
- configurable and modular hardware support
- ability to run TinyOS based code
- different sensor nodes can run different programs.

[6] Omnet++ homepage: www.omnetpp.org/pmwiki/index.php?n=Main.NesCT, accessed Jan/2011
[7] Omnet++ homepage: www.omnetpp.org/pmwiki/index.php?n=Main.NesCT, accessed Jan/2011
[8] Atemu homepage:www.cshcn.umd.edu/research/atemu/, access Jan/2011

The main drawbacks are:

- very limited number of supported sensor node hardware
- very simple radio propagation model (in the air, line of sight connection)

2.6.5 Freemote

Freemote is a lightweight and distributed Java based emulation tool that is utilized for developing Wireless sensor network software. The emulator supports emerging java based motes on optimized JVM (Squawk, Sentilla Point) and platforms (Java cards, SunSPOT)[9].

The emulator splits the Software architecture of a Mote in three independent layers connected through well defined interfaces (Application, Routing and Data Link and Physical). Routing and application layer codes are interchangeable. The real nodes can be any devices based on the IEEE802.15.4[25] standard communication interface (i.e., MICAz, JMotes, Tmote Sky).

its main drawbacks are:

- radio path modeling is currently very simple assuming no obstacles between sensor nodes
- realistic communication behavior emulation is possible, but it is still limited to just simple emulation scenarios and dedicated platforms (JMote).
- it is not targeted to performance analysis, which is important in algorithm development.

2.6.6 Avrora

The AVR Simulation and Analysis FrameworkPlatform is a research project of the UCLA Compilers group[10] . It is utilized as a set of simulation and analysis tools for programs written for the AVR micro-controller produced by Atmel and the Mica2 sensor nodes. Avrora provides nearly complete implementation of the mica2 hardware platform, including almost complete ATMega128L implementation, and an implementation of the CC1000 AM radio. Some features of Avrora include:

- the simulator can test programs with accurate duty cycle execution times
- it is possible to analyze energy consumption
- possibility for cross compilation of source code directly to sensor node architecture, where it can be executed on an emulated processor
- protocol development in a simulation environment is possible without any modifications in the code and with the ease of mind that the protocol will operate in a physical test-bed.

[9]Freemote homepage: www.assembla.com/wiki/show/freemote, access Jan/2011

[10]Avrora homepage: http://compilers.cs.ucla.edu/avrora/sensors.html, accessed in Jan/2011

The main Avrora's drawback in its intended operation area is its inability to model clock drift. Clock drift is a phenomenon where nodes may run at slightly different clock frequencies over time due to manufacturing tolerances, temperature, and battery performance [1].

2.6.7 AvroraZ

AvroraZ is a research project of the AWS center. It is an extension of the Avrora emulator (see above), which allows the emulation of the Atmel AVR microcontroller based sensor node platforms with IEEE 802.15.4 compliant radio, e.g. Crossbow's MicaZ[11]. AvroraZ is based on design, implementation and verification of several extensions to Avrora: the address recognition algorithm, an indoor radio model, the clear channel assessment (CCA) and link quality indicator (LQI) of the IEEE 802.15.4 standard [20].

The motivation of this implementation is to enable precise emulation of IEEE 802.15.4 based protocols without any modifications in the code developed for the real hardware. Currently, the tool is being tested and evaluated using the implementation of beacon-enabled mode of the IEEE 802.15.4 protocol stack developed in nesC, under the TinyOS operating system for the CrossBow MICAz motes called Open-zb as well as new add-ons.

2.6.8 VMNET

VMNET aims at realistic performance evaluation for WSN applications and simulates target WSN as a VMN (Virtual Mote Network) [65]. The simulator emulates the CPU of a mote at the CPU clock cycle level, and includes sufficient detail emulation for the sensing units and other hardware peripherals. The radio signal transmission is emulated by the communication between VMs with the effects of signal loss and noise. VMNet enables parameterization from the real world and keeps a detailed log about the running status of application code. This allows the binary code of the target WSN application to be run directly on the VMN, and the application performance, both in response time and in power consumption, to be reported realistically in VMNet.[65]. The advantages of VMNet are:

- its high precision and fidelity
- possibility to do energy consumption simulation
- compilation of the source code directly for sensor nodes.

On the other hand, the disadvantages are:

- lowered simulation speed because of precision and fidelity (10 times slower than TOSSIM)
- simple and basic radio propagation models, that are implemented with predefined functions (Gaussian noise + attenuation)

[11] AvroraZ homepage: http://citavroraz.sourceforge.net/, access on Jan/2011

- on energy simulation side, VMNet does not instrument application code.

2.6.9 NS2 (and NS3)

NS2 (and NS3) is a commonly used discrete simulator that models a network as multiple layers[12] . The N̈etwork Simulator 2ïs a discrete event simulator targeted at network research. It is probably the most prominent network simulator that supports WSN simulation. It includes a huge number of protocols, trafic generators and tools to simulate TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. Its main focus is the ISO/OSI model simulation, including phenomena on the physical layer and energy consumption models.

For wireless sensor networks it includes sensing channels, sensor models, battery models, lightweight protocol stacks for wireless micro sensors, hybrid simulation support and scenario generation tools. The main disadvantage of NS-2 is the highly detailed packet level simulation, making it virtually impossible to simulate really large networks. In principle, Ns-2 is capable of handling up to 16,000 nodes, but the level of detail of its simulations leads to a runtime that makes it hopeless to deal with more than 1,000 nodes[37].

2.6.10 SENSE

SENSE is a simulator specially developed for the simulation of sensor networks[13] . It offers different battery models as well as simple network and application layers. Radio implementation is limited to IEEE 802.11. In its current version, SENSE comes with a sequential simulation engine that can cope with around 5,000 nodes, but depending on the communication pattern of the network this number may drop to 500.

SENSE was developed, because NS-2 and many other simulators introduce unnecessary interdependency between the components. This will make extension of simulation very difficult (e.g. introducing a new protocol to the simulation).

2.6.11 J-Sim

J-Sim (formerly known as JavaSim) is a component-based, compositional simulation environment[14] . It was not developed for WSN simulation like SENSE, but the reason for its development was the same: enable extensibility. J-sim is a widely used simulator that employs layered network models. However, these simulators with layered network models are unsuitable for performance evaluation of WSNs because the performance of WSNs is affected by the hardware, the OS, the networking protocols, the application code, as well as cross-layer optimization techniques. Despite of this drawback, J-Sim can be important simulation tool because of its loosely-coupled component model, which enables broad hardware simulation/emulation

[12]NS-2 homepage: http://www.isi.edu/nsnam/ns/, access on Jan/2011

[13]Sense homepage: http://www.ita.cs.rpi.edu/sense/index.html, access on Jan/2011

[14]J-Sim (JavaSim) homepage: http://sites.google.com/site/jsimofficial/, access on Jan/2011

(with constraints) and easy and fast software side prototyping.

2.6.12 EmStar

EmStar is a software environment for developing and deploying microserver grade sensor nodes (microserver is a Linux platform) [15]. It can act as a pure simulation tool but also provides an interface to real low-power radios [27]. EmStar provides powerful tools to test and debug application programs. However, it does not perform CPU emulation and therefore cannot focus on realistic performance issues. Also the TinyOS and application code is compiled for PC and EmStar does not run it directly in the motes. EmStar allows trade-offs between fidelity and simulation speed by simulating at different levels (e.g., bit or packet level).

2.6.13 JProwler

JProwler[16] is a discrete event simulator for developing and assessing communication protocols for TinyOS based ad-hoc wireless networks. It supports pluggable radio models and MAC protocols, offering two implemented radio models (Gaussian and Rayleigh) as well as a MAC protocol: MICA2 without acknowledgements. This simulator is implemented n Java and optimized for speed, making a real-time simple network-wide broadcast for a 5000-node network in 1.3 seconds. The startup time for creating all data structures for this setup takes about 35 seconds and 1.5 seconds for a 1000-node network.

2.6.14 WiSeNet

Wisenet [17]is a simulator based on JProwler, adding a mechanism to simulate a typology of attacks on WSNs, along with functionalities to the interface side of the simulator. It was design having in mind the easiness to design and configure network topologies. To achieve this, it provides some interesting features:

- a graphics interface to visualize and configure the network, providing information on the state of network and each node(for example, a node's energetic state, id, or stack protocols)

- a energy model implementation that allows to assess energy consumption in normal operation and under attacks.

- a topology generation model, allowing for random, grid and structurally controlled distributions.

[15]EmStar homepage: http://cvs.cens.ucla.edu/emstar/, accessed on Jan/2011
[16]JProwler homepage: http://w3.isis.vanderbilt.edu/projects/nest/jprowler/, accessed in Jan/2011
[17]WiseNet homepage: http://code.google.com/p/secwsnsim/, accessed in Jan/2011

- a mechanism to introduce failures and attacks in the network. This mechanism allows to assess the impact of typified attacks on the implemented protocols.

- a set of utilities that gather simulation informations. in real or deferred time. These allow the extraction of measurements on network properties such as energy consumption, latency, reliability, protocol and events correction. This information is displayed in graphics which allows a better understanding of the information.

Given the above mentioned features the WiSeNet simulator arises as one of the most suited to be a part of the simulation platform for implementation and experimental assessment of the contributions planned for the dissertation.

2.7 Discussion

When analyzing security from a bottom up point of view, we can see that security services provided at the link layer level are able to cope with external attacks in one-hop and multi-hop settings, but fail to respond to intrusions. These services are based on cryptographic techniques which makes them vulnerable to intrusions where one or more nodes can be compromised.

On the network layer, most of the routing protocols are not designed to be intrusion tolerant. Albeit that, there are some protocols as the three studied on this chapter that do provide some mechanisms to cope with intrusions. They can either be based on the idea that intrusions will occur and as such the protocol must work baring that in mind, using techniques like multi-path routing to augment its resilience. The other possible approach is to try to detect failures/intrusions, and try to confine the damage of such attacks. Both approaches try to prevent the network from being totally affected, suffering a graceful degradation according to the number of attacks.

Using consensus as a security mechanism to tolerate intrusions on a WSN is bound by two impossibilities, the FLP and Santoro and Widmayer. The FLP states that no deterministic protocol can achieve non-trivial consensus on an asynchronous network. Santoro and Widemayer impossibility says that even in a fully synchronous system, no deterministic solution is possible if n-1 messages can be lost per round of communication. To circumvent these limitations, this thesis devises to use probabilistic randomized consensus protocols.

Secure data aggregation and processing protocols are another security mechanism traditionally used in WSNs given their own sensing and processing nature. These protocols are tightly coupled to the application level given that they use data semantics to operate.

The next table gives a good overview over the security provided by the services discussed above, as well as the type of security the devised solution aims to provide:

	topology		attacks	
	one hop	multi-hop	internal	external
consensus	yes	devised solution	yes	NA
routing intrusion tolerant	yes	devised solution	devised solution	yes
routing non intrusion tolerant	yes	yes	no	yes
communications security	yes	no	no	yes

3 . WSN network environment and system model

3.1 System model

3.1.1 WSN Reference settings

For the purpose of the thesis's objectives, we consider a WSN as a spatially distributed system composed by autonomous nodes (motes or sensors), monitoring events as measurements of values (typically associated to environmental or physical conditions). We consider large-scale settings in the sense that the network is composed by thousands of sensors (from 100 to 10000 as reference) forming a mesh-based network, in which, data-values pass through the network to main locations associated to capture or sync nodes, acting as base-stations. Sensors acts as values-origination nodes and as routing nodes in one multi-path route, interconnecting a value-origination node to one or more base-stations.

In our settings, the network can have more than one base-sation, interconnected by an internetworking environment, forming an independent communication behavior from the WSN itself. Inter-communication between the sensor nodes in the WSN is bi-directional, in terms of application data or control-data (or commands) enabling also to control or configure the activity of sensors.

As a practical reference mapping the above environment, the sensor nodes have limited computational resources, energy and communication range, equivalent to well-known existent sensors and their characteristics (like CrossBaw Mica-Motes, TMotes or SkyMotes, according with respective data-sheets [1][2] and a communication behavior supported by a IEEE 802.15.4 communication stack. Atop of the communication stack, a base security layer exists offering base security services, warranting security properties as stated in [5] [18] [44] with the adequate balance between computational and energy requirements. We also assume that part of the communication layer implements cryptographic key-distribution and secrecy establishment, for which, the references are stated and discussed in [28].

Also as a practical reference, base stations are special nodes, with computational resources equivalent to a medium-range portable computer and a TCP/IP stack implementing, at least, the transport layer, with standards like TCP or UDP. Base-stations can also process data and operate as gateways for the inter-connection between IEEE 802.15.4 [25] environments (WSN internal behavior) and wired or wireless LANs, typically implemented with switched IEEE 802.3 technology [2] or IEEE 802.11 standards[60]. Base-stations can use its communication facilities to implement themselves a mesh-based network in a single-hop environment supporting IP broadcasting or IP multicasting protocols.

From the thesis's objectives and viewpoints, sensors are static and homogeneous nodes, operating without human supervision and autonomous but finite energy sources (even enriched by

[1] CrossbowTechnology; www.xbow.com . Accessed in Julh/2011

[2] AvroraZ homepage: http://citavroraz.sourceforge.net/, access on Jan/2011

some energy harvesting capabilities. We are not considering mobile wireless sensor networks or VANET - vehicular sensor network settings. From the starting point of view, base-stations (BS) are also static nodes, but we will discuss during the rest of the thesis solutions that can address also the possible adoption of mobile ad-hoc base-stations. We assume that the WSN is deployed and organized as an ad-hoc network, and this can also happens with the base-station nodes, that can form a behavior in a way that we can not expect necessarily that a certain base-station "knows", in each moment, the existence of others.

According with the thesis's contributions and its approach-level, we are not considering specific application scenarios. However, we assume that applications are data-centered (transparently supported by the network-layer services proposed by the thesis), supported on communication primitves to send and receive data, in an asynchronous communication error-prone environment. In this environment, nodes are intermittently or partially connected (and/or disconnected) according with the life-cicle operation of the radio communication in a fully CSMA/CA mesh setting with no coordination.

As a practical reference, we devise families of application settings ranging from data monitoring environments of fixed targets to mobile tracking detection, but always requiring fixed monitoring points.

3.1.2 Formal system model definitions

The WSN exhibit an autonomous and homogeneous computational asynchronous environment, acting as a monitoring island interconnected by an intra-network of base-stations to external remote computational systems. These systems are not relevant for the purpose of the thesis, but we can easily imagine central-servers running specific applications, data-centers or data-storage cloud-based solutions.

Formally, the devised WSNs are distributed and self-organized systems, organized as a graph with limited and intermittent connectivity conditions. The graph has a finite set of nodes, each one running the same process. The set of nodes is unknown in each node that only knows (a priori) a very small part of the network, namely, the strict number of nodes in its radio communication range, after a first discovering phase to establish physical connections (or primary graph connections). We will call to this phase the discovering phase for stability conditions. After the termination of this phase, we consider that each node only "knows", in the connectivity graph, its physical neighbors.

Physical neighbors in the graph are only sub-sets of connectivity conditions in the graph connectivity (only assuring partial coverage conditions).

We apply the same above settings to the graph representing the interconnection of base stations.

Adversary model Processes running inside the WSN are subject to byzantine failures[40], i.e., they can deviate arbitrarily from the correct specification. From this model we associate the adversary model, as an equivalent model to the failure model, in a dependable system vision

model, where reliability and security concerns are looked together. Processes running in WSN nodes are subject to possible intrusions caused by byzantine adversaries as a complementary challenge to the base Dolev-Yao hypothesis [22] formulated for external attacks to radio communications, as stated in the OSI X.800 framework [30], complemented by the intrusion model that we describe below

3.2 Simulation Environment

In order to clearly show the architecture of the simulator, the following figure shows a stack-oriented view of its main components:

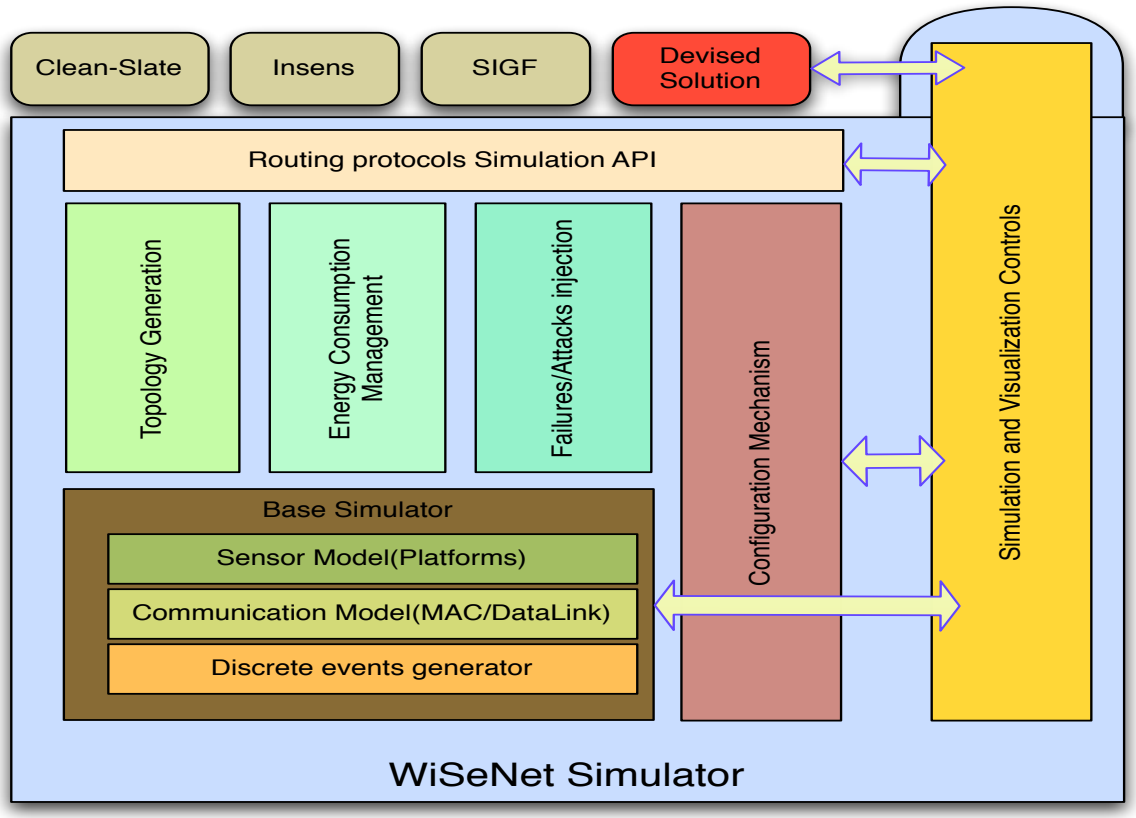

Figure 3.1 WiSeNet main components and devised solution positioning

As the show in figure 4.1, the main services provided by the simulator are: i) Base simulator or simulation engine; ii) Testing and instrumentation layers; iii) Routing protocols simulation API; iv) Simulation and visualization controls.

3.2.1 Base simulator or simulation engine

As mentioned on 3.4.14, Wisenet is based on JProwler[3] and therefore its engine is an extension to the JProwler's. This means that the simulator includes the base models of a Wireless Sensor Network possessing radio and MAC models.

3.2.2 Testing and instrumentation layers

This module contains the main mechanisms to assess the testing and instrumentation of the network setup. These mechanisms are:

3.2.2.1 Configuration mechanism

The platform's configuration is based on XML files, which are used throughout all the simulator in order to keep a flexible and yet stable configuration. Examples of this mechanism are information such as the base simulator parameters or the configuration of a given simulation which are all stored as XML files, allowing the repeating of an experiment while maintaining the its original settings and conditions.

3.2.2.2 Topology creation mechanism

One of the main aspects for the functioning of a wireless sensor network protocol is its network topology. Typically, a network's topology can either be random or structured in a grid way. As such, this component allows for the user to deploy any given amount of nodes on a specific amount of space, choosing between a grid or random topology. This mechanism, together with the configuration one allows to build and permanently store topologies which can then be later used.

3.2.2.3 Energy Module measurement mechanism

This is one of the most important mechanisms, given the scarcity of energy on wireless sensor networks. The lifetime of these networks depends on its energy consumption, which together with the security constraints makes the study of the impact of secure routing algorithms on the network lifetime a key aspect to assess. This mechanism is fully transparent to the routing layer and allows the user to define the energy consumption on each of the operations that nodes do such as a transmission, reception or a ciphering data for example. The results are shown on a chart.

[3]JProwler homepage: http://w3.isis.vanderbilt.edu/projects/nest/jprowler/, accessed in Aug/2011

3.2.2.4 Coverage/Reliability/Latency measurement mechanism

One of the main effects of an attack on a WSN is the loss of data, achieved by disrupting the normal routing behavior. This can be evaluated by measuring a network's ability to resist such an attack while still delivering its sensed data. Parameters such as coverage, reliability and latency are thus the ones used to objectively assess the network's resilience in the presence of attacks. Such assessment is only meaningful when in comparison with the network's performance in the absence of attackers.

This module allows the measurement of coverage, reliability and latency according to such definition of this parameters:

- Coverage: It is perceived as the ability to any node to transmit data to any other node in the network using the implemented routing protocol. The results may be compared with the coverage established by the radio communication. It is important to note that although all sensors are covered by radio transmission at some level, this does not guarantee that any each one of them is able transmit data to any other node, given that it might exist a network partition.

- Reliability: It is perceived as the quality of communication given the transmission method, which comprehends the entire software stack on the sensor. The result indicates the degree of quality of the information that a given sensor can send to any other one. It is measured by evaluating the number of sent messages against the number of received ones, typically using a sender node and a receiver one(usually a base-station). Some parameters may alter the result such as the number of retransmissions or the transmission interval, which when badly used may contribute to a degradation in the network's transmission quality.

