Protocol Design and Implementation for Wireless Sensor Networks

PIERGIUSEPPE DI MARCO

KTH Electrical Engineering

Masters' Degree Project
Stockholm, Sweden April 2008

XR-EE-RT 2008:005

Abstract

Designing efficient and reliable communication protocols for wireless sensor networks in indoor monitoring applications is a challenging task, due to the uncertainty and dynamics of the environment.

We consider SERAN, a two-layer semi-random protocol that specifies a routing algorithm and a MAC layer for clustered wireless sensor networks. It combines a randomized and a deterministic approach: the former provides robustness over unreliable channels, the latter reduces the packet collisions. We provide a mathematical model for the protocol that allows us to analyze its behavior and optimize performance. We define an optimization problem, considering the energy consumption as objective function and constraints in terms of error rate and end-to-end delay.

A TinyOS implementation of the protocol on a WSN test bed composed by Moteiv's Tmote Sky wireless sensors is presented. Experimental results validate the model and show excellent performance for low data rate transmissions, with low average node duty cycle, which yields a long network lifetime.

Contents

Abstract I

1 Introduction **1**
1.1 Motivations 1
1.2 Problem Formulation 2
1.3 Contribution of the Thesis 3
1.4 Outline 3

2 Wireless Sensor Networks: an Overview **4**
2.1 Wireless Sensor Networks Applications 5
2.2 Research Challenges 7
2.3 Protocol Stack 8
2.4 Routing Techniques 9
2.4.1 Classification 12
2.5 MAC Protocols 13
2.5.1 Scheduled-based Protocols 13
2.5.2 Contention-based Protocols 14
2.5.3 MAC Protocols for WSN 16
2.6 Cross-Layer Protocol Design 19
2.6.1 LEACH Protocol 20
2.6.2 Breath Protocol 21
2.6.3 SERAN Protocol 21

3 Model and Optimization Problem **22**
3.1 Assumptions 23
3.2 Routing 23
3.3 Hybrid MAC 24
3.4 Mathematical Analysis 26
3.4.1 Absorption Time 27

3.4.2 Access Probability 28
3.4.3 Scheduling Policy 30
3.4.4 Sustainable Traffic 31
3.5 Energy Consumption 32
3.6 Latency Requirement 35
3.7 Error Rate Requirement 37
3.7.1 Problem Formulation 37
3.7.2 P(n,S,k) Evaluation 38
3.7.3 Packet Reception Rate 39
3.8 Optimization Problem 41

4 Protocol Implementation **43**
4.1 Hardware Technologies 43
4.1.1 Tmote Sky platform 43
4.2 Software Technologies 45
4.2.1 TinyOS 45
4.3 Time Synchronization and Network Initialization 49
4.3.1 Token Passing Procedure 49
4.4 MAC/Routing Implementation 51
4.4.1 Acknowledgement Mechanism 54
4.5 Network Lifetime 55
4.5.1 Characterization of the CC2420 Transceiver 55
4.5.2 Battery Model 56
4.6 Drawbacks 57

5 Experimental Results **59**
5.1 Network Setup 59
5.2 Validation 59
5.3 Performance Analysis 65
5.4 Network Lifetime Estimation 68

6 Conclusions **70**
6.1 Conclusions of the Work 70
6.2 Future Developments 71

A Protocol Parameters **72**

B Packet Structure **74**

References **76**

List of Figures

1.1 Automatic Production Line (courtesy of ABB web site - available: http://www.abb.com) 2

2.1 Architectural layers of a WSN 8
2.2 Routing Protocols in WSNs 12

3.1 Connectivity graph 23
3.2 Hybrid MAC representation 24
3.3 Markov Chain 27
3.4 Expected forwarding time in number of CSMA-slot for $p = 1/k$ 30
3.5 Example of Scheduling Table 31
3.6 Markov Chain for $n = 1, S = 4, k = 3$ 38
3.7 PRR vs.$(S - k)$, for different values of k 40
3.8 S vs.k, for fixed values of PRR 40
3.9 PRR vs.$(S - k)$, for $k = 3$ - upper and lower bounds 41

4.1 Tmote Sky platform 44
4.2 Flow diagram of SERAN Protocol: transmission side (left) and receiving side (right) 51
4.3 Architectural layers of CC2420 Radio stack 53
4.4 State diagram and typical current consumption and transition times for CC2420 transceiver 56

5.1 Test-bed 60
5.2 Network Topology 60
5.3 Packet Reception Rate vs. TDMA-slot duration for $k = 3$ . . . 61
5.4 Average Delay, $\lambda = 1pkt/10s$ 63
5.5 Average Duty Cycle, $\lambda = 1pkt/10s$ 63
5.6 Delay distribution for clusters 1, 2 and 4 (3,2 and 1 hops to the Controller) 64

5.7 Average Delay, $\lambda = 3pkt/10s$ (left) $\lambda = 1pkt/5s$ (right) 67
5.8 Average Duty Cycle, $\lambda = 3pkt/10s$ (left) $\lambda = 1pkt/5s$ (right) . 67
5.9 Time distribution 69

List of Tables

4.1 Node power consumption 56
4.2 Node energy specification 57

5.1 Validation: average values of PRR, delay and duty cycle . . . 62
5.2 Performance analysis: average values of PRR, delay and duty cycle 65
5.3 Time distribution and energy consumption in a TDMA-cycle . 68

Chapter 1

Introduction

The rapid evolution of wireless technologies and the significant growth of wireless network services have made wireless communications an ubiquitous means for transporting information across many different domains. Within the framework of Wireless Sensor Networks (WSNs), there are many potential possibilities where a WSN can be deployed to support numerous applications. However, the current applications in real-life are very limited. The main reason for the delay in the adoption is the lack of a system level approach. This is a design methodology that, given a set of application constraints, is able to synthesize a design solution that guarantees the required latency and quality of service subject to unreliable channel conditions.

1.1 Motivations

Our approach is mainly motivated by industrial control applications. In particular, we are interested in designing WSNs in manufacturing cells, as in automatic production lines (Fig. 1.1). A WSN is deployed to measure sensitive parameters in specific regions and to send it to a controller.

Although there are several papers ([1], [2] and [3]) that model the networking performance of WSNs, the practical evaluations of networking in real test-bed environments are limited (paper [6]). The variability of the wireless environment and the simplified hypotheses, often assumed in these models, attribute great importance to the implementation stage. Consequently, we focused our efforts on the practical implementation, trying to mediate between the need of abstraction in the theoretical model and physi-

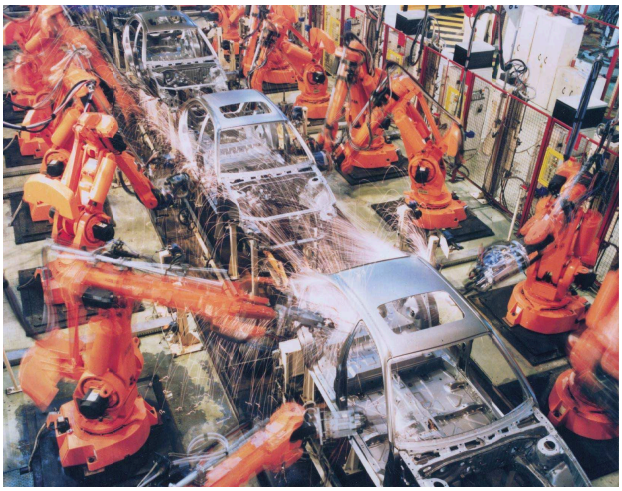

Figure 1.1: Automatic Production Line (courtesy of ABB web site - available: http://www.abb.com)

cal constraints on the platforms. We chose SERAN, a semi-random protocol for clustered WSNs, originally designed for manufacturing applications [1].

1.2 Problem Formulation

This study will evaluate the performance of the SERAN protocol in WSNs in real environments.

The main aim of protocols like SERAN is the maximization of the network lifetime subject to application requirements.
In paper [1], Bonivento et al. proposed a mathematical model and the related optimization problem for SERAN. The objective function is the energy consumption and the constraint is the delay. The problem is expressed as:

$$\begin{aligned} \text{minimize} \quad & E_{\text{tot}} \\ \text{subject to} \quad & D \leq D_{\max} \end{aligned} \tag{1.1}$$

According to the approach proposed in papers [3] and [6], we decided to improve it, inserting a requirement on the error rate. We analyzed the performance considering the probability that a packet is received at destination

(Packet Reception Rate) greater than a fixed threshold.

$$\begin{aligned} \text{minimize} \quad & E_{\text{tot}} \\ \text{subject to} \quad & PRR \geq PRR_{\min} \\ & D \leq D_{\max} \end{aligned} \tag{1.2}$$

1.3 Contribution of the Thesis

The main contributions of this thesis are two:

1. definition of a mathematical analysis of Packet Reception Rate (PRR) in SERAN, to enhance the optimization problem.

2. implementation of the protocol on real motes and performance evaluation in a test-bed environment.

1.4 Outline

In Chapter 2 we describe general features of WSNs and introduce design aspects of MAC and routing layer, with considerations about the cross-layer design. In Chapter 3 the mathematical model is presented, referring to the SERAN protocol. The implementation aspects are described in Chapter 4, while in Chapter 5 the experimental results are presented and discussed. Conclusions and future works are resumed in Chapter 6.

Chapter 2

Wireless Sensor Networks: an Overview

Wireless Sensor Networks (WSNs) are ad-hoc networks, consisting of spatially distributed devices (*motes*) using sensor nodes to cooperatively monitor physical or environmental conditions at different locations.

Devices in a WSN are resource constrained; they have low processing speed, storage capacity, and communication bandwidth. In most settings, the network must operate for long periods of time, but the nodes are battery powered, so the available energy resources limit their overall operation. To minimize energy consumption, most of the device components, including the radio, should be switched off most of the time [7]. Another important characteristic is that sensor nodes have significant processing capability in the ensemble, but not individually. Nodes have to organize themselves, administering and managing the network all together, and it is much harder than controlling individual devices. Furthermore, changes in the physical environment where a network is deployed make also nodes experience wide variations in connectivity and it influences the networking protocols.