- Latency: It is a very important metric on the analysis of a network. Typically, it can measured in two ways: a) the time it takes for a message to travel between two nodes; b) the number of hops that a message travels before reaching its destination. The first way is hard to measure in a simulation environment, given that the time factor may not have the necessary resolution in order to give quality results. The second way is a much more generic and precise way to measure and one can always assume the time by assigning an estimated value to the elapsed time based on the available bandwidth.

3.2.2.5 Attacks/Failure injection mechanism

This module is critical in order to assess a secure routing protocol performance under attack. It assumes that any attack on a sensor means the tampering or suppressing of a message content. It allows to trigger the attack mode on any given number of sensors, as well as selecting the type of attack that each should perform, allowing it to coexist with different types of attacks on other nodes. Only one attack can be performed at a time for each sensor. The impact of this module

can be measured by the number of messages that goes through the attacking nodes, as well as the difference in the indicators measured by the previous module.

3.2.3 Simulation and visualization controls

In order to allow the user to interact with the simulation, this module allows the user to control the execution and configuration parameters as well as extracting the results from the simulations either by numeric values or charts.

3.2.4 Routing protocols simulation API

The systematic analysis between different protocols and different implementations can only be made if all of them correctly use the above mentioned mechanisms. As such, the simulator offers an API to each layer(Application, Routing and MAC) so that by using it, the implementation takes full transparent use of available evaluating mechanisms.

3.3 INSENS

This thesis' proposed protocol, MINSENS, is born as an evolution on INSENS[21], a secure routing protocol for wireless sensor networks. INSENS assumes a more powerful adversary model than the one described by Dolev and Yao[22] given that it expects an attacker to be able perform internal attacks on the network nodes, gaining access to all their information and being able to modify, copy or inject code on such nodes. Albeit this, the base stations are assumed to be trustable nodes that communicate between each other through a non secure and non reliable network.

INSENS'[21] approach on intrusions is based on route redundancy, making it able to cope with their existence without even having to detect them. One of its major goals is to prevent a single node from affecting all others, aiming to offer a gracious decrease in the network performance as the number of attacking nodes raises. INSENS'[21] offers protection against three major attack types: DoS, attacks on routing behavior and intrusion attacks. The attacks on routing behavior can be made on three distinct phases of the protocol: on the route discovery phase by sending false routing information or through rushing attacks [67]; on the route selection phase using hello flood attacks[33], sinkhole attacks[33], wormhole attacks [66] or sybil attacks [31]; after the establishment of routes by using blackhole attacks[29] or spam attacks[54].

This protocol takes advantage of the asymmetry on the hardware level between base station and regular nodes, typical of wireless sensor networks, where the base station nodes are more powerful on computation, memory and transmission capabilities as well as less energy constrained. The heavy duty computations are therefore done by the base station freeing regular nodes from tasks such as calculating forwarding tables or dealing with intrusions. Each node shares a cryptographic key with the base station, which raises the problem of key distribution.

INSENS'[21] specification does not address this problem, which lead us to assume that the keys are distributed in a pre-deployment fashion.

The network nodes communicate by wirelessly broadcasting at the MAC level typically using standards such as 802.15.4[25] or ZigBee[6]. Given the transmission range limitations, a routing algorithm that creates a multi-hop network is necessary in order to fully communicate with all of the network nodes. On INSENS[21], the communications are limited to prevent a malicious node to target the entire network. Therefore, only the base station is allowed to do broadcasts at the routing level, authenticating them using one-way-hash-chains similar to those used in μTesla, using the chain to generate one-way-sequence(OWS) numbers. The authentication simultaneously prevents message's tampering because it has an associated hashing mechanism. Unicast communications on the routing level are all made through the base station enabling her to filter possible DoS attacks.

INSENS[21] makes use of multi-path routing in order to cope with intrusions on the network. By using several disjoint routes to deliver a message, it mitigates the risk of an attacker affecting every single copy of the message or preventing it from arriving to its destination. The routes used are as disjoint as possible, only sharing the sender and destination nodes on their path.

This protocol's operates in three phases and its behavior can be described as follow:

- Route request phase: the base station floods the network with a route request message, which is propagated by all the nodes. Each node discovers his neighbours by listening their broadcasts. The behavior can be described as follows:

```
Received Route request message
  if the message is fresh(ie, never seen by this node)
        if one-way-sequence number is valid
          update round's OWS number
          add message's sender as neighbour
          broadcast the message altering its sender id to this node's
          start message feedback response timer
```

- Route feedback phase: Each node sends a feedback message to the base station informing its detected neighbours. This message proceeds in the reverse path by which the route request message reached this node. The behavior can be described as follows:

```
Received Route feedback message
If message MAC == node's MAC
//if the message comes from a child node
  if node is sink
        store feedback message
else
        alter message payload to include this node's' father's MAC
        broadcast the modified message
```

- Routing table propagation phase: The base station calculates all the forwarding tables and sends them to the respective nodes using routing table propagation messages. The behavior can be described as follows:

```
Received route update message
If message is destined to this ndoe
        update forwarding table
else
        if routing is stable
                forward message
```

From the study on INSENS' characteristics, we can state that this protocol has some limitations on the defense against the previously mentioned type of attacks such as sinkhole/blackhole/-wormhole/rushing attacks. The specific faults are:

- lack of a intrusion detection mechanism
- lack of an external attacks detection mechanism
- lack of corrective activity in the presence of attacks
- does not enable any scheduling policies on route usage(it always uses all routes)
- single point of failure(the base station)
- does not take into account network balance on routing and energy consumption
- focus work load on the nodes near the base station

3.4 Objectives

MINSENS is born as an improvement over INSENS[21] using multi-path routing to several base stations. This protocol offers a variety of transmission semantics, as well as a probabilistic consensus mechanism between base stations in order to detect and react possible attacks on the network, by re-organizing the network or deleting compromised routes.

These properties aim to make MINSENS better than INSENS[21] in three major aspects:

- augment resilience towards sinkhole/blackhole/wormhole/rushing attacks
- balance the network routing by allowing flexible trade-offs between the resilience level and the energy consumption
- minimize central points of failure (nodes near the base station)

3.5 MINSENS

As previously mentioned, MINSENS is based on INSENS[21]. The starting point was an extension to its behavior to multiple base stations in order to achieve this thesis' previously mentioned objectives.

The increase in resilience against sinkhole/blackhole/wormhole attacks comes from the increase in the number of routes as well as the probabilistic consensus executed on the multiple bases stations. This way, not only do we statistically increase the number of compromised nodes necessary to a successful attack on all routes, as we enable the protocol to detect and recovery from possible attacks by successfully applying consensus over the received data discarding tampered information. The rushing attacks are also mitigated by the existence of several base stations, making necessary for the attacker to be present near all of the base stations to successfully execute the attack, given that just one correct base station dissemination tree is necessary to render the rushing attack useless.

The scheduling semantics on message transmission allow for an evenly balanced work load on all nodes, diminishing the computation and transmission load on nodes near the base stations and enabling the improvement of energetic efficiency which in turn increases the network's life time.

With only one base station, the compromising of the nodes near it would be enough to render the entire network useless by creating a partition on the topology. The use of several base stations mitigates this problem by increasing the number of necessary compromised nodes as well as varying the locations of such nodes.

3.5.1 Protocol description

MINSENS assumes the existence of N base stations, considering them as secure nodes that work in a independent way between each other, interconnected by a non secure network.

Let B be the number of base stations, and N the number of nodes, MINSENS specification is similar to INSENS'[21] except for the following aspects:

- Each base station works in parallel with others
- It is necessary to configure all nodes with B one-way sequences each instead of one
- There are BxN pairs of symmetric keys necessary instead of N pairs
- Each node has B symmetric keys pre-configured instead of one, being each key associated to a base station
- Each node has B parent nodes instead of one
- The Route Request messages include an integer as the base station identifier, and the MACR includes this number

- The Route feedback messages include an integer as the base station identifier
- The data messages include a list of route ids
- Each node has B forwarding tables, which consults when it wishes to route a message. The number of used forwarding tables depends on the desired level of resilience.

Each MINSENS base station works as a single INSENS[21] base station by creating its own dissemination tree that it sends to the nodes. The regular nodes participate in B parallel routing processes and receive B forwarding tables, merging them into its own single forwarding table. Given MINSENS' multiple base station attributes, every base station is considered as the same final destination after the routing setup process finishes.

One important aspect that is worth noting is the association of an Id to each route. On the original INSENS[21] specification, the nodes did not know what routes they had to the base station, only the ones whose membership they belonged to. To transmit a message, a node would simply broadcast the message to the air, which would be picked up by its neighbours. These would identify the message's route by the triple: source id, destination id and immediate sender. The route membership would be built based on these three attributes, which would cause every node in the membership to respectively forward the message once it was detected as coming from the previous node on the route. This was a problem when it came to making the sender node able to choose which routes he wanted to use. Using this procedure, it was impossible to send a message only using one route, firstly because the sender did not know how many routes it had to the base station, and secondly because all nodes in the route membership would forward the message.

The solution was to create a route Id, and add that information on the data messages. This gives control to the sender node, which also sends the list of route ids that it wishes to use on each message. The route ids are assigned by the base stations while building the forwarding tables. Each route id starts in 0 and ascends for each different path to the same destination. In the end, the number of routes to the base station is added to each nodes' forwarding table, enabling it to know how many routes it possess to each base station. This allows a node to choose to which base stations and by which routes should he forward data, according to its transmission scaling policies.

MINSENS integrates two mechanisms that use consensus to augment this protocol's resilience; a route disjointess mechanism and a pro-active recovering mechanism based on the idea of a byzantine agreement made on the received data. In order to implement and test this mechanisms, it was necessary to feature a reliable broadcast layer over which these techniques would run. Both the mechanisms and the broadcast layer will be explained in detail in the following sections.

3.5.2 Transmission scaling

The existence of several base-stations increases the ability to balance the network routing load. This can be achieved by distributing the routing along different paths that use different nodes,

therefore balancing energy consumption. The same idea can be applied to security, by adjusting a trade-off between energy consumption and the level of routing redundancy achieved through multi-path routing.

There are many existent scaling semantics, we choose four as most suited to MINSENS:

- one random route
- one route by round-robin
- K of all N routes
- simultaneous use of all routes

From the theoretical point of view, the random and round-robin strategies should yield similar results, offering an increase in security compared to the regular INSENS[21] by mitigating the use of compromised routes, as well as an overall lower and more balanced energy consumption due to the distribution of work load by different routes to different base stations. The use of all possible routes adds strain to the energy consumption but offers the best level of resilience to attackers, by using the full capabilities of the routing mechanism to escape compromised routes.

As previously stated, each node treats all base stations as a single one. This enables the routing layer to override the destination id provided by the application layer. A message that an application wishes to transmit to a given base station can, in the case of a multiple base station routing protocol, be forwarded to a different or even more than one base stations in a transparent way. Upon receiving a message to transmit from the application layer, the MINSENS protocol chooses which routes to use according to the scheduling policy, and creates as many copies as necessary to forward the information using all the chosen routes.

This transparency, working without any information from the above layer, has a drawback. The application layer assumes that one message is sent, and in the worst case scenario, if the routing layer is using an aggressive setting of multi-path usage, it can in reality multiply the number of real messages by the number of base stations times the number of routes per base station. This might be a problem to applications that have specific needs, although this behavior is configurable on the routing protocol.

3.5.3 Reliable and Echo Broadcast primitives

A reliable broadcast primitive must ensure two properties:

- all correct processes deliver the same messages
- if the sender is correct, then the message is delivered

One example of such protocol was proposed by Bracha[**?**] which works like this: the sender broadcasts a message (INIT,m) to all processes. Upon receiving this message a process replies

a (ECHO,m) message to all others. It then waits for at least $\lfloor$(n+f)/2$\rfloor + 1$ (ECHO,m) messages or $f + 1$(READY,m) messages, after which it transmits a (READY,m) message to all others. After receiving $2f + 1$(READY,m) messages, it delivers m.

The echo broadcast primitive is a relaxation of the reliable broadcast primitive. It is more efficient than reliable broadcast while maintaining most of its properties. However, it does guarantee that all correct processes deliver a broadcast message if the sender is corrupt[59]. This means that the protocol only guarantees that the subset of correct processes that deliver will do it for the same message. This protocol is similar to the *reliable broadcast* algorithm excluding the last step.

These primitives are necessary to enable the use of consensus which in turn is necessary to the route disjointness and data recovering mechanisms explained bellow.

3.5.4 Consensus on disjoint routes

The results of multi-path routing to multiple base stations depend on the level of disjointness between different routes. If different routes share internal nodes, the odds of an attacker successfully targeting more than one route with only one compromised node raise and render the multi-path routing useless. As such, although the alternative paths calculated by each base station are 100% disjoint considering that base station's paths, the routes of different base stations could be only partially disjoint between each others.

To improve the results achieved by MINSENS, we designed and implemented a consensus protocol that aims to achieve 100% disjointness between all existent routes for a given destination(independently of the source base station). The protocol involves all the base stations and works under the assumption that all members know the membership, which is stable throughout the entire routing setup process. It also uses a broadcast primitive through which it reliably and atomically communicates with the other base stations. The protocol's behavior can be described as follows:

```
Run the first three phases of MINSENS protocol normally:
        Gather network information
        Generate the network's graph and paths
Broadcast its generated paths to the other base stations
When routing information is received from all other base stations
       && this base station has finished the first three steps
        run deterministic algorithm to discart shared routes
Dissiminate the remaining routes that were generated by this base station
```

After having all routing information from his routing process and other base stations processes, the mentioned deterministic algorithm that chooses which routes to discard works as follows:

```
Merge all paths into one Vector
Sort the Vector ascendently by path size
for each path
        if path size is bigger than 2
                get route destination
                usedNodesList = list of nodes used to reach this destination
                for each node in route except for source and destination
```

```
                        if node exists in usedNodesList
                                exit this cicle
                        else
                                add this node to this usedNodesList
                                if this is the penultimate node in route
                                        add route to disjoint routes set
        else if path size is 2
                add route to disjoint routes set
allpaths = disjoint routes set
```

This algorithm sorts all the possible routes and it then iterates them storing the traveled nodes in each route per destination. When a route is analyzed, every traveled node(except for source and destination nodes) is checked to have been used on a previous route to this same destination. If any node is caught, then the route is not entirely disjoint with the previous ones to the same destination, and is not therefore added to the final set of routes.

When the algorithm finishes, only the totally disjoint routes are in the final set, which becomes the used sets in all base-stations.

The algorithm is fully deterministic, which allows it to run in parallel on every base station, achieving the exact same results on all base stations, given that it starts with the same original set of paths on every node that it operates.

3.5.5 Byzantine agreement: probabilistic consensus

In order to establish a Byzantine agreement, the typical protocols work in a deterministic way. This property makes them unsuited for failure-prone scenarios, such as the wireless sensor networks which operate in an asynchronous fashion over wireless networks, in a typically non-structured network. In scenarios such as this, the deterministic protocols are bound by the FLP impossibility result[24].This impossibility states that consensus is impossible to resolve by deterministic protocols in asynchronous systems if even one node can fail. In order to overcome this, one must use randomization[17] as a solution to this problem as demonstrated by Moniz[47].

As described[47], the consensus services run over of the *reliable* and *echo broadcast* primitives. The first level of consensus is the *binary* one, which lets the processes agree on a single bit. Using the *binary consensus* is possible to build the *multi-value consensus*, which enables the nodes to decide on an arbitrary set of values. Over this runs the *atomic broadcast* which ensures total order and the *vector consensus* which lets the processes agree on a vector composed of values proposed by a subset of the processes.

The protocols of RITAS[47] feature a set of properties that are relevant for this thesis' context of wireless sensor networks; They work in an asynchronous fashion without assumptions on the processes' relative execution or communication delays; They attain optimum resilience tolerating up to $f = \lfloor (n-1)/3 \rfloor$ malicious processes out of n processes; They are signature-free discarding the use of computationally expensive public-key cryptography which is important in wireless sensor networks.

The Byzantine agreement proposed on this thesis is a version of vector consensus, using the received data between all base stations to propose a set of correct values, enabling a corrective approach over tampered data by discarding it after the correct values have been agreed.

4. Implementation

This thesis comprehended an implementation of the original INSENS[21] protocol ,the proposed MINSENS protocol along with the extensions to the simulation platform made on the context of this thesis.

Although the set of features provided by the original implementation of the Wisenet[1] simulator covers most of the necessary indicators on the assessment of a routing protocol, information on the generated topology such as the number of generated routes or its distribution on the nodes was lacking and was therefore implemented.

4.1 Simulation Platform Extension

Given the above mentioned features and mechanisms, the assessment of this thesis' work was greatly facilitated. Even though this, there were some necessary parameter measurements and interfaces that were absent on the original implementation. In addition to that, as the experimental evaluation begun, some features arose as nice to have ones that would allow a more productive assessment.

Given this, the extensions made to the original Wisenet simulator were the follow:

4.1.1 Route Disjointness module

The original objective of this module was to measure the disjointness of the routes used in the multi-path routing of INSENS[21]. Along with this, some other indicators were interesting to retrieve, such as the average number of routes per node, the distribution of the number of routes from primary to alternative ones. Last, it would be important to have a chart demonstrating the distribution of routes through all the nodes.

Given the above mentioned features, this mechanism was thought as to be work with any routing protocol implementation, be it single or multiple base-station, using or not consensus. Being so, the implementation was made in two different directions. The first, aimed at consensus-free routing protocols, is as follows:

```
for all basestations
raise number of sink nodes
retriview the primary set of paths
for each path
        raise total number of routes
        raise total number of primary routes
        check route count for this destination
        raise route count for this destination
        for each alternative set of paths
                find alternative path on this set
                if path is found
```

[1] WiseNet homepage: http://code.google.com/p/secwsnsim/, accessed in Aug/2011

```
raise number of routes
raise number of alternative routes
raise route count for this destination
if path and alternative path are disjoint
        raise number of disjoint routes
else
        raise number of shared routes
```

Besides this, the mechanism obtains the total number of nodes and total number of stable(nodes participating in the routing process) nodes from the simulator.

As one can see from the above code, this algorithm searches in each base station every alternative set of paths for alternative routes and adds every parameter according to its finding. In the end, after running on all base-stations, the module has all the necessary information in order to calculate:

- The number of sink nodes
- The total number of routes
- The number of primary routes
- The number of alternative routes
- The percentage of disjointness of routes, by comparing the number of disjoint routes with the total number of routes
- The total number of nodes
- The total number of stable nodes
- The percentage of nodes with alternative paths(more than one path)
- The average number of routes per nodes, by doing an average of the route count for every node

The second algorithm, aimed at protocols that use consensus on base-stations and therefore store all routing information in each base-station, only runs on one(any) base-station and operates in the following manner:

```
update total number of sinks from simulator
get all paths from a given sink node
create a allpaths_clone
for each path in allpaths
get destination id
if allpaths_clone contains path
        raise total number of routes
        raise number of primary routes
```

```
        remove path from allpaths_clone
        check route count for this destination
        raise route count for this destination
        find and remove alternative path from allpaths_clone
        while alternative path exists
                raise total number of routes
                raise number of alternative routes
                raise route count for this destination
                if path and alternative path are disjoint
                        raise number of disjoint routes
                else
                        raise number of shared routes
                find and remove alternative path from allpaths_clone
for each node in simulation
if node is not stable
        put route count as 0 for this node
```

In the end, this algorithm retrieves the same information as the previous one, but only accessing one base-station to do so. The clone of the all paths structure is done so that we can remove paths from analysis while iterating, because once a path is considered as an alternative route to other, there is no need to consider it when the iteration gets to him.

4.1.2 Multiple-Test execution

To achieve reliable results, the analysis on the referenced network parameters(coverage, reliability, latency, etc..) must be done on more than one observation. On the experimental evaluation setup, we defined that any considered value should be the average from a twenty observation experiment. As such, the process of manually executing twenty separate tests, collecting its results and doing the average appeared as an ineffective and error-prone one. The solution was obvious, to change the simulator so that it could run a set of N experiments and show the average results.

An AverageAdhocTest class was created, which represents the new notion of an averaged test. Given a number of times to run, this test launches an equal number of childs which are modified versions of an AdhocTest. In order to obtain the same behavior of N sequential manual different experiments, each test must change the selected nodes on which it will work. That being said, every child test runs on the same network topology and routing protocol instance, but with different sender and attacker nodes. The receiver nodes stay the same because all tests were made using as receivers all available base-stations. Each child AdhocTest starts by clearing all nodes from sender and under attack selections, and then randomly select nodes in order to meet test criteria as to the number of sender and attacker nodes. It then runs normally registering its data on the father's EvaluationManager and when done, it notifies the father AverageAdhocTest.

Once all child tests end, the AverageAdHocTest ends. The results presented to user are modified so that they reflect not all the tests but the average results of them. Let N be the

number of runs the average test did, the results are changed in the following way:

- number of attacked messages : divided by N
- number of sender nodes: divided by N
- number of attacked nodes: divided by N
- number of unique messages sent: divided by N
- number of unique messages received: divided by N
- total of spent energy : divided by N
- average energy cost per node : each node's spent energy is divided by N

By altering the results in the above described way, the final results will represent a simple average based on all the results of the individual tests, providing the same result as an average on the results of N individual runs.

4.1.3 Changes on simulation configuration

The evaluation of routing protocols is typically based on assigning percentages to a given number of parameters, such as the amount of sender or attacker nodes for example. The original implementation of the simulator only allowed for manual selection of nodes by selecting them on the network map. This posed as a problem to an effective evaluation on the protocols.

The solution was to change the simulation configuration so that in addition to the manually selected nodes, the simulator would add random nodes (that were not already selected) to match the simulation node amount criteria. This can now be done by simply entering a number to the sender or attacker nodes and selecting the appropriate number interpretation (a button exists to use the inputed value as percentage, otherwise is treated as the number of nodes).