The main factors that complicate the protocol design for WSNs can be summarized in:

- Fault tolerance: the necessity to sustain sensor networks functionalities without any interruption, after a node failure.
- Scalability: the possibility to enlarge and reduce the network.
- Deployment: given a certain environment it should be possible to find

the suitable deploying location for each sensor.

- Power management: the network lifetime needs to be maximized.

In spite of a greater effort required for building a WSN, the interest in this technology is increasing. Recently a noteworthy research area covered WSNs and applications in industrial and commercial field but a lot of work has to be done to discover and exploit all their potentialities.

2.1 Wireless Sensor Networks Applications

The uses of WSN are generally classified into [7]:

- monitoring space
- monitoring targets

The former category includes for instance habitat monitoring, precision agriculture, electronic surveillance, intelligent alarms and generally what is called "domotics" [1]. The latter category embraces structural monitoring[2], medical diagnostics, industrial equipment maintenance and urban terrain mapping. Another category is represented by hybrid WSN, where the aim is to control the interaction between targets with each other and the surrounding environment. Emergency management, for example, involves risk analysis, prevention, supporting activities and recovering after disasters; it has civil implications but it is also important in terms of industrial emergency response (nuclear plants).

Security applications WSNs may be used for infrastructure security and counterterrorism applications. Critical buildings and facilities such as power plants and communication centers should be preserved from potential terrorists. Integrated networks of video, acoustic, and other sensors can be deployed around these facilities. These sensors can guarantee early detection of possible trouble. Improved coverage and detection and a reduced false alarm rate can be achieved by fusing the data from multiple sensors. Even

[1] It is a field within building automation, oriented to the application of automation techniques for the comfort and security of homes and their residents (internal climate or lighting control, fire and gas detection...)

[2] Used in earthquake engineering science

though fixed sensors connected by a fixed communication network protect most facilities, wireless ad hoc networks can provide more flexibility and additional coverage when needed. WSNs can also be used to detect biological, chemical, and nuclear attacks.

Industrial control Industry has shown interest in sensing as a means of lowering cost and improving machine and user performance and maintainability. Nowadays it is possible to monitor the machine state through determination of vibration or lubrication levels. Sensors can be inserted into regions inaccessible by humans. Remote wireless sensors can allow a factory to be equipped, after the fact to guarantee and maintain compliance with safety laws and guidelines while keeping installation costs low. In an industrial environment spectral sensors[3] are often used. Optical sensors[4] can replace existing instruments and perform material property and composition measurements. Optical sensing is also facilitated by miniaturization. The goal of this and other industrial applications of WSNs is to enable multipoint or matrix sensing: inputs from hundreds or thousands of sensors feed into databases that can be queried in any number of ways to show real-time information on a large or small scale.

Environmental monitoring Environmental sensors can be used to study vegetation response to climatic trends and diseases, and acoustic and imaging sensors can identify, track and measure the population of animals, for example birds or endangered species.

Traffic control WSNs are nowadays used for vehicle traffic monitoring and control. Most traffic intersections have either overhead or buried sensors to detect vehicles and control traffic lights. Video cameras are frequently used to monitor road segments with heavy traffic, with the video sent to human operators at central locations. However, these sensors and the communication network that connect them are costly, so traffic monitoring is usually limited to a few critical points. Inexpensive wireless ad hoc networks will completely change the scenario in the traffic monitoring and control. Cheap sensors with embedded networking capability can be deployed at every road intersection to detect and count vehicle traffic and estimate its speed. The sensors will

[3]They collect and transmit data from different parts of the electromagnetic spectrum
[4]They works in the optical wavelength range

communicate with neighboring nodes to eventually develop a global traffic picture, which can be handled by human operators or automatic controllers to generate control operations. A different and more radical revolution is the sensors attached to each vehicle. As the vehicles pass each other, they exchange summary information on the location and the speed and density of traffic, information that may be generated by ground sensors. These summaries propagate from vehicle to vehicle and can be used by drivers to avoid traffic congestion and organize alternative routes.

2.2 Research Challenges

Hardware and software constraints originate a lot of design issues that must be addressed to achieve an effective and efficient operation of WSNs. Besides, new application scenarios lead to new challenges. The following are just examples of some open questions:

- *Energy-aware algorithms*: sensor nodes are powered by external batteries and it can be difficult to replace them when consumed (often sensor nodes are deployed in remote and hostile environments), so it is critical to design algorithms and protocols that utilize minimal energy. To do that, implementers must reduce communication between sensor nodes, simplify computations and apply lightweight security solutions.

- *Location discovery*: many applications that can track an object require knowing the exact or approximate physical location of a sensor node, in order to link sensed data with the object under analysis. So many geographical routing protocols need the location of sensor nodes to forward data among the networks. Location discovery protocols must be designed in such a way that minimum information is needed to be exchanged among nodes to discover their location. Solutions like GPS are not recommended because of the energy consumption and the price of the components.

- *Cost*: this is another factor that influences design. Manufacturers try to keep the cost at minimum levels since most sensor nodes are usually needed for many applications. New technologies are always costly. If the cost is high, the adoption and spread of sensor technology will be prohibitive.

- *Security*: it is not possible to introduce a new technology without considering security aspects. However, as it happens with other technologies, security is not the top priority when designing something new. Security solutions are constrained when applying them to sensor networks. For example, cryptography requires complex processing to provide encryption to the transmitted data. Some of the many issues that need to be addressed in a security context are: secure routing, secure discovery and verification of location, key establishment and trust setup, attacks against sensor nodes, secure group management and secure data aggregation.

2.3 Protocol Stack

A simplified protocol stack for a WSN is summarized in Fig. 2.1.

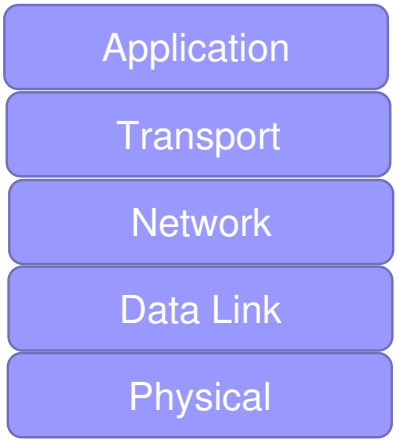

Figure 2.1: Architectural layers of a WSN

We can consider four main levels [13]:

- **Application layer**: It defines a standard set of services and interface primitives available to a programmer independently on their implementation on every kind of platform. An example is the so called *sensor network services platform* (SNSP) [14].

- **Transport layer**: It helps to maintain the flow of data if the sensor networks application requires it. This layer is especially needed when

the system is planned to be accessed through Internet or other external networks. Unlike protocols such as TCP, the end-to-end communication schemes in sensor networks are not based on global addressing. Therefore, new schemes that split the end-to-end communication probably at the sinks may be needed.

- **Network layer**: It takes care of routing the data, directing the process of selecting paths along which to send data in the network.
- **Data Link layer**: It provides the multiplexing of data streams, data frame detection and medium access control (MAC).
- **Physical layer**: it is responsible for frequency and power selection, modulation, and data encryption.

2.4 Routing Techniques

Routing in WSNs is a hard challenge due to the inherent characteristics that distinguish these networks from other wireless networks like mobile ad hoc networks or cellular networks [9]. Some important aspects are listed below.

Node deployment. It is application-dependent and can be either manual (deterministic) or randomized. Position awareness of sensor nodes is also important, since data collection is normally based on the location;

Energy consumption without losing accuracy. Sensor nodes are tightly constrained in terms of energy, processing, and storage capacities, so they require careful resource management. The lifetime of nodes is a critical issue because of the limited battery lifetime. In multi-hop networks, the malfunctioning of some sensor nodes due to power failure can cause significant topological changes, and might require rerouting of packets and reorganization of the network.

Data reporting method. It can be categorized as:

- time-driven, when data are transmitted at constant periodic time intervals;

- event-driven, when sensor nodes react immediately to the occurrence of a certain event;
- query-driven, when sensor nodes respond to a query generated by the BS or another node in the network.

It can be also a hybrid of all previous methods. The routing protocol is highly influenced by the data reporting method in terms of energy consumption and route calculations.

Node/link heterogeneity. In many studies, all sensor nodes were assumed to be homogeneous (e.g. have equal capacities in terms of computation, communication, and power), but, depending on the application, a sensor node can have a different role or capability. For example, some applications might require a diverse mixture of sensors for monitoring temperature, pressure, and humidity of the surrounding environment, detecting motion via acoustic signatures, and capturing images or video tracking of moving objects. Even data reading and reporting can be generated from these sensors at different rates, subject to diverse QoS constraints, and can follow multiple data reporting models.

Fault tolerance. Some sensor nodes may fail or be blocked due to lack of power, physical damage, or environmental interference. The failure of sensor nodes should not affect the overall task of the sensor network.

Scalability. Routing scheme must be able to work with a huge number of sensor nodes. In addition, sensor network routing protocols should be scalable enough to respond to events in the environment.

Network dynamics. In many applications both the base station or sensor nodes can be mobile. The routing protocol should consider this eventuality, making the design more complicated.

Addressing scheme. The relative large number of sensor nodes and the constraint in terms of overhead does not allow building a global addressing scheme as IP-based protocols.

Transmission media. The traditional problems associated with a wireless channel (e.g. fading, high error rate) may affect the operation of the sensor network. In general, the required bandwidth of sensor data will be low, on the order of 1–100 kb/s. Related to the transmission media is the design of MAC. One approach to MAC design for sensor networks is to use time-division multiple access (TDMA)-based protocols that conserve more energy than contention-based protocols like carrier sense multiple access (CSMA) (e.g. IEEE 802.11). Bluetooth technology can also be used.

Connectivity. High node density in sensor networks precludes them from being completely isolated from each other and sensor nodes are expected to be highly connected. However, it may not prevent the network topology from being variable and the network size from reducing due to sensor node failures. In addition, connectivity depends on the possibly random distribution of nodes.

Coverage. A given sensor's view of the environment is limited in both range and accuracy; it can only cover a limited physical area of the environment.

Data aggregation. Data sensed by many sensors in WSNs is typically based on common phenomena, so there is a high probability that this data has some redundancy, which needs to be exploited by the routing protocols to improve energy and bandwidth utilization. Data aggregation is the combination of data from different sources according to a certain aggregation function (e.g. duplicate suppression, minima, maxima and average). This technique has been used to achieve energy efficiency and data transfer optimization in a number of routing protocols.