4.1.4 Atomic broadcast to base stations interface

The characteristics of this thesis' implemented protocol required a out of range communication method between the base stations. The simulator's original implementation offered an interface through which every message would be sent according to the used MAC and radio models. As such, an interface was built on the simulator, through which any node can broadcast a message to all base stations. The method was made so that if a base station sends a message, it will not be a receptor of its own broadcast. This method offers an atomic broadcast semantic on every message sent.

4.1.5 Changes on energy charts

One of the simulator's visualization controls is the energy chart module. This module, on its original implementation, would only show 15 nodes on the graph, and the axis would be a zero to one hundred scale. Because this thesis' intended evaluation was on the entire network, this module was changes in three ways: first, the chart now shows all the nodes; second, because all the major simulations would only consume about ten to twenty percent of a node's energy, the axis is arranged as to its lower end is the least value on the graph minus 0.25; third, in order to get an easier understanding of all the energy consumption, the values are shown in an ordered form, so that they create a curve and therefore make it easier to understand the energy consumption pattern.

4.2 INSENS

INSENS'[21] implementation was necessary as a way to set the benchmark to the evaluation of MINSENS' performance. The code was developed using the simulator's API to the routing layer, making transparent the use of its core evaluation components.

One of the simulator's abstractions used on this implementation was the Timer. As the name indicates, it acts as a timer to trigger or schedule events according to an internal clock, much like the Timer implementation on TinyOS[2]. The Timers were used to coordinate the transition between the three protocol phases, scheduling the start of route feedback phase on the regular nodes as well as the forwarding tables calculation and propagation on the sink nodes.

The executed code on the sensors is very similar and only differs according to the role of the sensor in the network, as a base station(sink node) or as a regular node. On the first role, the node controls the information flow and is part of the trusted computer base, given that the protocol assumes the base stations as safe and reliable nodes. On the second role, the node generates events and forwards his and other nodes' data. Given the context of a simulation platform, the setting of the nodes' roles is made according to the simulation configuration.

On regards to the base station nodes, the most important implemented methods are:

- newRouteDiscovery(): sends the first route request message to the network carrying its id and starts the feedback messages receiver timer. This message is authenticated by the base station.
- processFDBKMessage(): verifies the received feedback message by analyzing its origin, its integrity based on the keyed MACs and on the information of the traveled path. If everything is ok the message is stored for later processing.
- startComputeRoutingInfo(): this method is called after the base station receiving information during the interval specified on the Timer. It calculates the forwarding tables based on the gathered information by running the Dijkstra algorithm.

[2]TinyOS homepage: www.tinyos.net, accessed Set/2011

- sendRouteUpdateMessages(): after calculating all forwarding tables, the base station propagates the information to the regular nodes, in an descending fashion according to hop distance. This enables the use of the newly calculated paths to distribute the forwarding tables.

On the regular nodes' side, the most important methods are:

- rebroadcastRREQMessage(): if it is the first time that this route request message is seen, the node broadcasts the message to its neighbours adding its id to the message's information. It triggers the route feedback message timer.

- sendFeedbackMessageInfo(): after the route feedback message timer expires, this method is called and all the information about the discovered neighbours and their signatures is sent to the base station.

- processFDBKMessage(): this method forwards route feedback messages back to the base station. Given that the route feedback messages travel the same path(on the opposite direction) that the original route request message travels, every node is responsible for forwarding to the base station the route feedback messages of the nodes who heard his broadcast of the original route request message.

- updateRoutingStatus(): after receiving the routing information from the base station, the node updates its forwarding table and turns its state into stable, meaning that it can now participate in the network's routing.

Given the experiment on a simulation platform, some minor details on the context of this thesis' work were relaxed according to the specification. These do not alter the specified protocol behavior, therefore not influencing the results.

4.3 MINSENS

During the implementation phase of this thesis' work, some choices were made in regards to aspects such as the way the multi-base-station routing semantics, the transmission scheduling policies or the consensus applied on the selection of disjoint routes.

The changes made on the routing mechanism in order to get it to work for multiple base stations were one of the most challenging ones. In the original protocol, each node knows one base station with who runs the specified protocol phases in order to participate in the routing process. With multiple base stations, the definition of the routing protocol setup phases were not trivial, also baring in mind the need for the route consensus algorithm implementation to be flexible enough to be able to be used or not through parameterization.

Although the base stations membership is assumed to be stable throughout all the routing process, we decided to parallelize the entire process. Each base station runs independently its

routing setup phases, building her dissemination tree based on the collected information. As previously mentioned, MINSENS was born as an evolution of INSENS[21], which builds the network routing in three different setup phases(Route request, route feedback and forwarding table propagation phases). Each MINSENS base station runs the same initial three phases with similar implementation, collecting data and building the forwarding tables which are then sent to the nodes in an INSENS'[21] fashion. If the consensus on routes is to be made, the disjointness consensus algorithm is executed prior to building the forwarding tables, resuming the normal process after his execution.

4.3.1 Routing

The extension to multiple base stations implies an extension to the information that each node must possess, as stated in the protocol description in chapter 5. This information is kept in the following structures:

- LinkedList<Short> baseStations: a list of the node ids of all base stations, pre-configured through the simulator
- Hashtable<Short,byte[]> privateKey: an hashtable containing the symmetric private keys shared with each of the existing base stations
- Hashtable<Short,Long> OWS: an hashtable containing the last used number of each base station's one way sequence number
- Hashtable<Short,Integer> roundNumber: an hashtable containing the round number of each base station
- Hashtable<Short,NeighborInfo> neighborInfo: an hashtable containing the neighbours detected in each base station's route request phase
- Hashtable<Short,byte[]> myRoundMAC : an hashtable containing the node's MAC for each base station

On the original INSENS [21] protocol, the regular nodes are spared from any heavy calculations and it is the base station that calculates and builds all the forwarding tables. MINSENS follows the same pattern, but allowing the nodes for bigger freedom in regards to the routing policy, which in turn raises the computational complexity on the regular nodes. Each node maintains its forwarding table, that is built by merging all the received forwarding tables. It can then use that information to decide which routes to use according to its scaling policy.

Routing wise, the implementation is very similar to the INSENS' one, adjusted baring in mind the differences in the data structures to maintain the state for all parallel routing processes. The less obvious methods are :

- onStartUp():
  - base station: setup of round number, roundOWS, private key, initial sequence number from the one way hash chain and timers
  - regular node: for each existent base station, setup of setup of round number, roundOWS, private key, initial sequence number from the one way hash chain and timers
- haveRoute(short destination, short source, short immediate, LinkedList<Integer> routeIds): it iterates through all the routing membership entries, in search for one that has the same source, destination, immediate sender and route id as the parameters which are from a received message. If it finds one, the node belongs to a route membership and should forward the message, otherwise it does not.
- updateRoutingStatus(RUPDPayload payload): It adds the received forwarding table to its own, marks the node as stable and replies to the base station acknowledging the received message.

4.3.2 Transmission Scheduling

As previously mentioned, the routing protocol intercepts the application's transmission requests and changes the destination base station according to its scheduling policies. This operation is made on the sendDATAMessage method, that is explained in detail bellow.

Some minor adjustments to the payload of the RouteUpdate/RouteUpdateAck/Data messages were made in order to include information such as the base station id that identified the message to a base station's routing process, as well as the route ids that are specified in the message header.

The transmission scaling policies were defined in the implementation as constants in the following way:

- public static final int $K = 3$;
- public static final int $K\_RANDOM\_ROUTES = 0$;
- public static final int $K\_BALANCED\_ROUTES = 1$;
- public static final int $ONE\_RANDOM\_ROUTE = 2$;
- public static final int $ONE\_ROUND\_ROBIN = 3$;
- public static final int $ALL = 4$;

The most important methods are the follow:

- sendDATAMessage(Message message): it chooses which routes to retrieve and how many messages to send according to the chosen policy

- getKRoutesBalancedByBS(int k): it divides the K parameter by the number of base station, retrieving K routes per base station. If any of the base stations does not have enough routes, it gets all the available ones from that base station.
- getRandomRoute(): it generates a pseudo-random number between 0 and the total amount of routes to the base stations. It then retrieves the route that corresponds to that number, iterating every base station's forwarding table and counting the routes until it reaches the generated number.
- getKRandomRoutes(int k): it runs the getRandomRoute() k times in order to obtain K random routes
- getNextRouteRoundRobin(): it keeps track of the last used route, and increments it each time it is called. This method uses that information to retrieve the next route by assigning a number to all routes and using a circular buffer style counter to get the next route.
- getAllRoutes(): it iterates all the forwarding tables and returns all the existing routes.
- createRoutingMessage(Message message, short destBS, LinkedList<Integer> routes): It clones the received message and encapsulates it on a routing layer message, setting the source, destination, immediate sender and route ids.
- processRoutes(Message message, Hashtable<Short,LinkedList<Integer» routes): iterates the collection of given routes creating(through the createRoutingMessage() method) and sending a message per base station which has the list of route ids for that same base station.
- processOneRoute(Message message, SimpleEntry<Short,Integer> route) : creates(through the createRoutingMessage() method) and sends a message to the specified base station using the given route id. This method is used to send a single message by one route.

The option to include a list of route ids on a given message was made to enable only one transmission for all the copies per base station. If each message only possessed one route id, then it would be necessary to transmit every message individually. By using a list of route ids, the message payload is bigger but the entire message is only broadcasted once per destination(base station) and picked up by all neighbours, which can identify their routing membership in the list of ids. This decision reduces the energy consumption of the protocol, which is beneficial for the network's lifetime.

4.3.3 Consensus on disjoint paths

The implementation of a consensus based on an atomic broadcast method required first of all an implementation of such communication method. To achieve so, we implemented this feature

on the simulator, bypassing all simulation models such as radio or MAC ones. This allows the transmission to be made according to the atomic broadcast semantic with the respective properties. This implementation works by making the simulator iterate through the list of sink nodes and delivering information on every base station except for the one requesting the operation.

Once this method was available, the implementation of the consensus algorithm to ensure the total disjointness between all routes to a given destination was made using the following methods:

- startComputeRoutingInfo(): verifies the state of the protocol in order to start the calculation of the forwarding tables. In case of use of the consensus protocol, it obtains the membership of the base stations from the simulator.

- calculateForwardingTables(): triggered by startComputeRoutingInfo, it does all the calculations running the Dijkstra algorithm and calculation all paths. If using the consensus on routes, it sends the calculated routes to the other base stations using the simulator's atomic broadcast method

- registerSinkFeedback(LinkedList<Object> paths): is the method called by the simulator to deliver the atomic broadcast data. It merges the received routes and when the last set of routes is received, removes the non-disjoint routes and triggers the building and forwarding of the final forwarding tables.

- receiveRoutesFromOtherBaseStations(LinkedList<Object> receivedPaths): triggered by registerSinkFeedback(), it merges the received paths with the existing ones and when the last set is received, it runs the algorithm to discard the non-disjoint routes.

- discardNonDisjointRoutes(): called by receiveRoutesFromOtherBaseStations(), it runs the algorithm described in section 5 to exclude the non-disjoint routes, leaving the result as the set of all routes.

4.3.4 Byzantine Agreement on data

The Byzantine agreement implemented comprehends two directions, an agreement made on a local level by a single base station and the agreement made on a global level by all base stations.

The local level agreement operates by counting the number of correct replicas received from the different routes. The threshold by which the base station agrees on the received value is configurable on the protocol, and is typically a number between 1 and the number of disjoint routes available. It can however be set to a value bigger than the number of disjoint routes if we consider the use of retransmissions by the sender node. By configuring the MINSENS routing protocol to execute the local consensus, the protocol calculates the number of messages whose number of received replicas with the same exact payload are equal or bigger than the configured threshold value. After having this information, it calculates the respective percentage of that number in relation to all different messages received, giving and indicator of how many messages passed the byzantine agreement.

To implement the Byzantine agreement based on the global consensus, we made relaxation

of the specification described in 3.5.5. We adopted a simplified protocol that has a global parameter which defines the number of global replicas that a value must have in order to make all base stations agree on it. As explained in 5, this agreement yields better results than the local variant because it makes use of all base stations' information. The implemented protocol works as follows:

```
double getGlobalDataConsensusResult(){
  myVotes = calculateLocalConsensusVotes()
  For (BaseStation b : otherBaseStations){
          Vector v = b.receiveLocalConsensusVotesVector()
          totalVotes = mergeVotes(myVotes,v)
   }
   return getPercentageOfConsensualizedValues(totalVotes,voteThreshold)
}
```

Each base station calculates how many correct replicas of each value it has, and sends a vector with all those numbers for all received messages. Each base station then merges the received vectors with its own, achieving a final consensual vector, which enables her to count how many of the total received messages have a total of votes bigger than the global vote threshold.

The implementation of the original model described in 3.5.5 requires a port of the RITAS stack[47] over the base stations which may be of special interest to an experimental evaluation based on a implementation on real nodes as an evolution of this thesis' work.

5 . Experimental Evaluation

In order to assess the MINSENS performance and characteristics agains the INSENS[21], an exhaustive set of tests were conducted. These experiments fully evaluate the protocol according to its previously described implementation on the simulation platform.

This section demonstrates the development of an in-depth experimental assessment study of the INSENS[21] and MINSENS protocols by simulating their behavior in large scale WSN settings evaluating the runtime operation and performance indicators, according with the following criteria:

- impacts in the provided routing support for tree-based data dissemination models in randomized non-supervised topologies: energy cost, connectivity, effective reliability and latency conditions.
- impact in the topology and ad-hoc organization, given the flexibility of the simulation platform to customize the test settings with regards to the following indicators:

indicator	**observation**
Fanout metrics : number of nodes selected to disseminate events at each routing step in order to retransmit information (correlating the observation in terms of trade-offs between desired reliability level and multi-path redundancy level)	The use of scheduling transmission policies along with the variable setting of the number of sender and attacker nodes
Number of maximum rounds: rounds in event retransmission to achieve a certain reliability degree	The setting of different levels of retransmission level by the sender nodes
Degree-distribution metrics: the number of node neighbors in the physical range of each node	The possibility to graphically create, edit and store different network topologies varying the average neighbor count
Average shorted path metrics: correlating the observation with the latency conditions	The ability to use different scheduling policies and measure the respective latency conditions by message hop counting

Clustering coefficient metrics: for the average of the number of links (physical range) connecting nodes to neighbors divided by the number of links between those neighbors	The ability to use different network topologies to measure routing indicators such as the number of routes to the base stations, the number of neighbours, the average route count and the total number of routes
Number of hops in the established multi-hop routes	The possibility to use different scheduling policies and measure message hop indicators
Number of resilient multi-path and multi-hop routes	The possibility to use different scheduling policies measuring multi-path indicators such as the average route count per node, total number of routes or the disjointness level of between routes

5.1 Setup conditions

5.1.1 Parametrization

The experimental evaluation aimed to assess the performance of the proposed protocol in a variety of different conditions given the flexibility of the simulator platform. These evaluations were made through a series of tests. A test is defined by the following parameters which vary in the described way :

parameter	**setting**
number of stable nodes[1]	varies according to the network topology
number of observations	fixed to 10
number/percentage of sender nodes	variable from 10 to 30% of all the network nodes
number of receiver nodes(base stations)	varies from 1 to 4 base stations(according to the network topology)
number/percentage of attacker nodes	variable from 0 to 30%
number of different messages sent per node	fixed to 1
interval between each message transmission	fixed to 10 seconds
the number of retransmissions	fixed to 0

[1] A node is considered stable when it is able to participate in the routing process

type of attack performed by the attacker nodes	no attack or black-hole attack

The parameters that are not fixed vary according to each test and are referenced in the respective evaluation section beginning.

5.1.2 Evaluation indicators

To assess the network performance using the proposed protocol, we chose to evaluate the following indicators:

- **Network connectivity percentage:** It is perceived as the ability to any node to transmit data to any other node in the network using the implemented routing protocol. It is therefore calculated by calculating the percentage of covered nodes(the ones who sent a received message) in relation to the total number of designated sender nodes.

- **Network reliability percentage:** It is perceived as the quality of communication given the transmission method. It is calculated by calculating the percentage of received messages in relation to the number of sent messages.

- **Latency:** It measures the number of hops that a message travels before reaching its destination.

- **Energy:** Evaluation of energy cost metrics associated to protocols and executions.

- **Route count:** the total number of different routes for the entire network(also evaluating and correlation load distributed aspects)

- **Route average per node:** the average number of routes per node to the base stations

- **Local and global data messages consensualized percentage:** the percentage of the messages whose number of received untempered replicas equals of surpasses the consensus thresholds, both local a global.

5.1.3 Network topology

The simulation environment allows the study of large scale WSNs from hundreds to tenths of thousands nodes. For the following experiments, the tested network topologies varied in the number of nodes; 300, 500 and 1000 nodes. These topologies represent respectively $900 * 500$, $900 * 800$ and $900 * 1600$ meters of terrain.The network topologies were the follow:

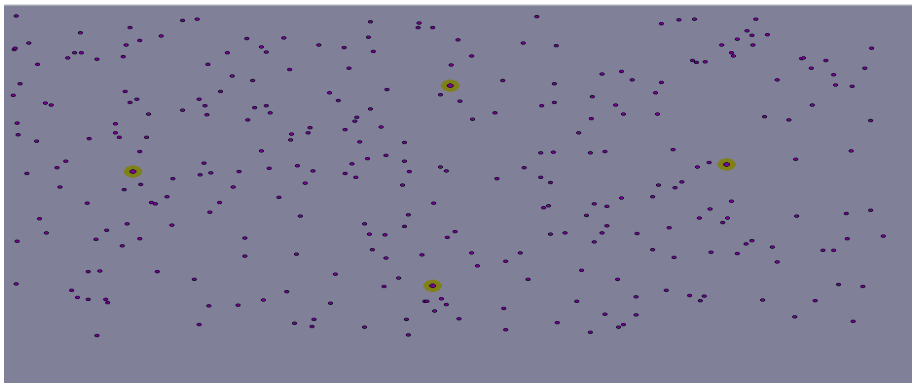

Figure 5.1 Network topology with 300 nodes and 4 base stations

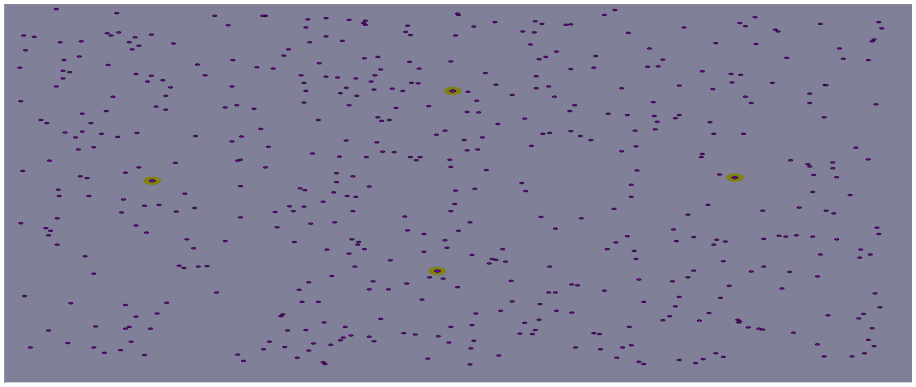

Figure 5.2 Network topology with 500 nodes and 4 base stations

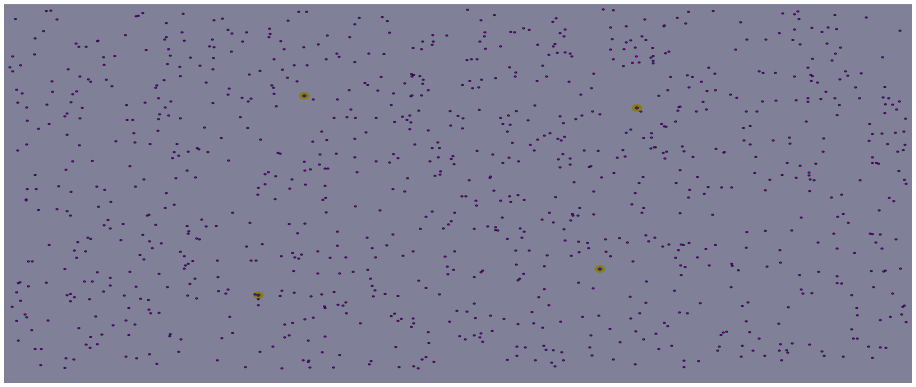

Figure 5.3 Network topology with 1000 nodes and 4 base stations

5.1.4 Test results

The test results shown on this section are calculated as the average values for all the indicators mentioned in 5.1.2, measured in 10 observations per test. Each of these observations use a

different set of sender and attacker nodes, chosen randomly by the simulator. Every test and respective observations were made under the same topology and protocol instanciation. The routing protocol was started before the tests, in order to build the ad-hoc network organization. The tests are performed after the setup phase finishes in order to have the maximum amount of stable nodes. A node is considered stable when it is part of routing process and is able to send and forward messages.

5.2 Comparison: INSENS vs MINSENS

This section presents a general comparison between the original INSENS [21] implementation and the proposed protocol, MINSENS. Unlike INSENS[21], the MINSENS protocol is highly customizable with regards to operation parameters such as the number of base stations, the transmission scheduling policy, the use of route disjointness mechanism or the Byzantine agreement mechanism applied to the received data. For simplicity's sake, this section compares the simpler "weaker" setup of MINSENS, using two base stations with round-robin scheduling policy, without the use of route disjointness or Byzantine Data agreement protocols. The possible setup variations will be evaluated in detail in the next sections.

5.2.1 Connectivity

In this test we used the following test parameters:

parameter	setting
number of stable nodes	250/426/680 - INSENS | 297/490/996 MINSENS
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	1 -INSENS | 2 - MINSENS
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

Results The observed results were the follow:

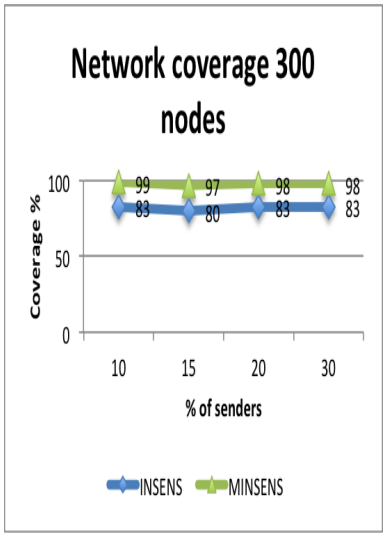

Figure 5.4 Network coverage with 300 nodes

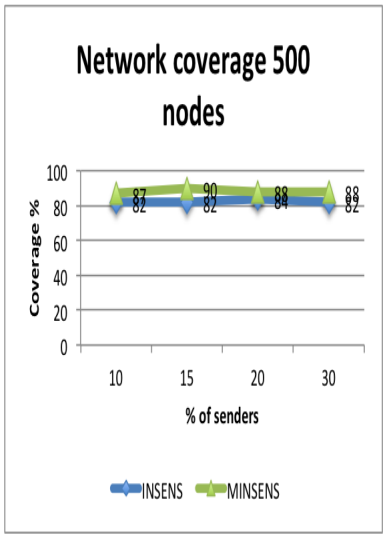

Figure 5.5 Network coverage with 500 nodes

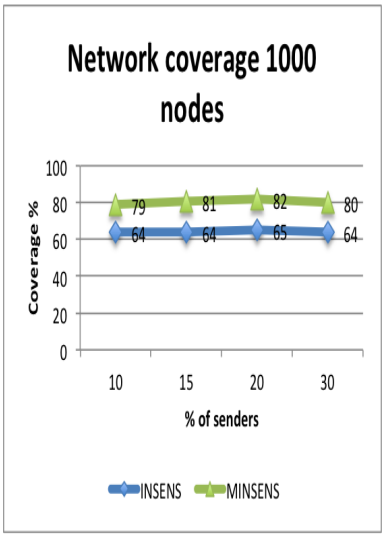

Figure 5.6 Network coverage with 1000 nodes

The network connectivity is limited by the number of stable nodes which is significantly bigger on MINSENS in every condition. As such, MINSENS offers a superior network coverage which scales better as we can see by the connectivity results on the 1000 nodes setting which are approximately 33% superior to the INSENS protocol.

5.2.2 Reliability

In this test we used the following test parameters:

parameter	setting
number of stable nodes	250/426/680 - INSENS | 297/490/996 MINSENS
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	1 -INSENS | 2 - MINSENS
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

Results The observed results were the follow:

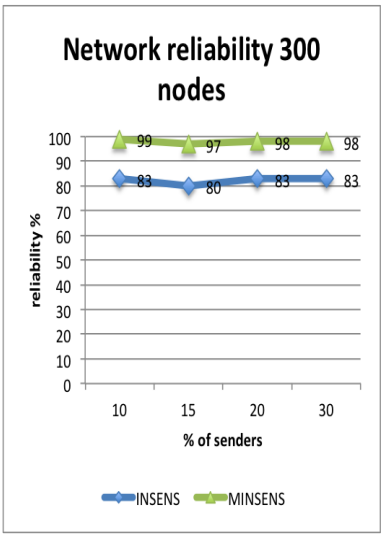

Figure 5.7 Network reliability with 300 nodes

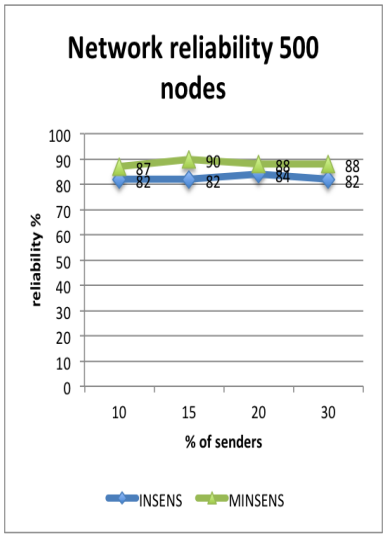

Figure 5.8 Network reliability with 500 nodes

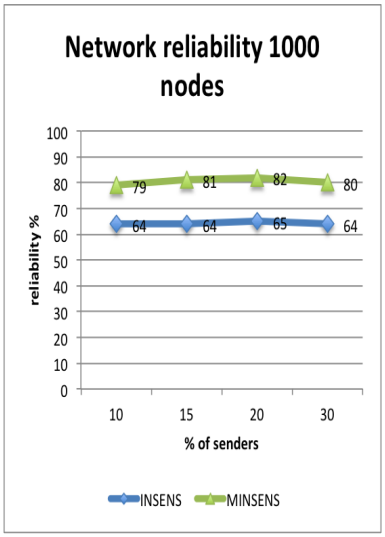

Figure 5.9 Network reliability with 1000 nodes

The communications' reliability is also limited by the amount of stable nodes given that the source nodes are chosen from all the existent nodes. The reliability is better in the MINSENS protocol on all network topologies, again showing specially on the 1000 nodes setting where it is superior by almost a third to INSENS[21] result.

5.2.3 Latency

In this test we used the following test parameters:

parameter	setting
number of stable nodes	250/426/680 - INSENS | 297/490/996 MINSENS
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	1 -INSENS | 2 - MINSENS
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

Results The observed results were the follow:

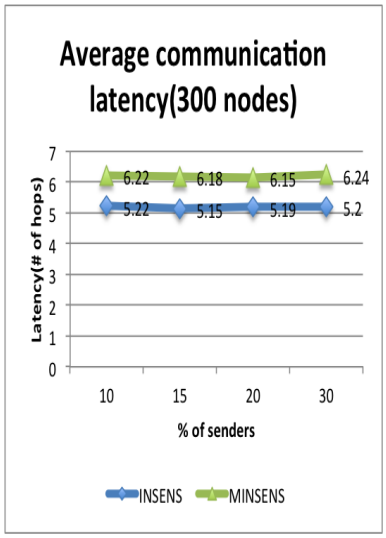

Figure 5.10 Network latency with 300 nodes

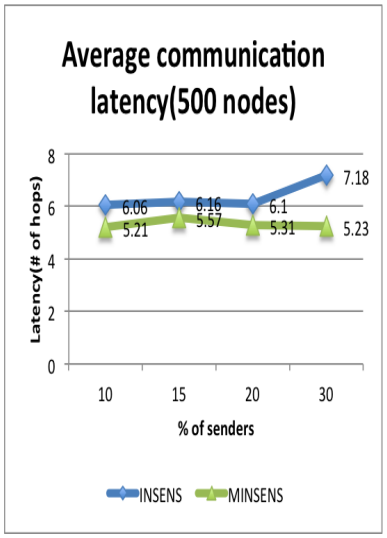

Figure 5.11 Network latency with 500 nodes

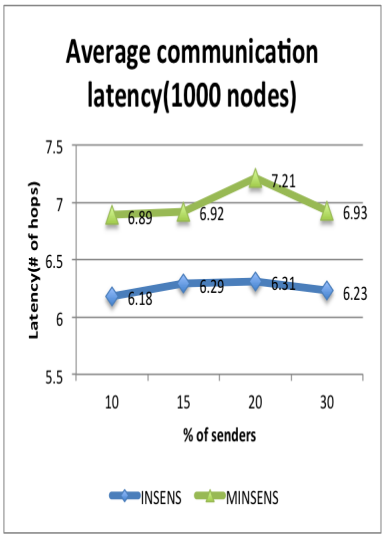

Figure 5.12 Network latency with 1000 nodes

The obtained results are very close and not consistent across topologies. This can be partially explained by the testing settings, where although the MINSENS protocol is set to use a round-robin policy, the testing settings assigns only one message transmission per node, reducing the effectiveness of the round-robin policy.

5.2.4 Energy

In this test we used the following test parameters:

parameter	setting
number of stable nodes	250/426/680 - INSENS | 297/490/996 MINSENS
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	1 -INSENS | 2 - MINSENS
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

Results The observed results were the follow:

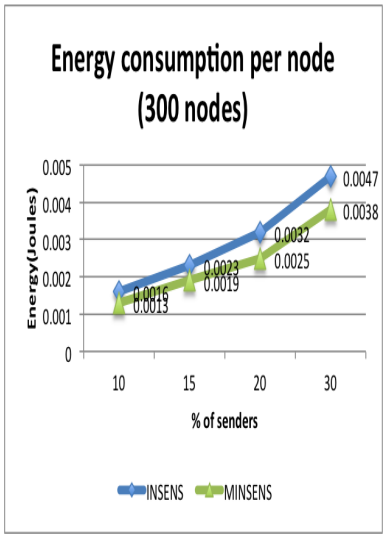

Figure 5.13 Energy per node with 300 nodes

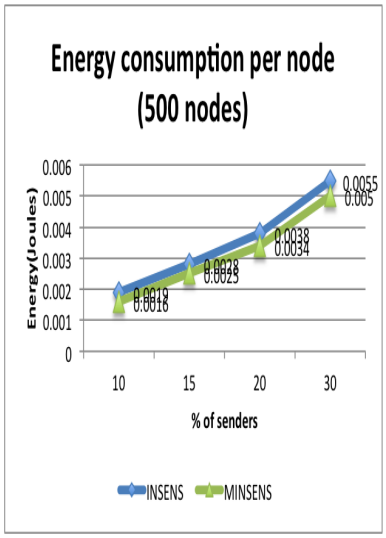

Figure 5.14 Energy per node with 500 nodes

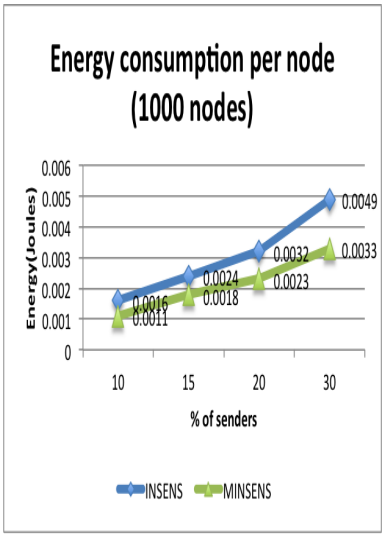

Figure 5.15 Energy per node with 1000 nodes

The energy consumption per node is lower on MINSENS which is a direct consequence of a better work load distribution together with only using one route on the contrary of INSENS[21] that always uses both available.

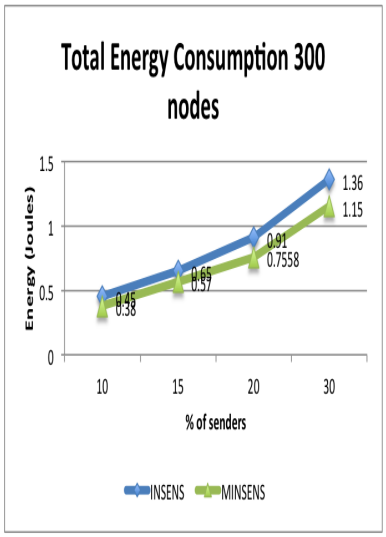

Figure 5.16 Network energy consumption with 300 nodes

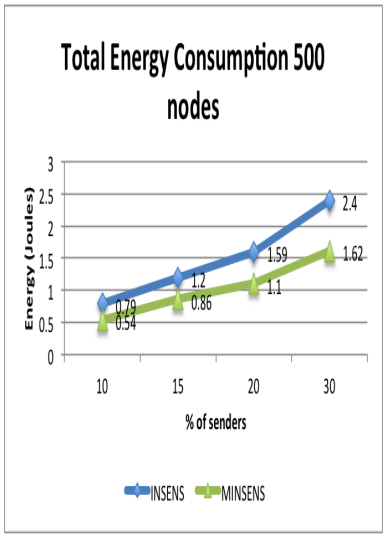

Figure 5.17 Network energy consumption with 500 nodes

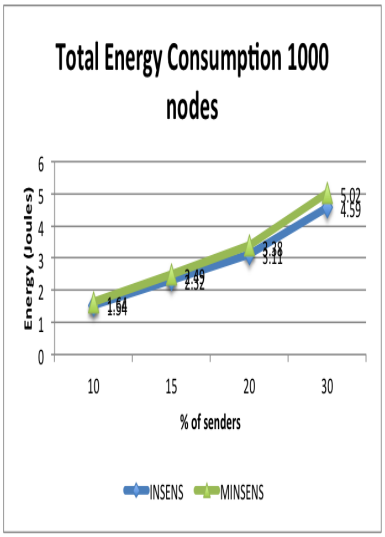

Figure 5.18 Network energy consumption with 1000 nodes

The network's total energy consumption is a direct match of the sums of the individual consumptions, so it comes as no surprise to verify a very similar result.

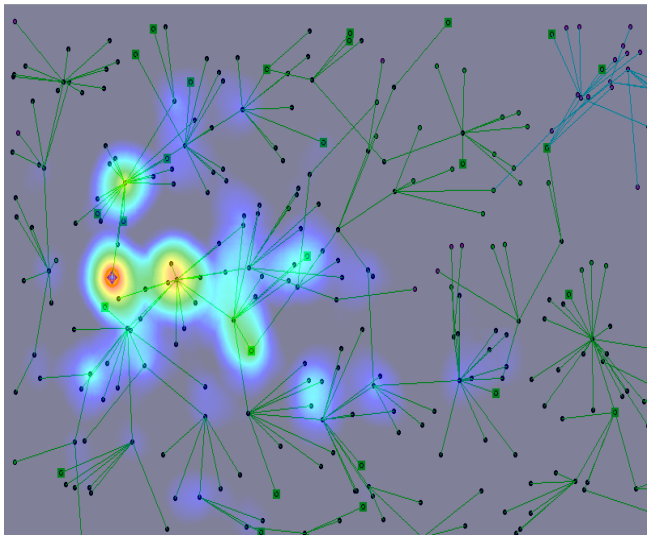

Figure 5.19 Insens heat map with 300 nodes

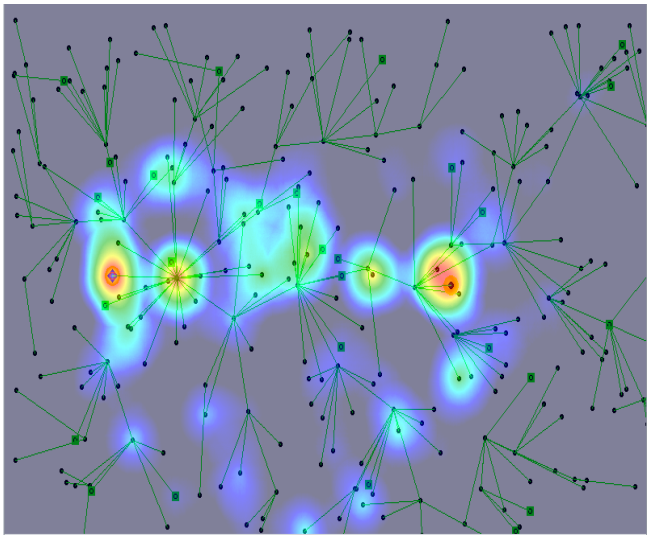

Figure 5.20 MINSENS heat map with 300 nodes

These figures show the heat (and consequently, energy) dispersion through the network. The MINSENS map confirms the lower energy consumption distributed in more even way.

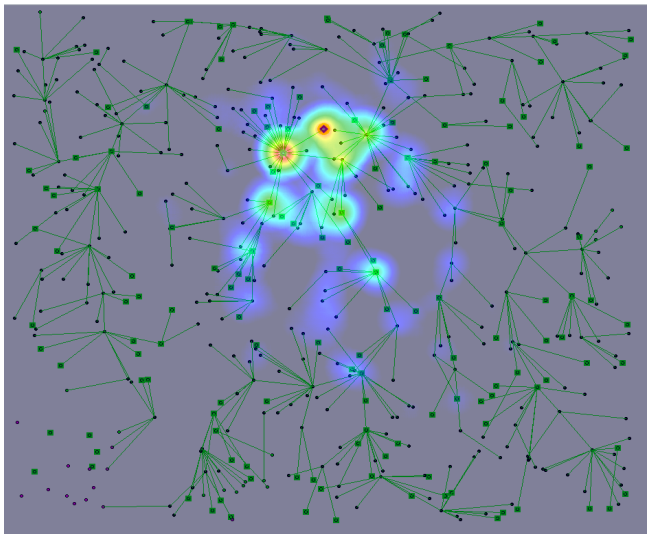

Figure 5.21 Insens heat map with 500 nodes

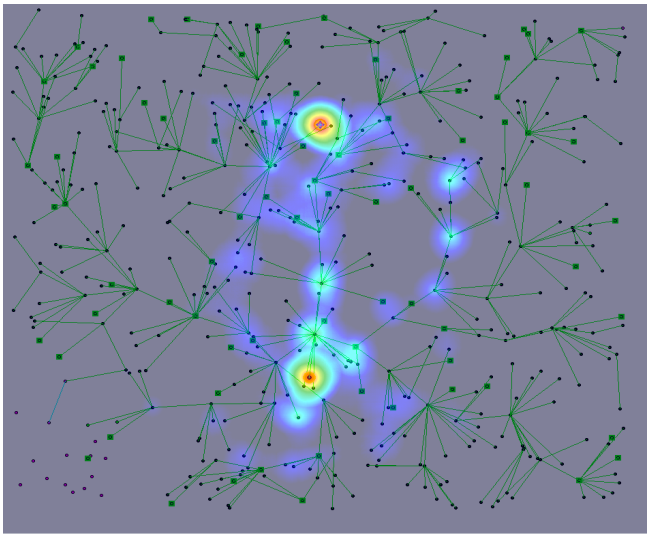

Figure 5.22 MINSENS heat map with 500 nodes

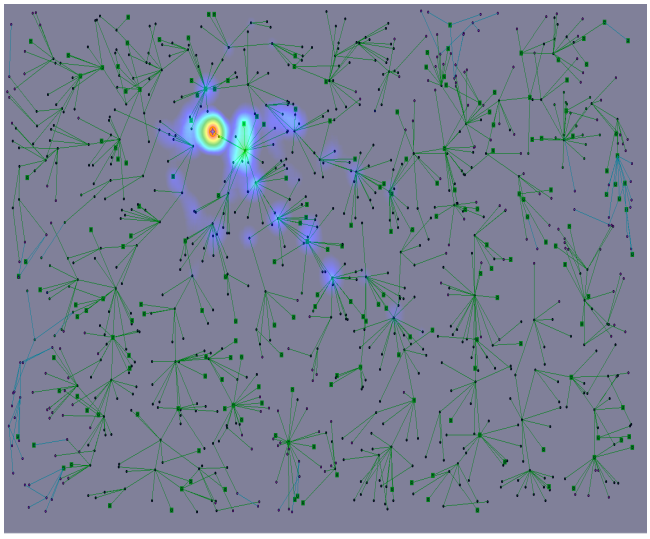

Figure 5.23 Insens heat map with 1000 nodes

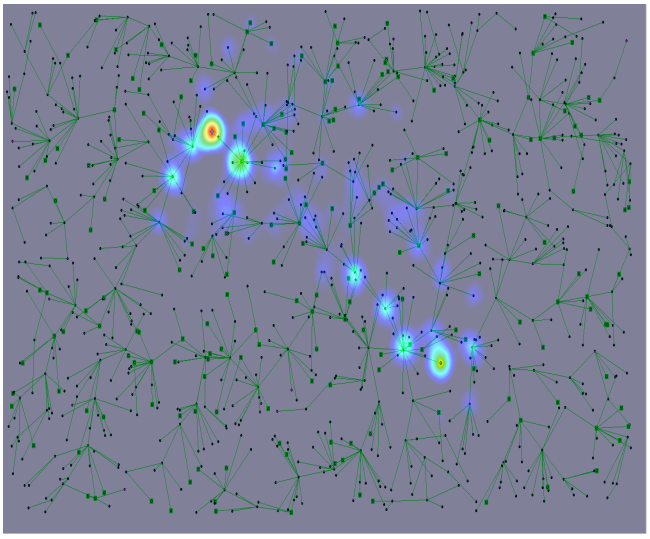

Figure 5.24 MINSENS heat map with 1000 nodes

5.2.5 Number of routes

In this test we used the following test parameters:

parameter	setting
number of stable nodes	250/426/680 - INSENS | 297/490/996 MINSENS
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	1 -INSENS | 2 - MINSENS
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

Results The observed results were the follow:

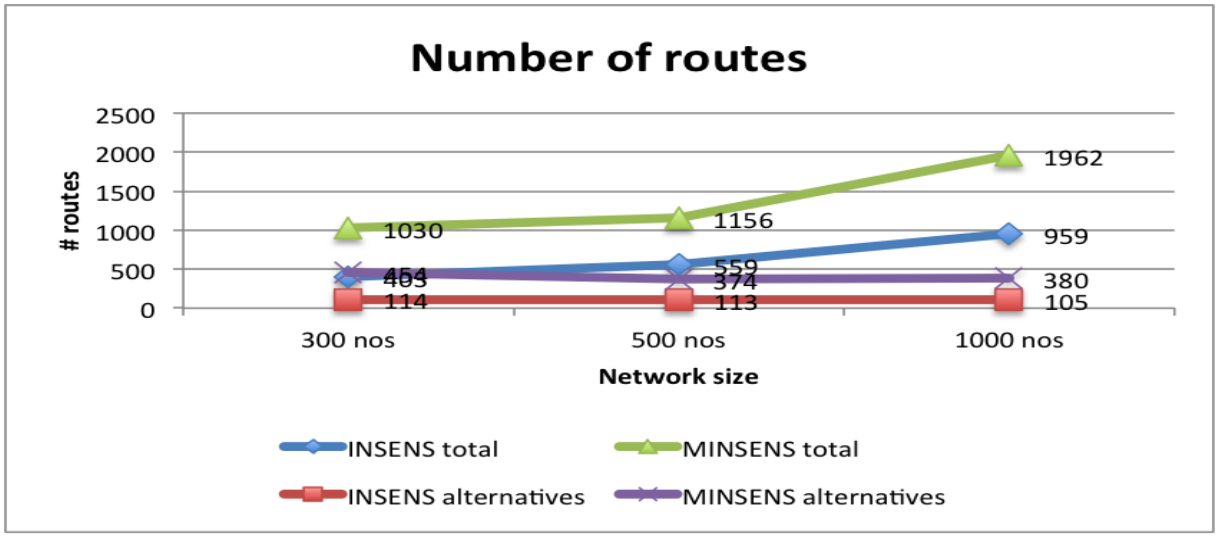

Figure 5.25 Number of routes

The number of generated routes is clearly superior on the MINSENS protocol, accounting in every setting for more than the double of routes. It is curious to note that the scaling of the number of routes is similar in both protocols, even though the use of two base stations by MINSENS.