Quality of service. In many applications, conservation of energy is considered relatively more important than the quality of data sent. Hence, as energy is depleted, the network may be required to reduce the quality of results in order to reduce energy dissipation in the nodes (energy-aware routing protocol).

Consequently, routing, power management and data dissemination protocols for WSNs must be specifically designed.

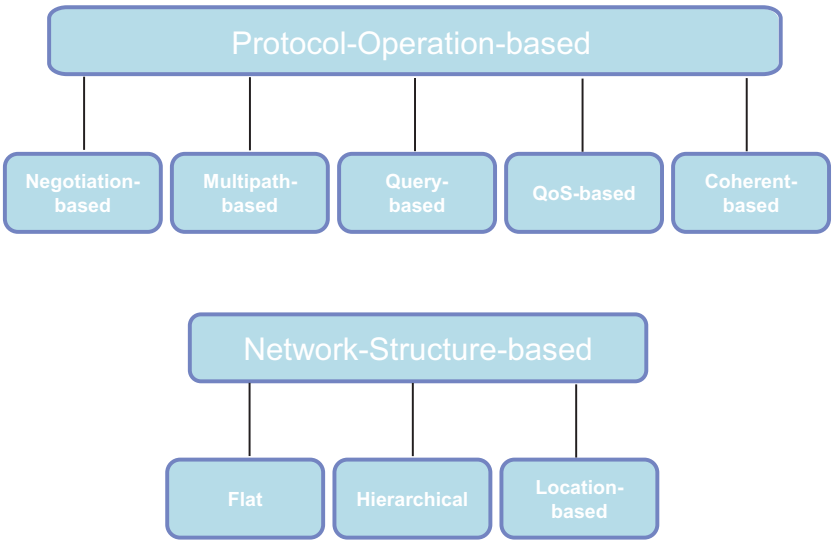

Figure 2.2: Routing Protocols in WSNs

2.4.1 Classification

Routing protocols in WSNs might differ depending on the application (*Protocol-Operation-based*) and network architecture (*Network-Structure-based*) as shown in Fig. 2.2. Based on the underlying network there are three protocol categories:

- **Flat Routing**: each node plays the same role and sensor nodes collaborate to perform the sensing task.
- **Hierarchical (Cluster-based) Routing**: higher-energy nodes are used to process and send the information, while low-energy nodes are used to perform the sensing in the proximity of the target. The creation of clusters and assigning special tasks to cluster heads can greatly contribute to overall system scalability, lifetime, and energy efficiency. Hierarchical routing is an efficient way to lower energy consumption within a cluster, performing data aggregation and fusion in order to decrease the number of transmitted messages to the sink node;
- **Location-based**: sensor nodes are addressed by means of their locations. The distance between neighboring nodes can be estimated on the basis of incoming signal strengths. Relative coordinates of neighboring nodes can be obtained by exchanging such information between neigh-

bors or by communicating with a satellite using GPS. To save energy, some location-based schemes demand that nodes should go to sleep if there is no activity.

Depending on the protocol operation we can divide routing protocols in:

- **Multipath-based**: use multiple paths rather than a single path in order to enhance network performance. For instance the fault tolerance can be increased by maintaining multiple paths between the source and destination at the expense of increased energy consumption and traffic generation.
- **Query-based**: the destination nodes propagate a query for data from a node through the network, a node with this data sends the data that matches the query back to the node that initiated it.
- **Negotiation-based**: use negotiation in order to eliminate redundant data transmissions. Communication decisions are also made based on the resources available.
- **QoS-based**: when delivering data, the network balances between energy consumption and data quality through certain QoS metrics as delay, energy or bandwidth.
- **Coherent-based**: the entity of local data processing on the nodes distinguish between coherent (minimum processing) and non-coherent (full processing) routing protocols.

2.5 MAC Protocols

MAC protocols can be roughly divided into two groups [15]: scheduled-based and contention-based protocol.

2.5.1 Scheduled-based Protocols

Scheduled protocols are very attractive for applications in sensor networks because of their energy efficiency. Since slots are pre-allocated to individual nodes, they are collision-free. These protocols are characterized by a duty cycle built-in with the inherent collision-free nature that ensure low energy

consumption. On the other side, the complexity of the design is high due to problems of synchronization. In general, they are not flexible to changes in node density or movement, and lack of peer-to-peer communication.

The representative schedule-based protocols are:

- **Time Division Multiple Access (TDMA)**: it allows several users to share the same frequency channel by dividing the signal into different time-slots. It has a natural advantage of collision free medium access. It supports low duty cycle operation: a node only needs to turn on its radio during the slot that it is assigned to transmit or receive. However, it includes clock drift problems and decreased throughput at low traffic loads due to idle slots. The limits with TDMA systems are synchronization of the nodes and adaptation to topology changes (i.e. insertion of new nodes, exhaustion of battery capacities, and corrupted links due to interference). The slot assignments, therefore, should be done with regard to such possibilities. However, it is not easy to change the slot assignment within a decentralized environment for traditional TDMA, since all nodes must agree on the slot assignments.