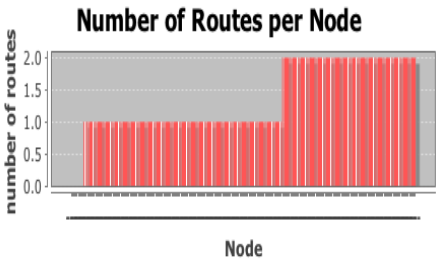

Figure 5.26 Route distribution in INSENS with 300 nodes

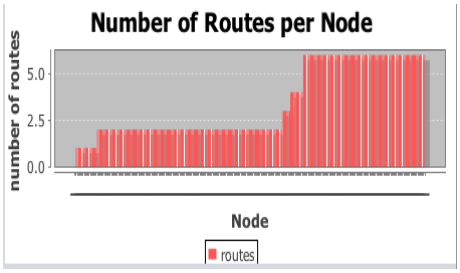

Figure 5.27 Route distribution in MINSENS with 300 nodes

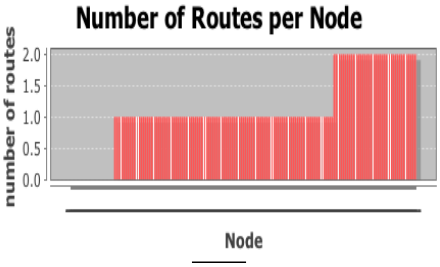

Figure 5.28 Route distribution in INSENS with 500 nodes

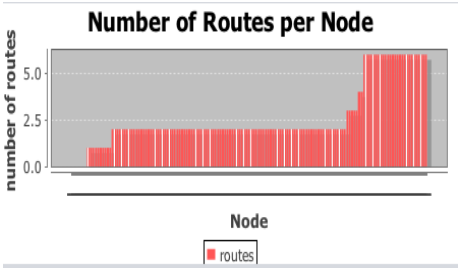

Figure 5.29 Route distribution in MINSENS with 500 nodes

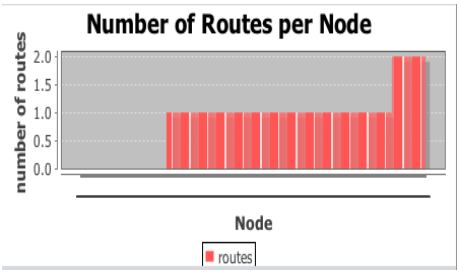

Figure 5.30 Route distribution in INSENS with 1000 nodes

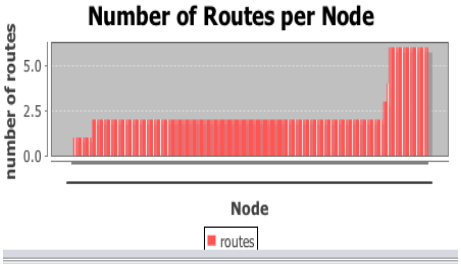

Figure 5.31 Route distribution in MINSENS with 1000 nodes

The route distribution for all nodes is similar between the protocols, with the difference explained by the fact that MINSENS tries to generate 3 routes per base station while INSENS only generates 2 .

5.3 MINSENS transmission scheduling: All base-stations vs Round-Robin

In this section we evaluate the difference between the use of two transmission scheduling settings, the round-robin and the all base stations. The first one makes each sender node distribute its messages through all the available routes in a evenly manner. The second makes full use of all routes for every message sent.

5.3.1 Connectivity

In this test we used the following test parameters:

parameter	setting
number of stable nodes	299/490/996 - All | 299/479/982 RoundRobin
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	2 - All | 2 - RoundRobin
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

Results The observed results were the follow:

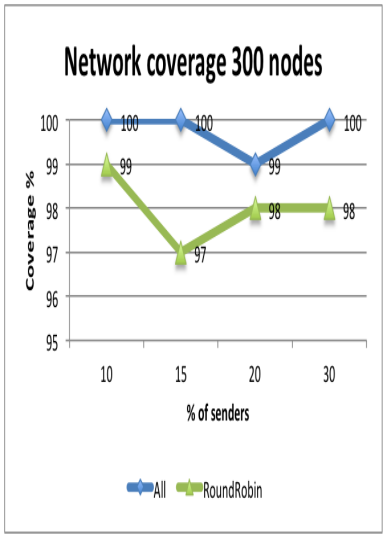

Figure 5.32 Network coverage with 300 nodes

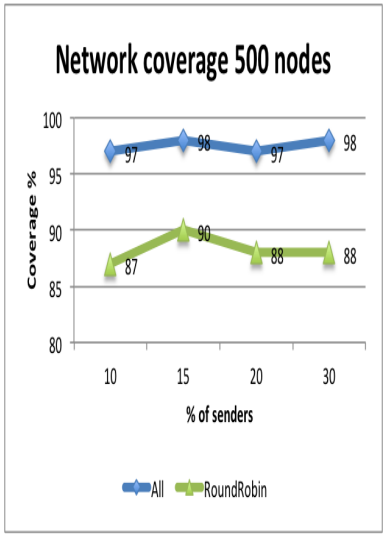

Figure 5.33 Network coverage with 500 nodes

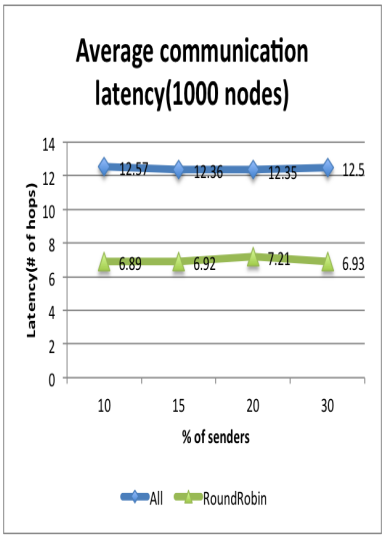

Figure 5.34 Network coverage with 1000 nodes

The all base stations scheduling obtains a higher network connectivity, which is simply due to the constant use of all forwarding resources which minimizes the chance of loosing a

message. Given that the used test settings send one message per node with no retransmission, a message loss translates into a disconnected node. The all base stations scheduling also scales better as can be noted by the difference of only 5% in network coverage from the 300 nodes to the 1000 nodes setting, where the round robin scheme looses about 18%.

5.3.2 Reliability

In this test we used the following test parameters:

parameter	setting
number of stable nodes	299/490/996 - All | 299/479/982 RoundRobin
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	2 - All | 2 - RoundRobin
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

Results The observed results were the follow:

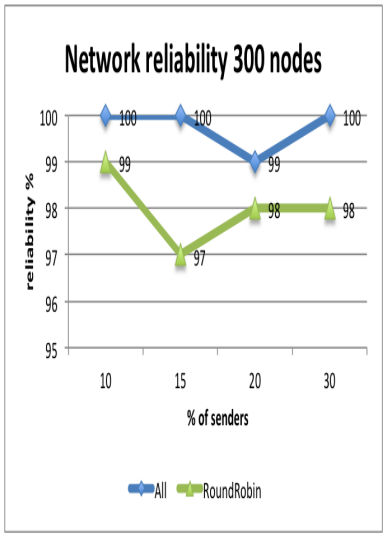

Figure 5.35 Network reliability with 300 nodes

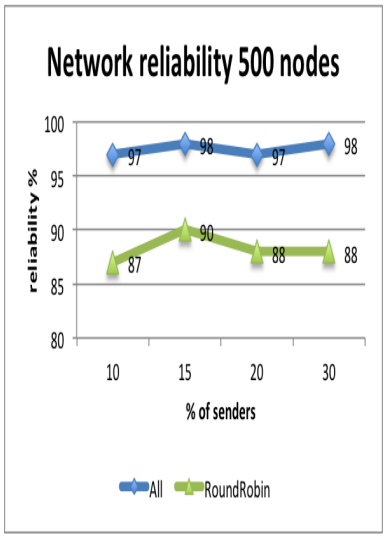

Figure 5.36 Network reliability with 500 nodes

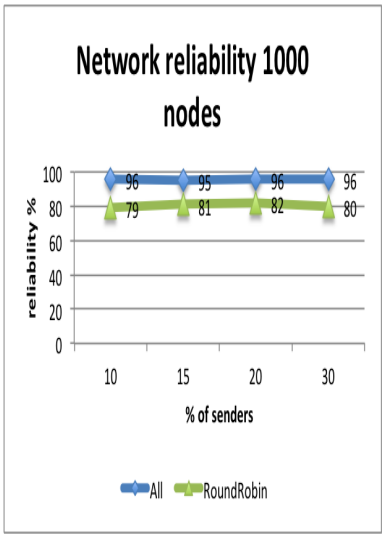

Figure 5.37 Network reliability with 1000 nodes

The All base stations scheduling achieves a higher communications' reliability, again due to the use of all routes which minimizes the odds of loosing one message.

5.3.3 Latency

In this test we used the following test parameters:

parameter	setting
number of stable nodes	299/490/996 - All | 299/479/982 RoundRobin
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	2 - All | 2 - RoundRobin
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

Results The observed results were the follow:

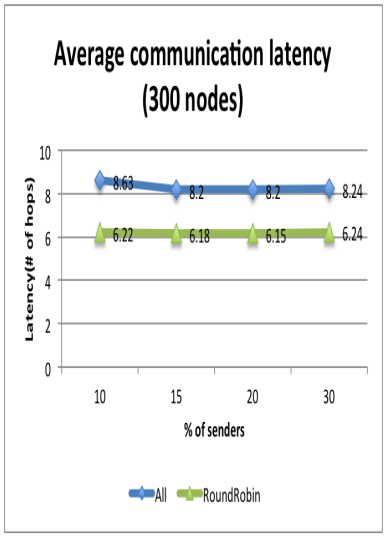

Figure 5.38 Network latency with 300 nodes

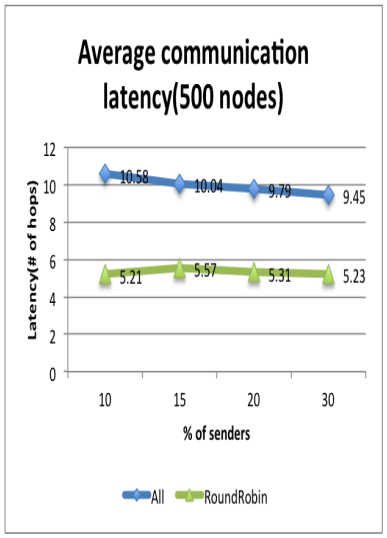

Figure 5.39 Network latency with 500 nodes

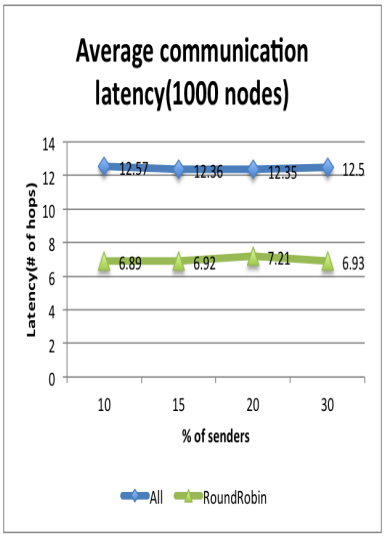

Figure 5.40 Network latency with 1000 nodes

The average communication latency is higher in the All base stations strategy, for it pays the price of using all routes simultaneously, which makes it constantly use the longer ones penalizing the average hop count. The RoundRobin gets a more balanced result.

5.3.4 Energy

In this test we used the following test parameters:

parameter	setting
number of stable nodes	299/490/996 - All | 299/479/982 RoundRobin
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	2 - All | 2 - RoundRobin
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

The observed results were the follow:

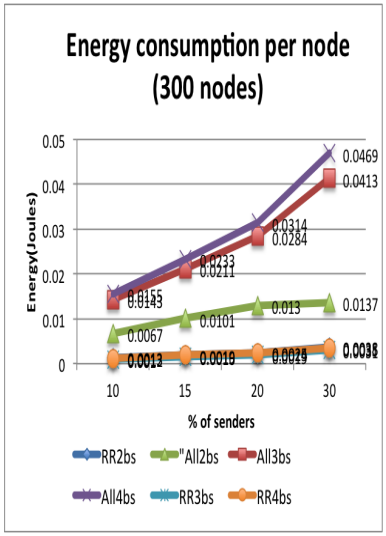

Figure 5.41 Energy per node with 300 nodes

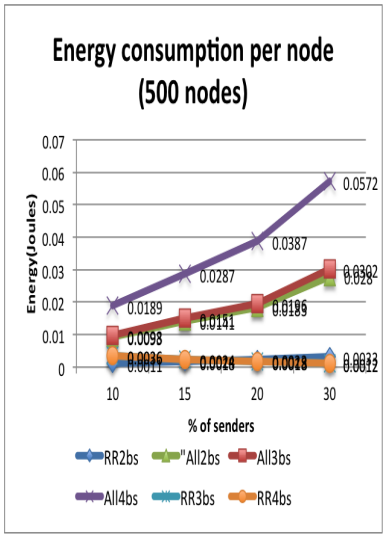

Figure 5.42 Energy per node with 500 nodes

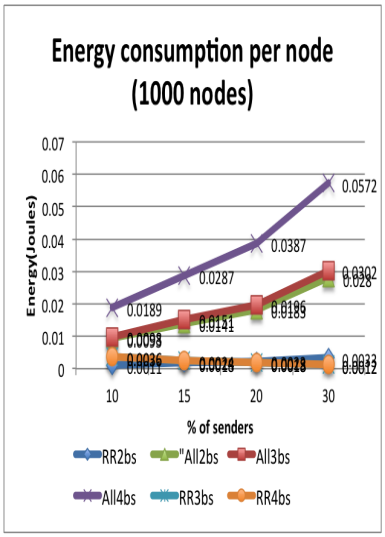

Figure 5.43 Energy per node with 1000 nodes

The energy consumption per node is higher using the all base stations scheduling, which is only natural given the burden to forward all messages through every route. The RoundRobin method is much lighter in terms of energy consumption, making a significant difference of up to $1/7$ of the energy consumption.

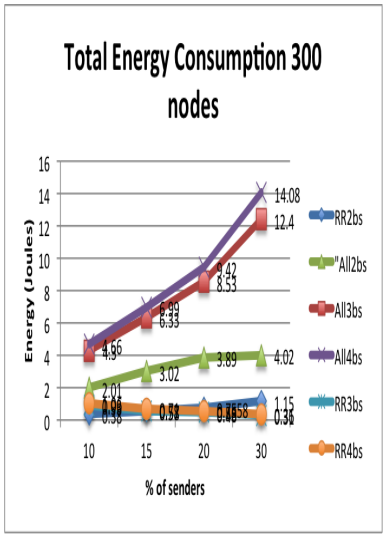

Figure 5.44 Energy per node with 300 nodes

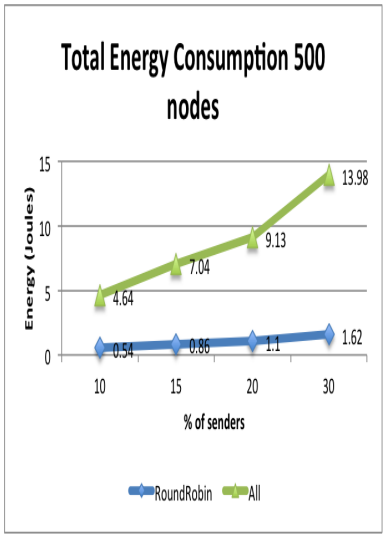

Figure 5.45 Energy per node with 500 nodes

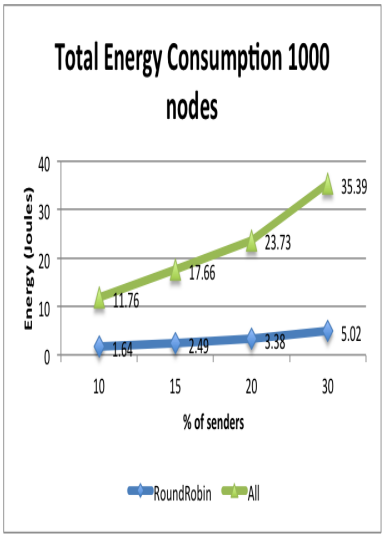

Figure 5.46 Energy per node with 1000 nodes

The network's total energy consumption is a direct match of the sums of the individual consumptions. There is however an interesting result in the 500 and 1000 nodes topologies where energy consumption trend steeps for the All base stations scheduling.

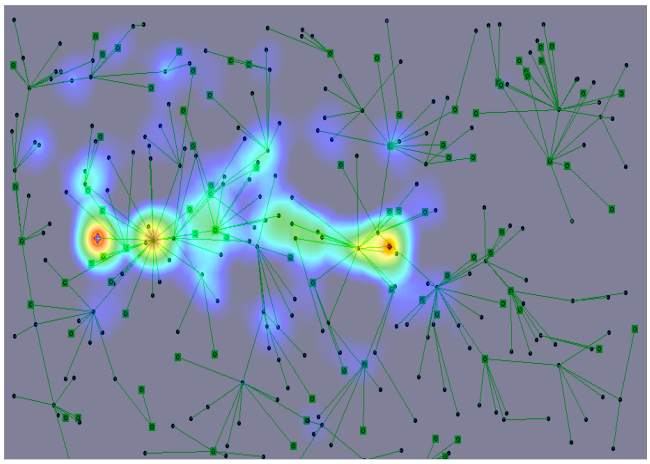

Figure 5.47 All base stations heat map 300 nodes

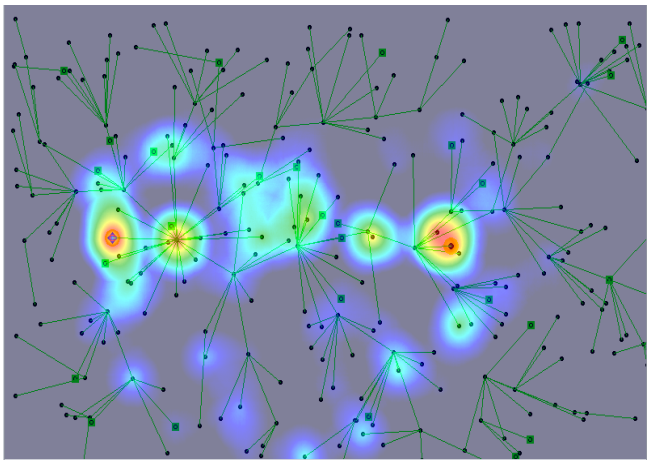

Figure 5.48 Round robin heat map 300 nodes

These figures show the heat (and consequently, energy) dispersion through the network. The All base stations scheduling shows significant heat dispersion, specially in the 500 node topology.