- **Frequency Division Multiple Access (FDMA)**: it allocates users with different carrier frequencies of the radio spectrum. It is another scheme that offers a collision-free medium, but it requires additional hardware to dynamically communicate with different radio channels. This increases the cost of the sensor nodes, which is in contrast with the philosophy of sensor network systems.

- **Code Division Multiple Access (CDMA)**: it employs spread spectrum technology and a special coding scheme (where each transmitter is assigned a code) to allow multiple users to be multiplexed over the same physical channel. It also offers a collision-free medium, but its high computational requirement is a major obstacle for the minimum energy consumption objective in WSNs.

2.5.2 Contention-based Protocols

Contention schemes differ in principle from scheduled schemes since a transmitting user is not guaranteed to be successful. Unlike scheduled protocols, contention protocols do not divide the channel into sub-channels or

pre-allocate the channel for each node to use. Instead, a common channel is shared by all nodes and it is allocated on demand. At any moment, a contention mechanism is employed to decide which node has the right to access the channel. Contention protocols have several advantages compared to scheduled protocols. First, because contention protocols allocate resources on demand, they can scale more easily across changes in node density or traffic load. Second, contention protocols can be more flexible as topologies change. There is no requirement to form communication clusters, and peer-to-peer communication is directly supported. Finally, contention protocols do not require fine-grained time synchronization as in TDMA protocols. The major disadvantage of a contention protocol is its inefficient usage of energy. The resolution process does consume resources. If the probability of interference is small, such as might be the case with bursty users, taking the chance of having to resolve the interference compensates for the resources that have to be expanded to ensure freedom of conflicts. Moreover, in most conflict-free protocols, idle users do consume a portion of the channel resources; this portion becomes major when the number of potential users in the system is very large to the extent that conflict-free schemes are impractical. In contention schemes idle users do not transmit and thus do not consume any portion of the channel resources.

The representative contention-based protocols are:

- **ALOHA**: a node simply transmits a packet when it is generated (pure ALOHA) or at the next available slot (slotted ALOHA). Should the transmission be unsuccessful, every colliding user, independently of the others, schedules its retransmission to a random time in the future. This randomness is required to ensure that the same set of packets does not continue to collide indefinitely.

- **Carrier Sense Multiple Access (CSMA)**: when a user generates a new packet the channel is sensed and if found idle the packet is transmitted. When a collision takes place every transmitting user reschedules a retransmission of the collided packet to some other time in the future (chosen randomly) when the same operation will be repeated. In accordance with common networking lore, CSMA methods have a lower delay and promising throughput potential at lower traffic loads, which generally happens to be the case in WSNs. However, additional collision avoidance or collision detection methods should be employed.

2.5.3 MAC Protocols for WSN

Medium Access Control protocols designed for wireless LANs have been optimized for maximum throughput and minimum delay, while the low energy consumption has been left as a secondary requirement. In WSNs, energy efficiency is the main task. There are large opportunities of energy savings at the MAC layer. In parer [15] four sources of energy waste have been identified: collisions, control packet overhead, listening to a transmission destined to someone else (overhearing) and idle listening. Most important source of energy savings in a sensor network is to avoid idle listening. One way to avoid idle listening is to use the TDMA protocol, but various protocol solutions have been proposed in this direction.

IEEE 802.11 MAC

IEEE 802.11 is the first wireless LAN (WLAN) standard proposed in 1997 [21]. The medium access mechanism, called the *Distributed Coordination Function*, is basically a Carrier Sense Multiple Access with Collision Avoidance mechanism (CSMA/CA). A station wanting to transmit senses the medium. If the medium is busy then it defers. If the medium is free for a specified time (called Distributed Inter Frame Space, DIFS in the standard), then the station is allowed to transmit. The receiving station checks the CRC of the received packet and sends an acknowledgment packet. If the sender does not receive the ACK, then it retransmits the frame until it receives ACK or is thrown away after a given number of retransmissions. According to the standard, a maximum of seven retransmissions are allowed before the frame drops.
In order to reduce the probability of two stations colliding due to not hearing each other, which is well-known as the "*hidden node problem*", the standard defines a Virtual Carrier Sense mechanism: a station wanting to transmit a packet first transmits a short control packet called RTS (Request To Send), which includes the source, destination, and the duration of the intended packet and ACK transaction. The destination station responds (if the medium is free) with a response control packet called CTS (Clear to Send), which includes the same duration information.
Obviously, collisions are still possible because the efficiency of CSMA/CA depends on the sensing range of each node and the presence of a hidden station. In general, the performances of CSMA/CA are strictly related to the

network topology and the nodes density: the more nodes can hear each other the better quality of communication can be achieved avoiding collisions. Inevitably, large latency times affect the efficiency of the system, because before transmitting each station has to wait an unpredictable amount of time that mainly depends on the demands of users and topology of the network.