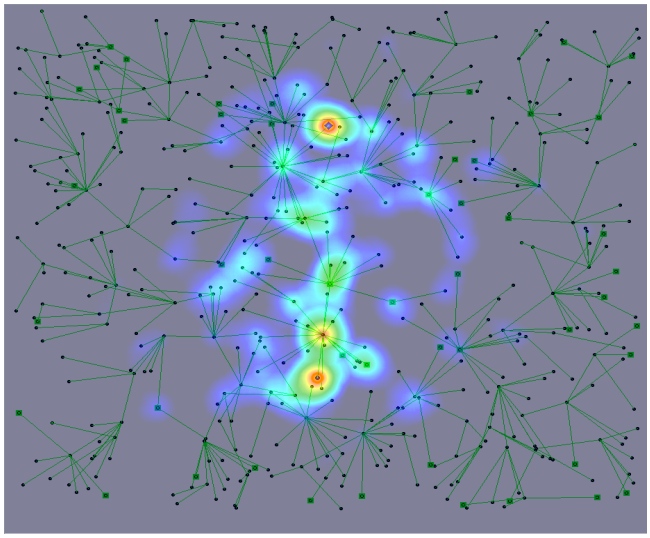

Figure 5.49 All base stations heat map 500 nodes

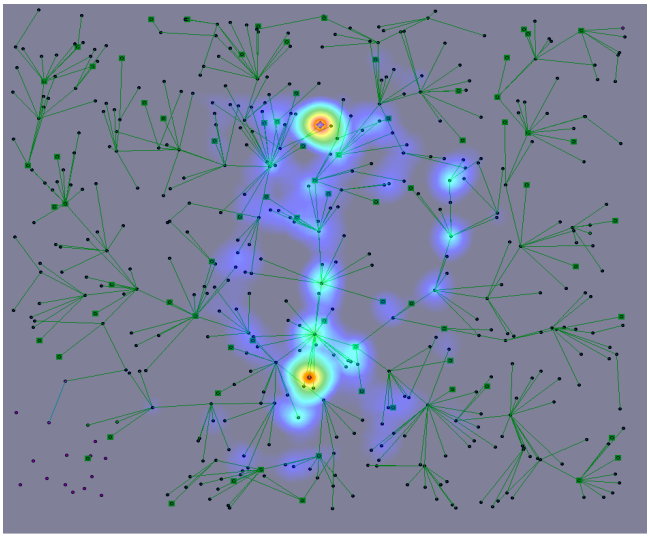

Figure 5.50 Round robin heat map 500 nodes

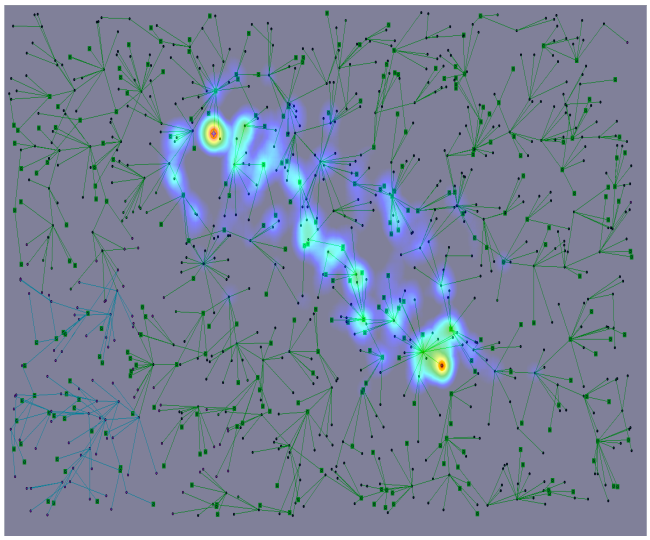

Figure 5.51 All base stations heat map 1000 nodes

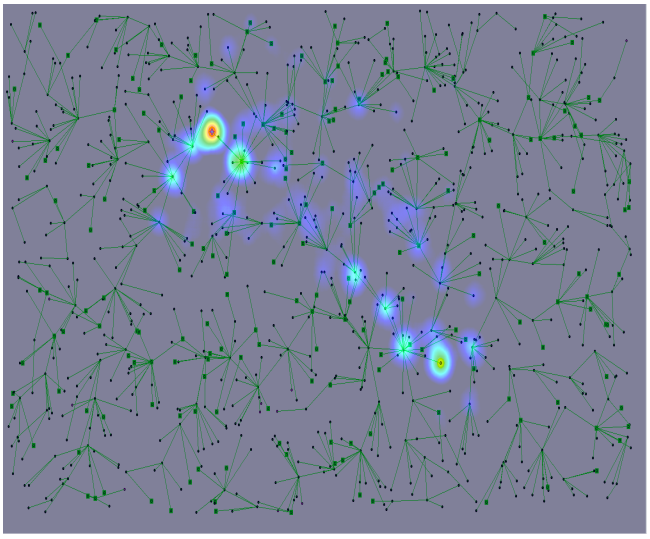

Figure 5.52 Round robin heat map 1000 nodes

5.4 MINSENS scaling: number of base stations

In this section we evaluate the consequences of an increase in the number of base stations.

5.4.1 Connectivity

In this test we used the following test parameters:

parameter	setting
number of stable nodes	299/490/996 - All2bs | 297/487/997 - All3bs | 294/496/996 - All4bs | 299/479/982 RR2bs | 297/494/966 RR3s | 299/478/994 RR4bs
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	2 to 4
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

Results The observed results were the follow: As seen on the graph, the results on a 300

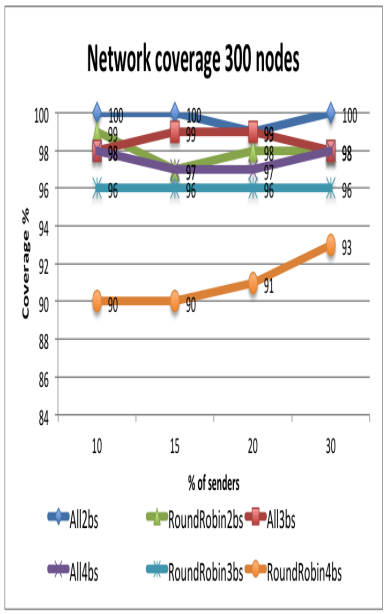

Figure 5.53 Network coverage with 300 nodes

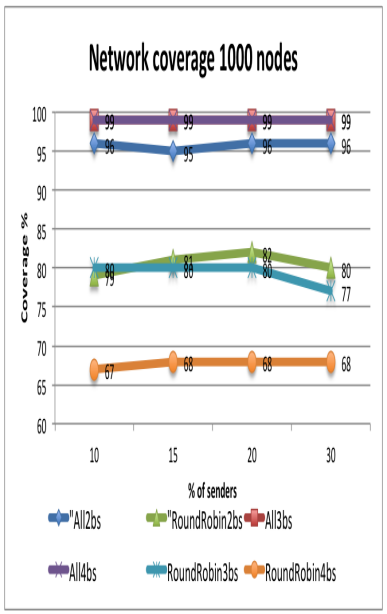

Figure 5.54 Network coverage with 500 nodes

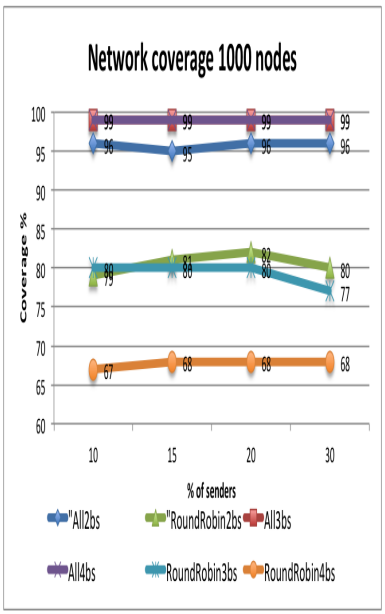

Figure 5.55 Network coverage with 1000 nodes

node topology are very similar between settings, with only the Round robin scheduling using

4 routes having a lower result. On the 500 and 1000 node topologies, we can again see the trend of Round robing scheduling policy not performing as well as the all base stations policy, although it scales better with the raise of base stations number.

5.4.2 Reliability

In this test we used the following test parameters:

parameter	setting
number of stable nodes	299/490/996 - All2bs | 297/487/997 - All3bs | 294/496/996 - All4bs | 299/479/982 RR2bs | 297/494/966 RR3s | 299/478/994 RR4bs
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	2 to 4
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

Results The observed results were the follow:

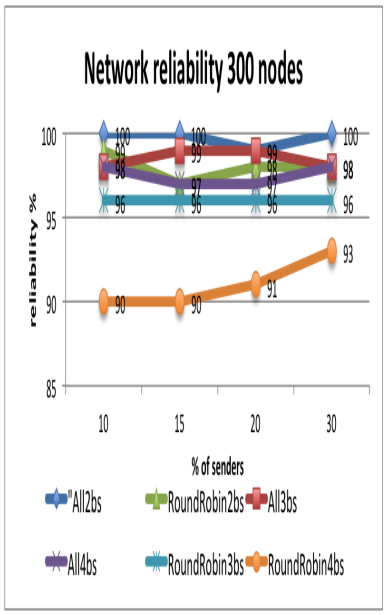

Figure 5.56 Network reliability with 300 nodes

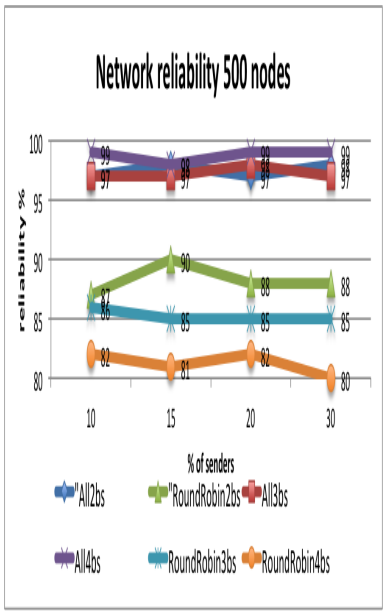

Figure 5.57 Network reliability with 500 nodes

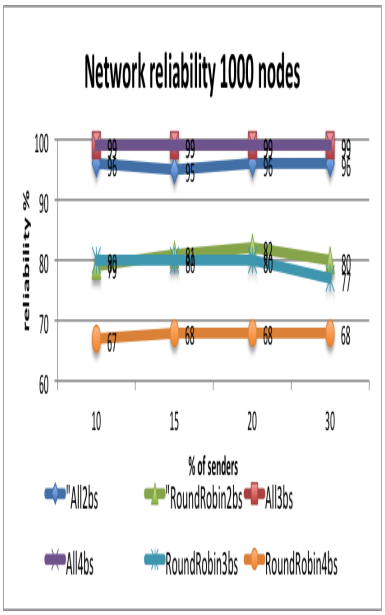

Figure 5.58 Network reliability with 1000 nodes

As seen on the graph, the results are pretty similar on both topologies, with the Round robin policies having the worse results, specially the 3 and 4 base station versions. The increase in the number of base stations translates into a decrease in round robin reliability, as opposed to all base stations that increases the reliability.

5.4.3 Latency

In this test we used the following test parameters:

parameter	setting
number of stable nodes	299/490/996 - All2bs | 297/487/997 - All3bs | 294/496/996 - All4bs | 299/479/982 RR2bs | 297/494/966 RR3s | 299/478/994 RR4bs
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	2 to 4
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

Results The observed results were the follow:

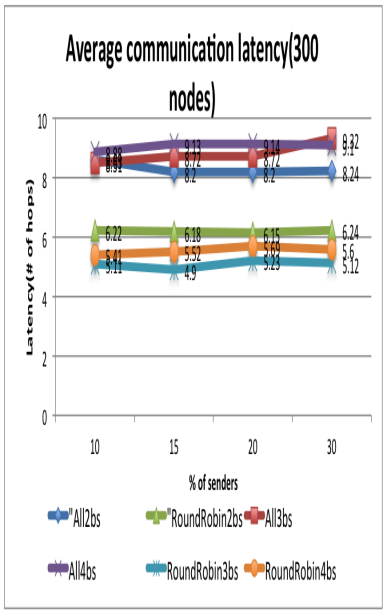

Figure 5.59 Network latency with 300 nodes

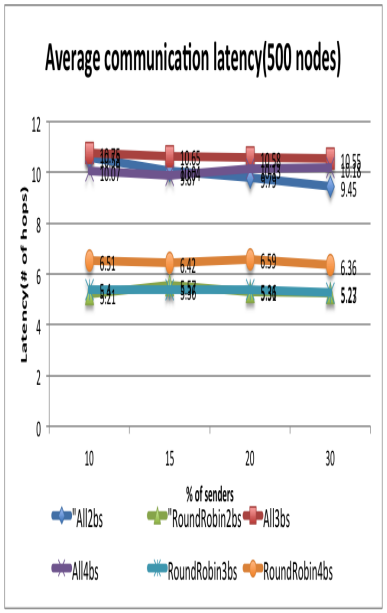

Figure 5.60 Network latency with 500 nodes

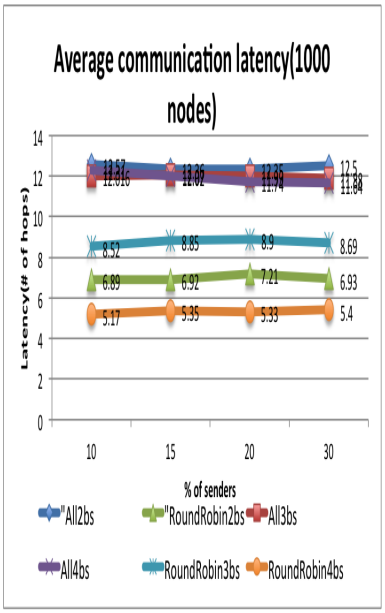

Figure 5.61 Network latency with 1000 nodes

The obtained results show that on the bigger topology setting (1000 nodes), the bigger the number of base stations available the better is the average communication latency. On the other settings, the data seems to indicate that the use of 3 base stations favors the latency, which is represented by the 4 base station round robing consistently having worse results than the 3 base station version on the 300 and 500 node topologies.

5.4.4 Energy

In this test we used the following test parameters:

parameter	setting
number of stable nodes	299/490/996 - All2bs | 297/487/997 - All3bs | 294/496/996 - All4bs | 299/479/982 RR2bs | 297/494/966 RR3s | 299/478/994 RR4bs
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	2 to 4
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

The observed results were the follow:

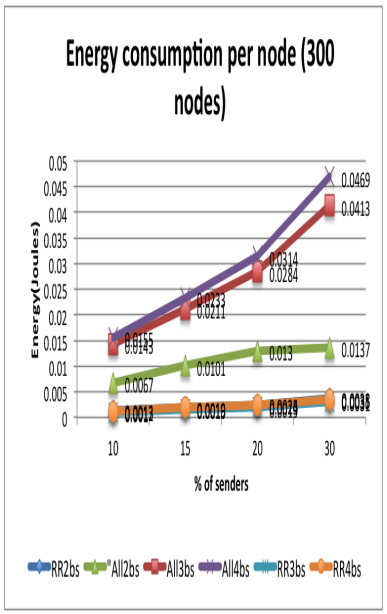

Figure 5.62 Energy per node with 300 nodes

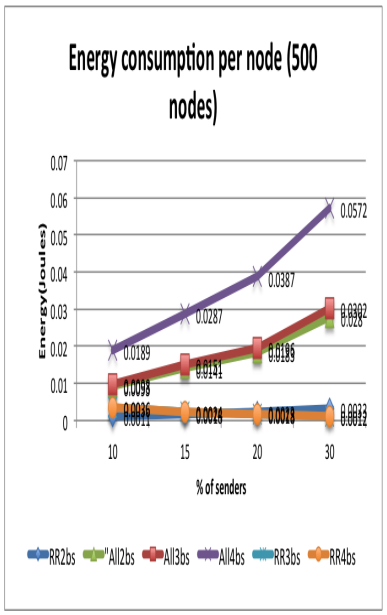

Figure 5.63 Energy per node with 500 nodes

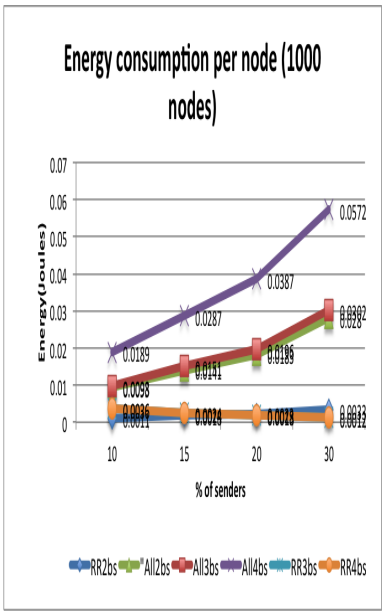

Figure 5.64 Energy per node 1000 nodes

The energy consumption per node is higher in the all base stations scheduling according to the number of used base stations. This makes sense given that the more base stations, the more routes and therefore a bigger load of every node to forward his neighbor's messages. On the round robin policy, the data shows that the number of base stations does not significantly impact the energy consumption.

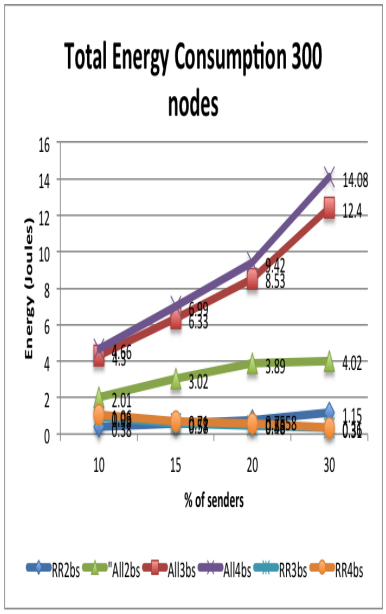

Figure 5.65 Energy per node with 300 nodes

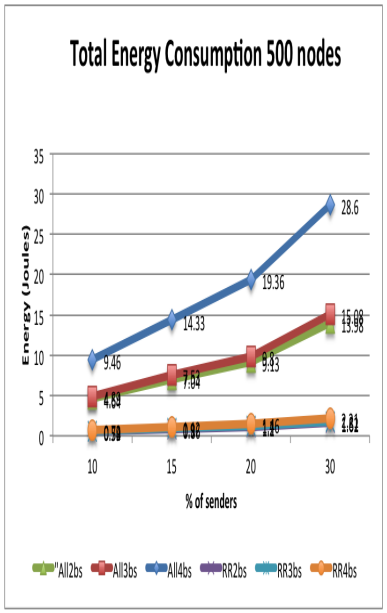

Figure 5.66 Energy per node with 500 nodes

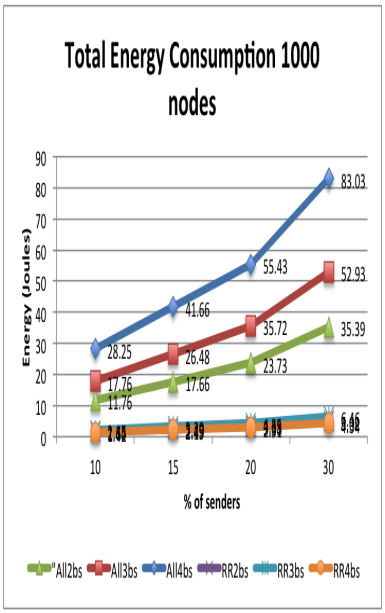

Figure 5.67 Energy per node 1000 nodes

The network's total energy consumption is a direct match of the sums of the individual consumptions. There is however an interesting result in the 500 node topology where energy consumption trend steeps for the All base stations scheduling.

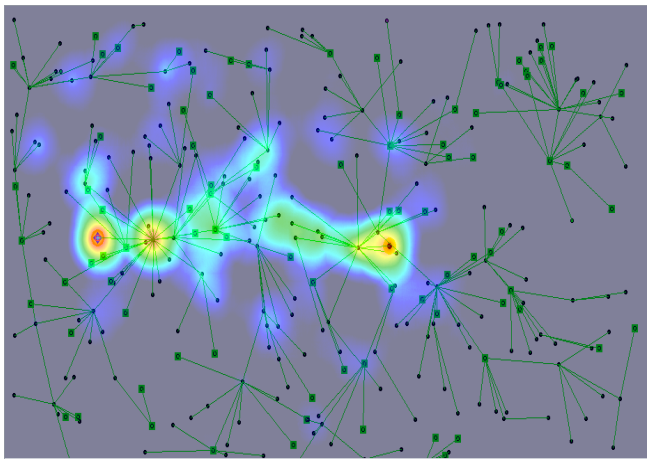

Figure 5.68 All base stations heat map 300 nodes

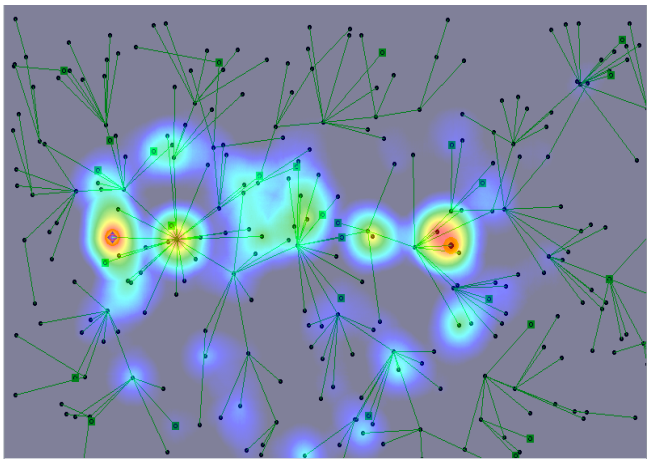

Figure 5.69 Round robin heat map 300 nodes

These figures show the heat (and consequently, energy) dispersion through the network. The All base stations scheduling shows significant heat dispersion, specially in the 500 and 1000 nodes topologies.