Sensor MAC (S-MAC)

The basic concept behind the Sensor-MAC (S-MAC) protocol is the locally managed synchronization and the periodic sleep–listen schedules [17]. Basically built in a contention-based fashion, S-MAC strives to retain the flexibility of contention-based protocols while improving energy efficiency in multihop networks. S-MAC includes approaches to reduce energy consumption from all the major sources of energy waste: idle listening, collision, overhearing and control overhead. Neighboring nodes form virtual clusters so as to set up a common sleep schedule. If two neighboring nodes reside in two different virtual clusters, they wake up at the listen periods of both clusters. Schedule exchanges are accomplished by periodic SYNC packet broadcasts to immediate neighbors. The period for each node to send a packet is called the synchronization period. Collision avoidance is achieved by a carrier sense. Furthermore, RTS/CTS packet exchanges are used for unicast-type data packets. Periodic sleep may result in high latency, especially for multi-hop routing algorithms, since all intermediate nodes have their own sleep schedules. The latency caused by periodic sleeping is called sleep delay. The adaptive listening technique is proposed to improve the sleep delay and thus the overall latency. In that technique, the node that overhears its neighbor's transmissions wakes up for a short time at the end of the transmission. Hence, if the node is the next-hop node, its neighbor could pass data immediately. The end of the transmissions is known by the duration field of the RTS/CTS packets. The energy waste caused by idle listening is reduced by sleep schedules in S-MAC. In addition to its implementation simplicity, time synchronization overhead may be prevented by sleep schedule announcements. However broadcast data packets do not use RTS/CTS, which increases collision probability. Adaptive listening incurs overhearing or idle listening if the packet is not destined to the listening node. Sleep and listen periods are predefined and constant, which decreases the efficiency of the algorithm under variable traffic load.

Timeout MAC (T-MAC)

Timeout-MAC (T-MAC) is proposed to enhance the poor results of the S-MAC protocol under variable traffic loads. As indicated above, the static sleep–listen periods of S-MAC result in high latency and lower throughput. In T-MAC, the listen period ends when no activation event has occurred for a time threshold. The main drawback of this protocol is an early sleeping problem, as defined in paper [18].

Berkeley MAC (B-MAC)

B-MAC is highly configurable and can be implemented with a small code and memory size. B-MAC consists of: clear channel assessment (CCA), packet back-off and link layer acknowledgements. For CCA, B-MAC uses a weighted moving average of samples when the channel is idle in order to assess the background noise and to better be able to detect valid packets and collisions. The packet back-off time is configurable and is chosen from a linear range as opposed to an exponential back-off scheme typically used in other distributed systems. This reduces delay and works because of the typical communication patterns found in a WSN. B-MAC also supports a packet by packet link layer acknowledgement. In this way only important packets need to pay the extra cost. A low power listening scheme is employed where a node cycles between awake and sleep cycles. While awake, it listens for a long enough preamble to assess if it needs to stay awake or can return to sleep mode. This scheme saves significant amounts of energy. Many MAC protocols use a request to send (RTS) and clear to send (CTS) style of interaction. This works well for ad hoc mesh networks where packet sizes are large (1000s of bytes). However, the overhead of RTS-CTS packets to set up a packet transmission is not acceptable in WSNs where packet sizes are on the order of 50 bytes. B-MAC, therefore, does not use a RTS-CTS scheme.

Zebra MAC (Z-MAC)

Z-MAC is a hybrid MAC scheme for sensor networks that combines the strengths of TDMA and CSMA while offsetting their weaknesses [20]. The main feature of Z-MAC is its adaptability to the level of contention in the network so that under low contention, it behaves like CSMA, and under high contention, like TDMA. By mixing CSMA and TDMA, Z-MAC becomes

more robust to timing failures, time-varying channel conditions, slot assignment failures and topology changes than a stand-alone TDMA. In Z-MAC, a time slot assignment is performed at the time of deployment and higher overhead is incurred at the beginning.
Each node is *owner* of one or more slots, but, unlike TDMA, a node may transmit during any time slot in Z-MAC. Before a node transmits during a slot (not necessarily at the beginning of the slot), it always performs carrier-sensing and transmits a packet when the channel is clear. However, the owner of that slot always has higher priority over its non-owners in accessing the channel. The priority is implemented by adjusting the initial contention window size in such a way that the owners are always given earlier chances to transmit than non-owners.

There are various MAC protocols for WSNs besides the presented solutions. Optimal choice of MAC protocols is determined by application specified goals such as accuracy, latency, and energy efficiency.
However, B-MAC protocol is widely used because it has good results even with default parameters and it performs better than the other protocols.

2.6 Cross-Layer Protocol Design

Most of the communication protocols for WSNs follow the traditional layered protocol architecture. While these protocols may achieve very high performance in terms of the metrics related to each of these individual layers, they are not jointly optimized to maximize the overall network performance while minimizing the energy consumption [22]. Considering the energy constraint and processing resources of WSNs, joint optimization and design of networking layers, (i.e. cross-layer design), stands as the most promising alternative to inefficient traditional layered protocol architectures. The central idea of cross-layer design is to optimize the control and exchange of information over two or more layers to achieve significant performance improvements by exploiting the interactions between various protocol layers.
An important question in the area of cross-layer design is what parameters need to be shared among different layers of the protocol stack and how can each layer be made robust to the changing network conditions. The benefits and advantages from relaxing the rigid layered structure needs to be quantified, and the associated complexity and stability issues with implementing

such cross-layer design need to be studied more thoroughly.

In literature, the cross-layer design focuses on the interaction or modularity among physical, MAC and routing layers. Some examples of cross-layer approaches are illustrated, to introduce the SERAN protocol.