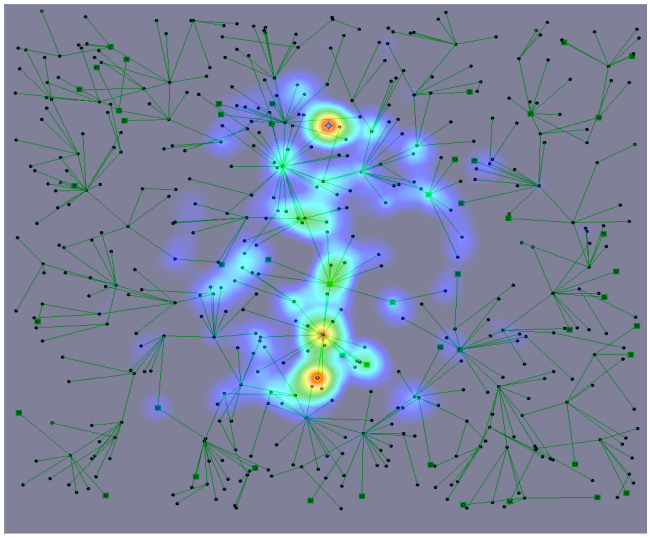

Figure 5.70 All base stations heat map 500 nodes

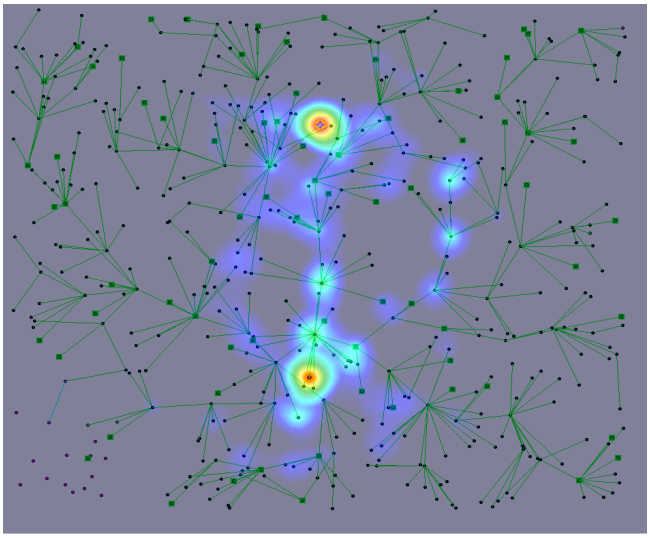

Figure 5.71 Round robin heat map 500 nodes

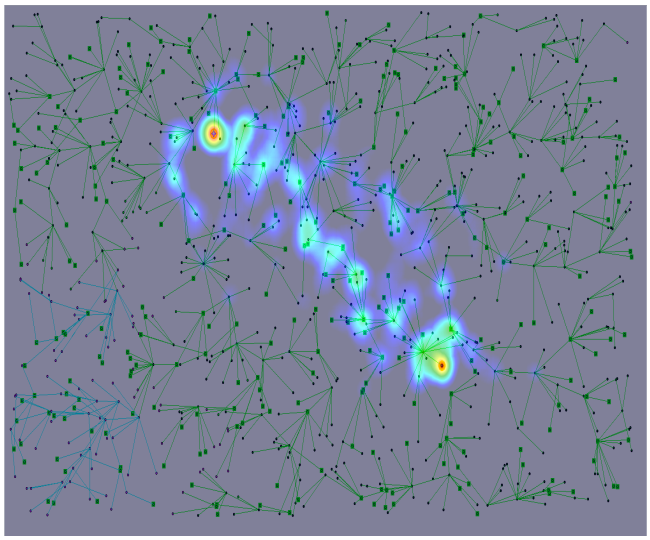

Figure 5.72 All base stations heat map 1000 nodes

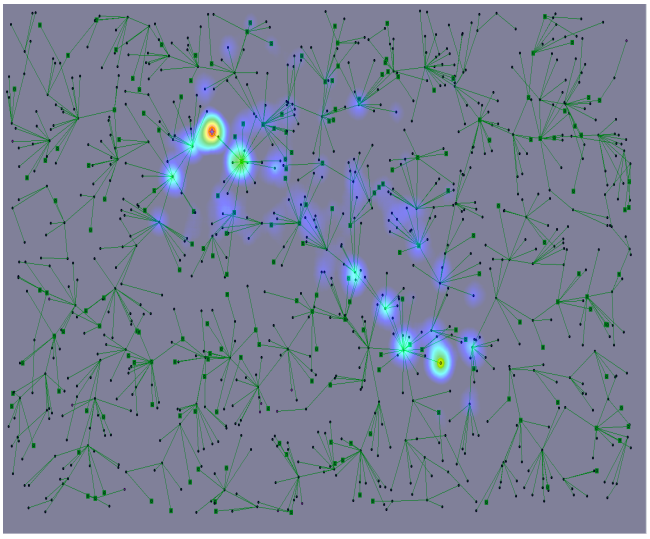

Figure 5.73 Round robin heat map 1000 nodes

5.4.5 Number of routes

In this test we used the following test parameters:

parameter	setting
number of stable nodes	299/490 - All2bs | 297/487 - All3bs | 294/496 - All4bs | 299/479 RR2bs | 297/494 RR3s | 299/478 RR4bs
number of observations	10
number/percentage of sender nodes	10 to 30%
number of receiver nodes(base stations)	2 to 4
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

The observed results were the follow:

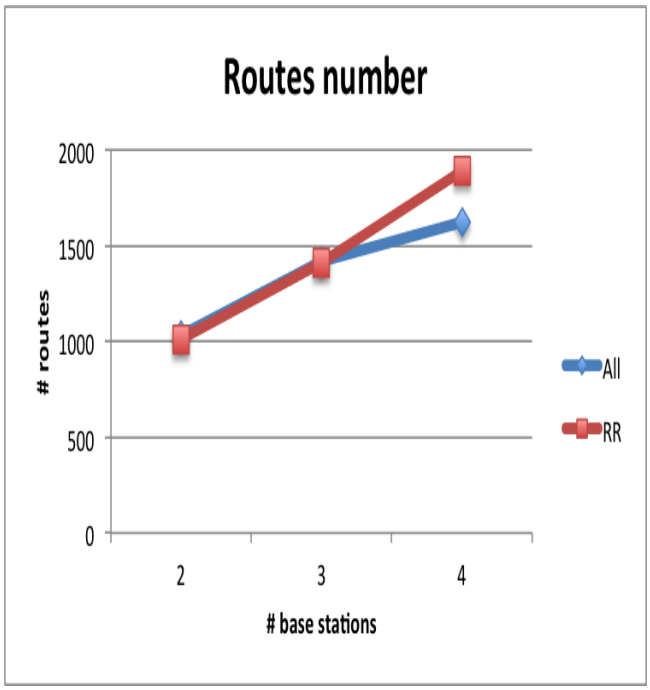

Figure 5.74 Number of routes

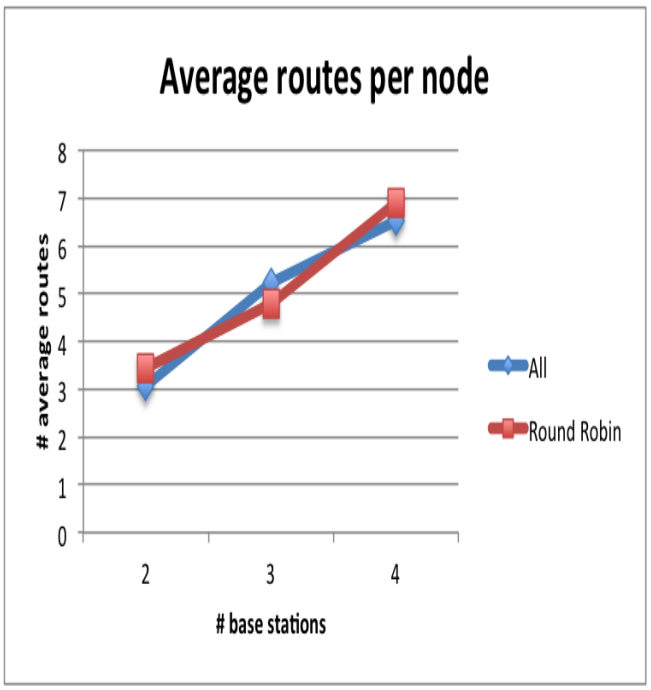

Figure 5.75 average routes per node

The total number of generated routes scales in a almost linear way with the number of base stations. The average number of routes per node scales in a faster way, but still in a linear way.

5.5 Attacks Evaluation

In this section we evaluated the response of both analyzed protocols when under attack by a percentage of the network nodes acting maliciously.

5.5.1 Connectivity

In this test we used the following test parameters:

parameter	setting
number of stable nodes	297 - All2bs | 296 - All3bs | 296 - All4bs | 299 RR2bs | 294 RR3s | 298 RR4bs
number of observations	10
number/percentage of sender nodes	20%
number of receiver nodes(base stations)	2 to 4
number/percentage of attacker nodes	0 to 30%
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	blackhole attack

Results The observed results were the follow:

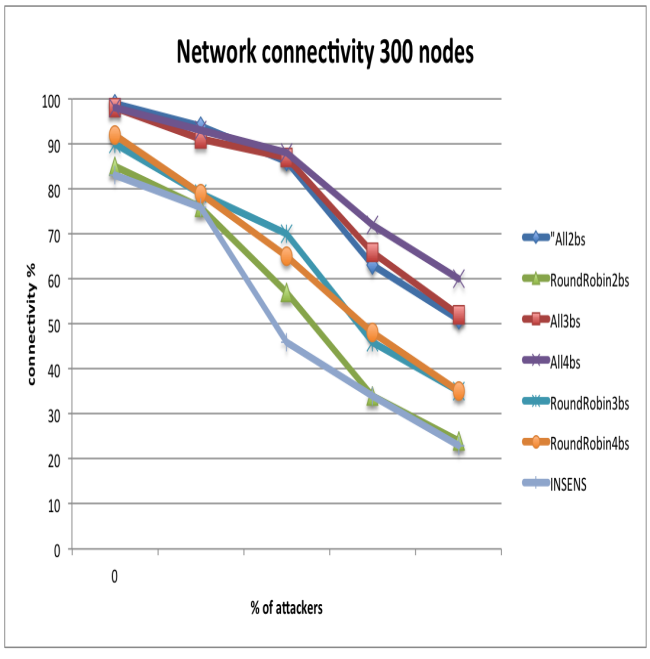

Figure 5.76 Attack results on various settings

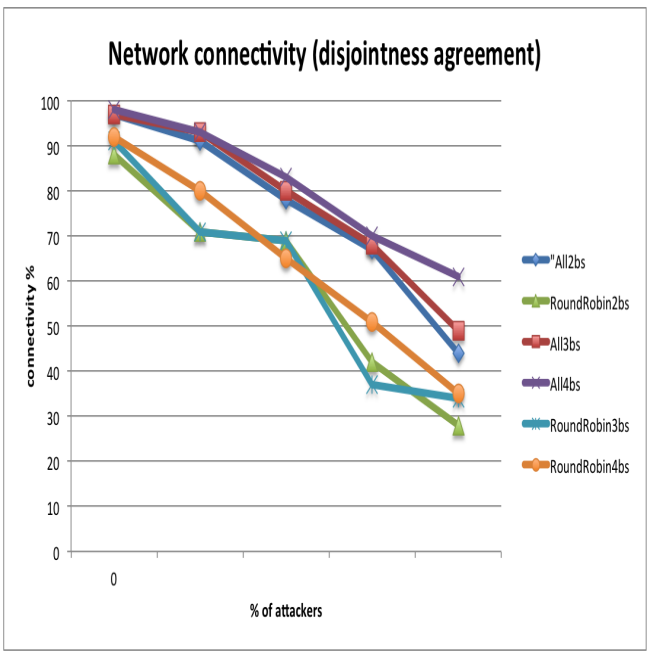

Figure 5.77 Route disjointness agreement working

The results show that the MINSENS protocol has a far greater resilience against attacks

when compared to INSENS[21], specially using the all base stations scheduling policy. The route disjointness mechanism causes a slight improvement on the results, specially in the round robin settings.These results are very similar to the reliability ones, which will be compared in further detail bellow.

5.5.2 Reliability

In this test we used the following test parameters:

parameter	setting
number of stable nodes	297 - All2bs | 296 - All3bs | 296 - All4bs | 299 RR2bs | 294 RR3s | 298 RR4bs
number of observations	10
number/percentage of sender nodes	20%
number of receiver nodes(base stations)	2 to 4
number/percentage of attacker nodes	0 to 30%
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	blackhole attack

Results The observed results were the follow:

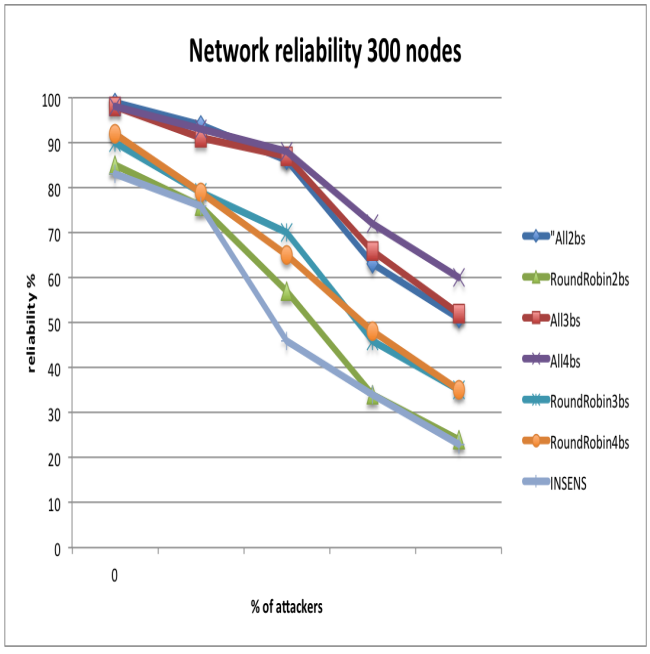

Figure 5.78 Attack results on various settings

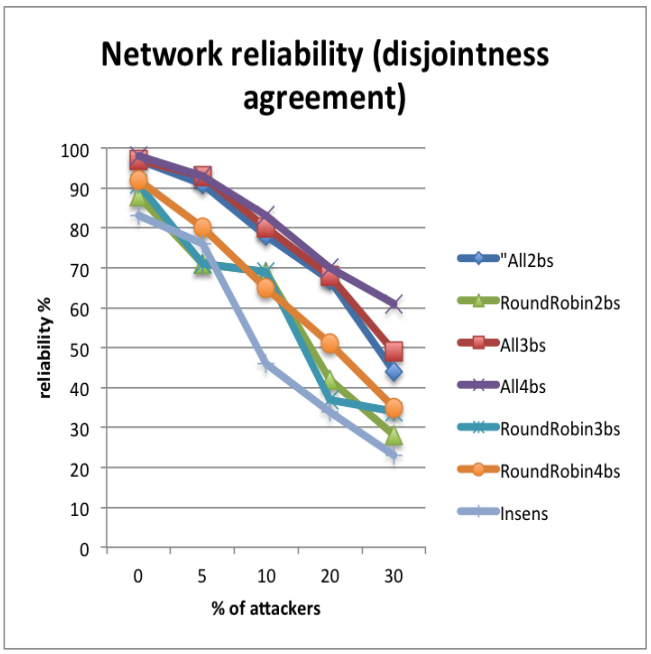

Figure 5.79 Route disjointness agreement working

These results are similar to the connectivity given that the tests were made with no retransmissions and only one message per sender. This was made to simulate the worst case scenario for the routing protocols. Next we show the impact of the route disjointness mechanism in each base station setting.

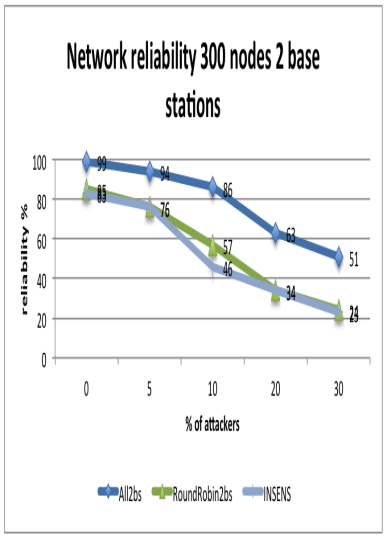

Figure 5.80 Reliability 2 base stations

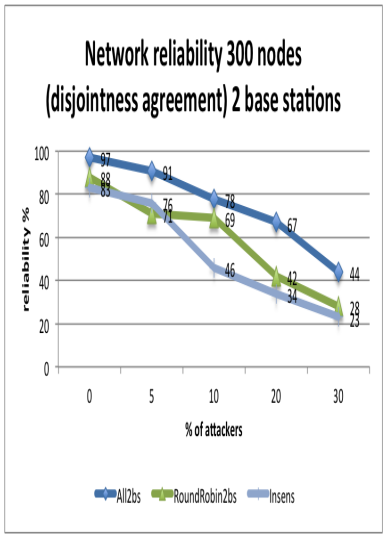

Figure 5.81 Reliability 2 base stations route disjointness

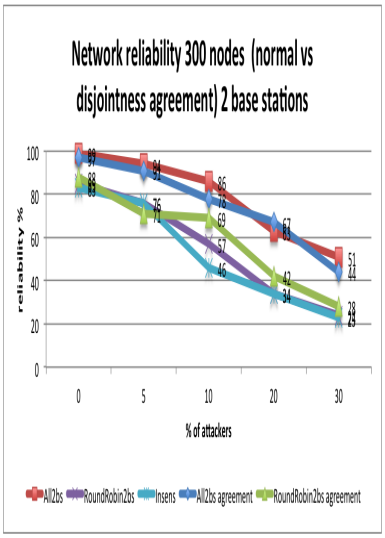

Figure 5.82 Reliability 2 base stations comparison

In this setting, the route disjointness improves the reliability of the MINSENS protocol with the round robin policy by 12%, while showing inconclusive results on the all base stations policy.

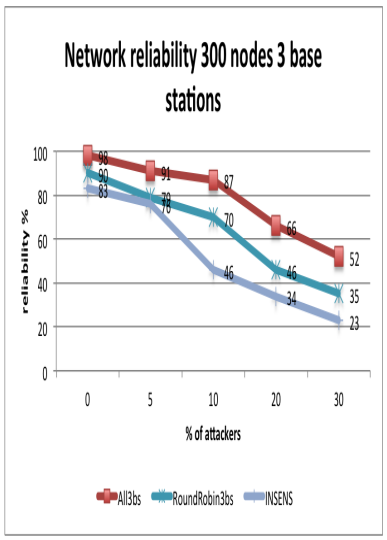

Figure 5.83 Reliability 3 base stations

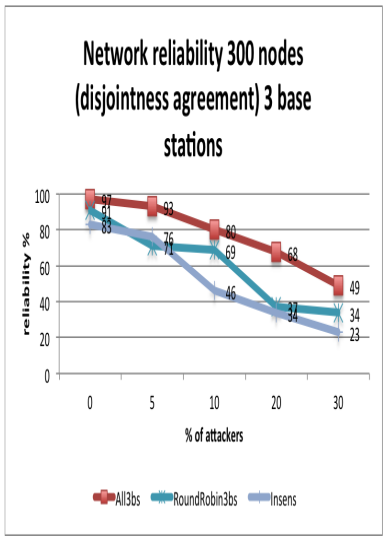

Figure 5.84 Reliability 3 base stations route disjointness

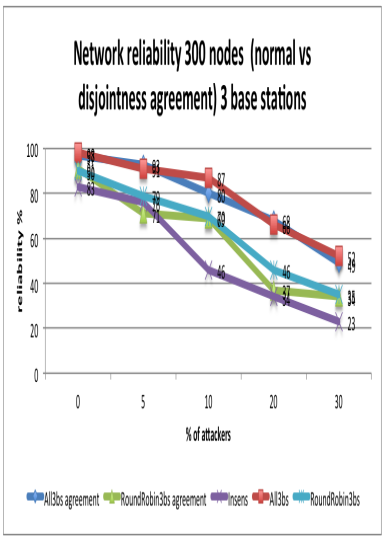

Figure 5.85 Reliability 3 base stations comparison

With the 500 nodes topology, the data show no significant improvement by using the route disjointness algorithm on both scheduling policies.

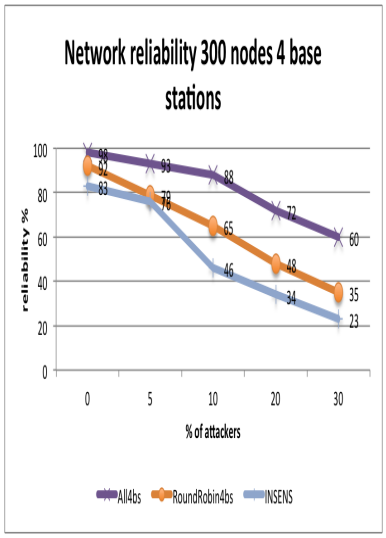

Figure 5.86 Reliability 4 base stations

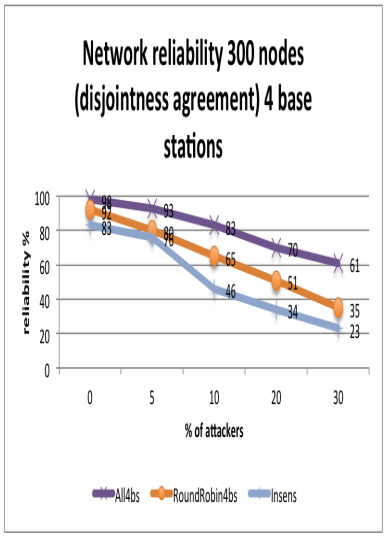

Figure 5.87 Reliability 4 base stations route disjointness

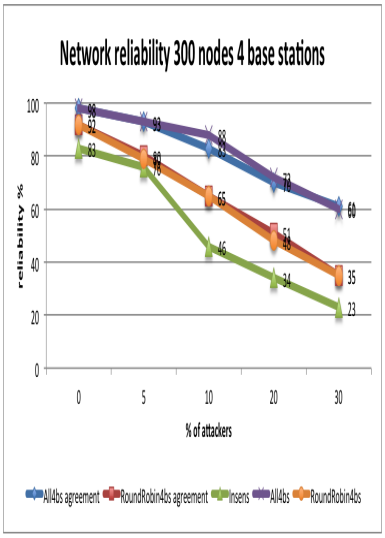

Figure 5.88 Reliability 4 base stations comparison

Similar to the 500 nodes topology, in the 1000 nodes topology the use of the route disjointness agreement made no significant difference on the end results.

By analyzing the results from all three topologies, we can conclude that the implemented route agreement protocol improves the MINSENS reliability on smaller topologies (300 nodes), while having a neutral effect on bigger topologies (500 and 1000 nodes). This can in part be explained by the decrease in the number of available routes that this agreement produces, making it unclear which side of the trade-off between more routes not totally disjoint and less routes totally disjoint produces more benefits to the MINSENS performance.

5.5.3 Route disjointness agreement: number of routes

The influence of the route disjointness agreement on the number of nodes was measured using the following parameters:

parameter	setting
number of stable nodes	297 - All2bs | 296 - All3bs | 296 - All4bs | 299 RR2bs | 294 RR3s | 298 RR4bs
number of observations	10
number/percentage of sender nodes	20%
number of receiver nodes(base stations)	2 to 4
number/percentage of attacker nodes	0 to 30%
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	blackhole attack

Results The observed results were the follow:

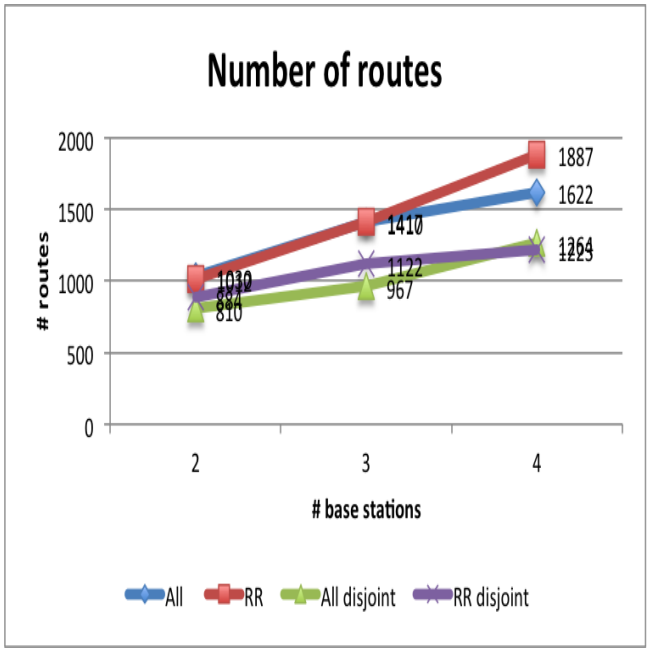

Figure 5.89 Number of routes on various settings

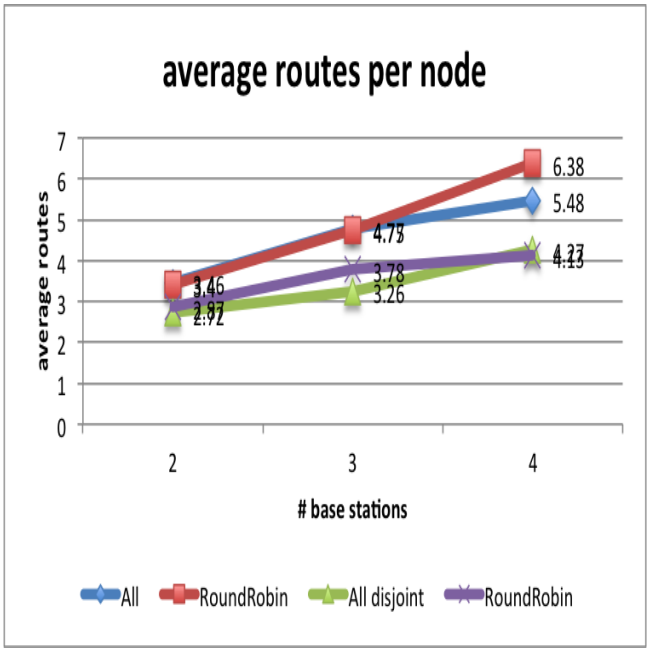

Figure 5.90 Average routes per node

When comparing both the total amount of routes and the average routes per node, we can see that the disjointness agreement protocol causes a quite significant drop in the route numbers, which might help to explain its moderate improvement on attack resilience.

5.6 Byzantine Agreement

In order to test the Byzantine agreement implementation, we made a small evaluation of the percentage of consensualized data packets given a certain local and global threshold. That threshold is the minimum number of equal replicas for that agreement to be made. We used the following parameters:

parameter	**setting**
number of stable nodes	296
number of observations	10
number/percentage of sender nodes	20%
number of receiver nodes(base stations)	4
number/percentage of attacker nodes	0
number of different messages sent per node	1
interval between each message transmission	10 seconds
the number of retransmissions	0
type of attack performed by the attacker nodes	none

Results The observed results were the follow:

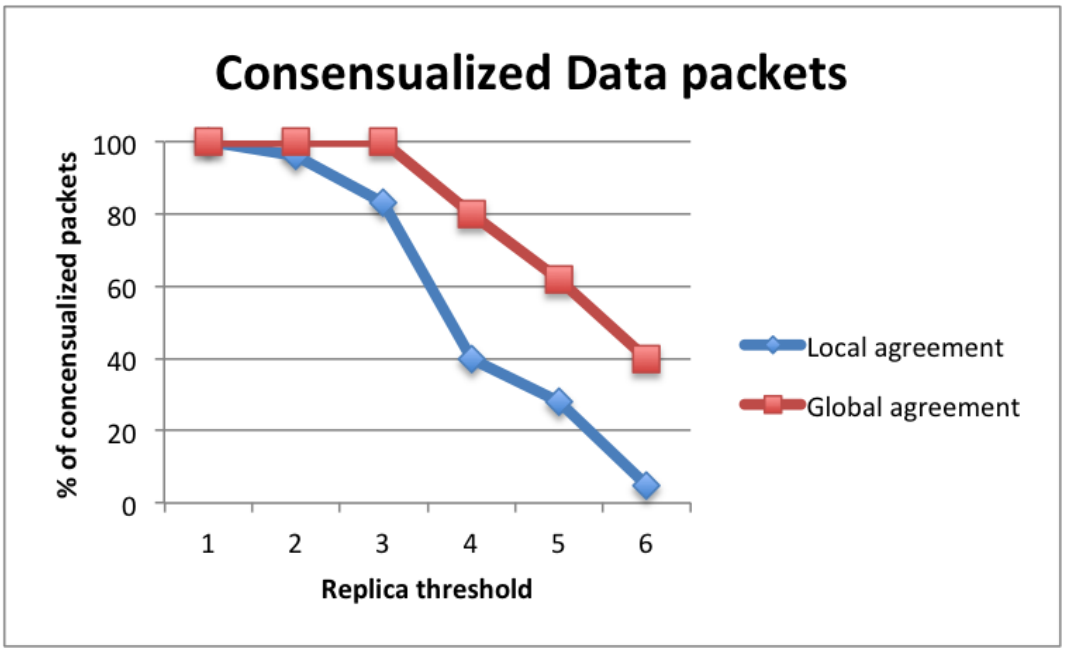

Figure 5.91 Number of routes on various settings

For an average of 6.35 routes per node, we can conclude that the Global agreement provides a significant improvement by using all base stations' information to agree on the original replica.

6. Conclusions and future work

This chapter presents the conclusions from this thesis' work as well as the open issues and possible future work that we believe it is important in the context of this dissertation.

6.1 Conclusions

In this thesis a very powerful and flexible simulation environment was used, the Wisenet. It enabled us to simulate a wide variety of situations with highly customizable settings and the support of a very complete measurement mechanism. This enabled us to implement, test and compare one secure routing protocol, the INSENS, as well as the proposed protocol, the MINSENS.

The problem of dealing with internal attacks on WSNs is very relevant and poses a great challenge given the many constraints inherent to the WSNs own nature. The WSNs research community has proposed numerous solutions to deal with this problem, yet the problem of consensus on WSNs is still a open issue. As such, we studied, modeled and implemented a solution based on multi-path routing strategies as a primary pro-active mechanism to mitigate intrusion attacks, complemented with intrusion tolerance properties. These feature two forms of consensus in a not-yet proposed perspective, using randomized consensus techniques to overcome the FLP impossibility result. The implemented consensus mechanisms are a route disjointness agreement and a Byzantine agreement over data-sets, initially captured in the mesh environment auto-organized at the WSN level. These last two aspects are based on a broadcast primitive which was implemented for simulation purposes via simulator, yet is itself an open issue and a interesting research direction.

In conclusion, the main contributions presented are:

- the design, implementation and evaluation of a simulation environment with the possibility to inject a typology of attacks to the routing layer, with relevant consequences to the network resilience and reliability;

- The demonstration and evaluation of INSENS and the proposed MINSENS in the simulation environment created, in order to assess their practical security properties and runtime performance indicators, as well as, the flexibility and rich functionality offered by the simulation tools;

- The in-depth experimental assessment study of the above protocols, according with a relevant set of criteria to support tree-based data dissemination models in randomized non-supervised topologies, supported by intrusion tolerant routing services. Those criteria included the evaluation of metrics such as: energy cost, connectivity conditions, effective

reliability and latency. From the experimental simulation and observations, it is possible to frame a critical analysis study, correlating the observations and observed metrics in different WSN settings, namely: fan-out metrics, number of rounds to achieve certain reliability degrees, degree-distribution metrics, average shorted path metrics versus latency conditions and clustering conditions or effective resilience induced by multi-path multi-base station strategies.

6.2 Future research work directions

From the contributions and experimental evaluation of the thesis, we devise interesting research directions for complementary studies on byzantine consensus models and algorithms applied to intrusion-tolerant WSNs.

In this direction, a critical analysis of completely asynchronous consensus protocols inspired on randomized consensus strategies, performed by sync nodes (or by specially designed internal sensors for internal intrusion tolerant data-consensus at the WSN level), seems to be an interesting research direction.

On the other hand, considering the fact that data-consensus strategies is being widely studied in the context of classical networks, few studies have been conducted in order to solve it in the context of dynamic ad-hoc organized systems and, from our own understanding, it is yet an open field, particularly considering the system model in the background of the thesis.

In the scenario of exploring completely asynchronous and intrusion-tolerant consensus mechanisms for non-uniform data-sets, the problem of establishment of reliable byzantine consensus with unknown participants (namely the BFT-CUP) is a novel field, given the stringent connectivity conditions of WSNs.

This open field is related with a new problem with the additional requirement that participants can perform maliciously (or with an hybrid of malicious and intermittent faulty settings). This interesting direction requires the study of new knowledge bases on specific WSN connectivity conditions in order to solve WSN BFT-CUP problems, under complete asynchrony requirements.

Other future work directions that come directly from the thesis results are:

- The extension of evaluations and experimental evaluations to more scalable WSNs;
- The verification of experimental simulation results, putting the hardware (and real networks) in the loop;
- The extension of data-consensus strategies and mechanisms at the level of special sensors (as super-nodes) running not only as base-stations but as internal nodes or actuators, inside the WSN itself.

Bibliography

[1] *Avrora: scalable sensor network simulation with precise timing*, April 2005.

[2] IEEE std 802.3 - 2005 part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications - section five. Technical report, 2005.

[3] Secure aggregation, location, and cross-layer, in book chapter of security. In *in Security in Sensor Networks, Yang Xiao (Eds*. CRC Press, 2006.

[4] book:security in sensor networks. In *Yang Xiao (Eds*. CRC Press, 2007.

[5] Victor Wen David Culler Adrian Perrig, Robert Szewczyk and J. D. Tygar. Spins: Security protocols for sensor networks. In *Seventh Annual International Conference on Mobile Computing and Networks (MobiCom 2001)*, pages 189–199, Rome, Italy, 2001.

[6] ZigBee Alliance. Zigbee specification. In *Technical Report Document 053474r06*, 2005.

[7] Th. Arampatzis, J. Lygeros, and S. Manesis. A survey of applications of wireless sensors and wireless sensor networks. In *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, pages 719 –724, 2005.

[8] Baruch Awerbuch, David Holmer, Cristina Nita-rotaru, and Herbert Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *in ACM Workshop on Wireless Security (WiSe*, pages 21–30, 2002.

[9] Pillai P. Chook V. Chessa S. Gotta A. Baronti, P. and Y. F. Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. In *IComputer Communications*.

[10] Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, PODC '83, pages 27–30, New York, NY, USA, 1983. ACM.

[11] Brain M. Blum, Tian He, Sang Son, and John A. Stankovic. Igf: A state-free robust communication protocol for wireless sensor networks. 2003.

[12] Gabriel Bracha. An asynchronous [(n-1)/3]-resilient consensus protocols. Technical report, Ithaca, NY, USA, 1984.

[13] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. In *in Proc. 19th ACM Symposium on Principles of Distributed Computing (PODC*, pages 123–132, 2000.

[14] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20:398–461, November 2002.

[15] S. Cheekiralla and D.W. Engels. A functional taxonomy of wireless sensor network devices. In *Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on*, pages 949 –956 Vol. 2, 2005.

[16] Kyung Jun Choi and Jong-In Song. A miniaturized mote for wireless sensor networks. In *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, volume 1, pages 514 –516, 2008.

[17] B. Chor and C. Dwork. Randomization in byzantine agreement. In *Advances in Computing Research 5: Randomness and Computation*, pages 443–497. JAI Press, 1987.

[18] Naveen Sastry Chris Karlof and David Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, 2004.

[19] D. Davies and J.F. Wakerly. Synchronization and matching in redundant systems. *IEEE Transactions on Computers*, 27:531–539, 1978.

[20] Rodolfo de Paz Alberola and Dirk Pesch. Avroraz: extending avrora with an ieee 802.15.4 compliant radio chip model. In *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, PM2HW2N '08, pages 43–50, New York, NY, USA, 2008. ACM.

[21] Jing Deng, Richard Han, and Shivakant Mishra. Insens: Intrusion-tolerant routing in wireless sensor networks. 2002.

[22] D. Dolev and A. C. Yao. On the security of public key protocols. In *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*, pages 350–357, Washington, DC, USA, 1981. IEEE Computer Society.

[23] Michael Fischer. The consensus problem in unreliable distributed systems (a brief survey). In Marek Karpinski, editor, *Foundations of Computation Theory*, volume 158 of *Lecture Notes in Computer Science*, pages 127–140. Springer Berlin / Heidelberg, 1983.

[24] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. volume 32, pages 374–382, New York, NY, USA, April 1985. ACM.

[25] IEEE Standard for Information technology. Part 15.4: Wireless medium access control (mac) and physical layer(phy) specifications for low-rate wireless personal area networks (wpans). IEEE Computer Society, 2006.

[26] P. Gajbhiye and A. Mahajan. A survey of architecture and node deployment in wireless sensor network. In *Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the*, pages 426 –430, 2008.

[27] Lewis Girod, Jeremy Elson, Thanos Stathopoulos, Martin Lukac, and Deborah Estrin. Emstar: a software environment for developing and deploying wireless sensor networks. In *In Proceedings of the 2004 USENIX Technical Conference*, pages 283–296, 2004.

[28] A. Perrig H. Chan and D. Song. Random key predistribution schemes for sensor networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*.

[29] W. Li H. Deng and D. P. Agrawal. Routing security in wireless ad hoc networks. In *IEEE Communications Magazine*.

[30] The international telegraph and telephone consultative committee. Security architecture for open systems interconnection for ccitt applications. In *Data Communication networks: open systems interconnection(OSI), Recommendation X.800*.

[31] D. Song J. Newsome, E. Shi and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *International Symposium on Information Processing in Sensor Networks*.

[32] M. Johnson, M. Healy, P. van de Ven, M.J. Hayes, J. Nelson, T. Newe, and E. Lewis. A comparative review of wireless sensor network mote technologies. In *Sensors, 2009 IEEE*, pages 1439 –1442, 2009.

[33] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. In *IEEE International Workshop on Sensor Network Protocols and Applications*.

[34] R.M. Kling. Intel motes: advanced sensor network platforms and applications. In *Microwave Symposium Digest, 2005 IEEE MTT-S International*, page 4 pp., 2005.

[35] Kärkkäinen N. Tukeva P Korkalainen M., Sallinen M. Survey of wireless sensor networks simulation tools for demanding applications. In *Fifth International Conference on Networking and Services*, page 102 – 106. IEEE Computer Society, 2009.

[36] Panayiotis Kotzanikolaou, Rosa Mavropodi, and Christos Douligeris. Secure multipath routing for mobile ad hoc networks. In *Proceedings of the Second Annual Conference on Wireless On-demand Network Systems and Services*, pages 89–96, Washington, DC, USA, 2005. IEEE Computer Society.

[37] A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer. Shawn: A new approach to simulating wireless sensor networks, 2005.

[38] Vijayraman Kumar, Johnson Thomas, and Ajith Abraham. Secure directed diffusion routing protocol for sensor networks using the leap protocol.

[39] Leslie Lamport. Using time instead of timeout for fault-tolerant distributed systems. *ACM Trans. Program. Lang. Syst.*, 6:254–280, April 1984.

[40] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4:382–401, 1982.

[41] Jongdeog Lee, Krasimira Kapitanova, and Sang H. Son. The price of security in wireless sensor networks. *Comput. Netw.*, 54:2967–2978, December 2010.

[42] Javier Lopez and Jianying Zhou. book: Wireless sensor network security. Ios Press, 2008.

[43] K.S. Low, W.N.N. Win, and M.J. Er. Wireless sensor networks for industrial environments. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, volume 2, pages 271 –276, 2005.

[44] Mark Luk, Ghita Mezzour, Adrian Perrig, and Virgil Gligor. Minisec: a secure sensor network communication architecture. In *Proc. of the 6th Int* $\|Conf.onInformationProcessinginSensorNetworks, pages 479--488.ACMPress, 2007.$

[45] Mohamed Eltoweissy Michael chorzempa, Jung-min Park and Y. Thomas Hou. Key management for wireless sensor networks in hostile environments, in book chapter of security. In *in Security in Sensor Networks, Yang Xiao (Eds*. CRC Press, 2006.

[46] Henrique Moniz, Nuno Ferreira Neves, Miguel Correia, and Paulo Veríssimo. Experimental comparison of local and shared coin randomized consensus protocols. In *SRDS*, pages 235–244. IEEE Computer Society, 2006.

[47] Henrique Moniz, Nuno Ferreira Neves, Miguel Correia, and Paulo Veríssimo. Ritas: Services for randomized intrusion tolerance. *IEEE Trans. Dependable Sec. Comput.*, 8(1):122–136, 2011.

[48] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. 2002.

[49] Bryan Parno, Mark Luk, Evan Gaustad, and Adrian Perrig. Secure sensor network routing: A clean slate approach. In *IN PROCEEDINGS OF CONFERENCE ON FUTURE NETWORKING TECHNOLOGIES (CONEXT)*, 2006.

[50] Christina Pöpper, Mario Strasser, and Srdjan Čapkun. Anti-jamming broadcast communication using uncoordinated spread spectrum techniques. *IEEE J.Sel. A. Commun.*, 28:703–715, June 2010.

[51] Rodrigo Roman, Cristina Alcaraz, and Javier Lopez. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. *Mob. Netw. Appl.*, 12:231–244, August 2007.

[52] A Roy and N Sarm. Energy saving in mac layer of wireless sensor networks: a survey. In *National Workshop in Design and Analysis of Algorithm (NWDAA)*.

[53] K. Lai S. Marti, T. J. Giuli and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *ACM/IEEE International Conference on Mobile Computing and Networking*.

[54] V. Coskun S. Sancak, E. Cayirci and A. Levi. Sensor wars: detecting and defending against spam attacks in wireless sensor networks. In *IEEE International Conference on Communications*.

[55] Nicola Santoro and Peter Widmayer. Time is not a healer. In *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science*, pages 304–313, London, UK, 1989. Springer-Verlag.

[56] Bruce Schneier. In wiley, editor, *Applied cryptography (§4 section 4.10)*, 1996.

[57] Mario Strasser, Boris Danev, and Srdjan Čapkun. Detection of reactive jamming in sensor networks. *ACM Trans. Sen. Netw.*, 7:16:1–16:29, September 2010.

[58] M.A. Taleghan, A. Taherkordi, M. Sharifi, and Tai-Hoon Kim. A survey of system software for wireless sensor networks. In *Future Generation Communication and Networking (FGCN 2007)*, volume 2, pages 402 –407, 2007.

[59] Sam Toueg. Randomized byzantine agreements. In *Proceedings of the third annual ACM symposium on Principles of distributed computing*, PODC '84, pages 163–178, New York, NY, USA, 1984. ACM.

[60] Dimitris Vassis, George Kormentzas, Angelos N. Rouskas, and Ilias Maglogiannis. The ieee 802.11g standard fo high data rate wlans. *IEEE Network*, 19(3):21–26, 2005.

[61] M.A.M. Vieira, Jr. Coelho, C.N., Jr. da Silva, D.C., and J.M. da Mata. Survey on wireless sensor network devices. In *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference*, volume 1, pages 537 – 544 vol.1, 2003.

[62] Jun Fung Vojislav B. Misic and Jelena Misic. Mac layer security of 802.15.4-compliant networks. Department of Computer Science, University of Manitoba Winnipeg, Manitoba, Canada.

[63] John Paul Walters, Zhengqiang Liang, Weisong Shi, and Vipin Chaudhary. Wireless sensor network security: A survey," in book chapter of security. In *in Distributed, Grid, and Pervasive Computing, Yang Xiao (Eds*, pages 0–849. CRC Press, 2007.

[64] Anthony D. Wood, Lei Fang, John A. Stankovic, and Tian He. Sigf: A family of configurable, secure routing protocols for wireless sensor networks. In *In Proceedings of ACM SASN*, pages 35–48. ACM Press, 2006.

[65] Hejun Wu, Qiong Luo, Pei Zheng, and Lionel M. Ni. Vmnet: Realistic emulation of wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 18:277–288, 2007.

[66] A. Perrig Y. C. Hu and D. B. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *IEEE INFOCOM*.

[67] A. Perrig Y. C. Hu and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *ACM Workshop on Wireless Security*, page 3040. ACM, 2003.

[68] Minghui Shi Yixin Jiang, Chuang Lin and Xuemin Shen. Key management schemes for wireless sensor networks, in book chapter of security. In *in Security in Sensor Networks, Yang Xiao (Eds*. CRC Press, 2006.

[69] MyungJune Youn, Young-Yul Oh, Jaiyong Lee, and Yeonsoo Kim. IEEE 802.15.4 Based QoS Support Slotted CSMA/CA MAC for Wireless Sensor Networks. In *2007 International Conference on Sensor Technologies and Applications (SENSORCOMM 2007)*, pages 113–117. IEEE, October 2007.

[70] S. Zhu and W. Zhang. Group key management, in book chapter of security. In *in Security in Sensor Networks, Yang Xiao (Eds*. CRC Press, 2006.