2.6.1 LEACH Protocol

Low Energy Adaptive Clustering Hierarchy (LEACH) is a cluster-based protocol, which includes distributed cluster formation and a hierarchical clustering algorithm [10]. LEACH randomly selects a few sensor nodes as cluster heads (CHs) and rotates this role to evenly distribute the energy load among the sensors in the network. In LEACH, the CH nodes compress data arriving from nodes that belong to the respective cluster, and send an aggregated packet to the BS in order to reduce the amount of information that must be transmitted to the BS. LEACH uses a TDMA/CDMA MAC protocol to reduce inter-cluster and intra-cluster collisions. However, data collection is centralized and performed periodically. LEACH is able to increase the network lifetime, but has some problem linked to the assumptions used:

- It should be possible for all nodes to transmit with enough power to reach the BS if needed. Each node should have computational power to support different MAC protocols, so it is not applicable to networks deployed in large regions.

- It also assumes that nodes always have data to send, and nodes located close to each other have correlated data. It is not obvious how the number of predetermined CHs (p) is going to be uniformly distributed through the network, so there is the possibility that the elected CHs will be concentrated in one part of the network; hence, some nodes will not have any CHs in their vicinity.

- The idea of dynamic clustering brings extra overhead (head changes, advertisements, etc.) may diminish the gain in energy consumption.

- The protocol assumes that all nodes begin with the same amount of energy capacity in each election round, assuming that being a CH consumes approximately the same amount of energy for each node.

2.6.2 Breath Protocol

In paper [6] a cross-layer protocol based on a randomized routing, MAC and duty cycling is presented. According to Breath, a node sends a data packet to another one randomly selected in a forwarding region, which is located in the direction toward the sink node of the network. This procedure is driven by beacon messages exchange from nodes in the forwarding region available to receive data packets. The MAC is randomized and does not implement any acknowledgement or retransmission scheme. Each node, either transmitter or receiver, does not stay in an active state, but goes to sleep for a random amount of time, which depends on the traffic conditions making the duty cycling algorithm also randomized. Breath is optimized to minimize the energy consumption of the network while ensuring a desired reliability and end-to-end delay in the packet delivery. The main drawback of the protocol is the bad operative condition in terms of high wake up rate.

2.6.3 SERAN Protocol

Originally proposed in paper [1], SERAN is a clustered two-layer protocol based on a semi-random approach. It combines randomized and deterministic components to jointly define routing and MAC layer.
Unlike LEACH, SERAN does not have cluster heads and the related problems. It uses a Hybrid TDMA/CSMA MAC protocol. The TDMA scheme is implemented at cluster level and reduces the wake up rate, while the CSMA provides robustness over unreliable channels and an acknowledgement-based contention scheme allows reducing duplicated packets. A similar double nature is in the routing algorithm. The combined result is a high reliability and good energy saving.
For these reasons, SERAN seems to be one of the best candidate protocols for WSNs and it is taken as reference in this work.

Chapter 3

Model and Optimization Problem

In this chapter we will formulate a mathematical model of SERAN, introducing the constrained optimization problem and the adopted solution.

As shown in Section 2.2, saving energy is one of the most important research challenges in WSNs. Sensor nodes are powered by external batteries and often it is hard to replace them after consuming, while most of the applications require long lifetime in the order of years. Hence, the choice of an objective function in terms of energy consumption is clearly justified. On the other hand, a mere optimization for energy can lead the network to work without fulfilling its tasks. Energy efficiency has to be well-balanced with the assigned requirements and network purposes.

Basically, the application requires two constraints:

1. *Error rate guarantee*: in terms of Packet Reception Rate (PRR) defined as the probability that a packet is received at destination.

2. *End-to-End delay guarantee*: in terms of maximum delay between the furthest node and the destination node.

The problem is rewritten as:

$$\begin{aligned} \text{minimize} \quad & E_{\text{tot}} \\ \text{subject to} \quad & PRR \geq PRR_{\min} \\ & D \leq D_{\max} \end{aligned} \tag{3.1}$$

where the objective function E_{tot} is the total energy consumption of the network, PRR_{min} is the minimum threshold for the PRR and D_{max} is the maximum admitted end-to-end delay.

3.1 Assumptions

Without loss of generality, SERAN is presented referring to the clustered topology of Fig. 3.1, as in paper [2]. Each star is a cluster of node and the connectivity between two clusters is represented by the double arrow. The Controller, denoted with C in the graph, can be represented by a sink node, linked to an actual application controller.

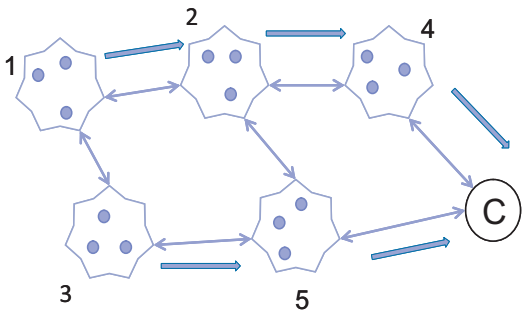

Figure 3.1: Connectivity graph

There are some important assumptions to consider:

- the Controller knows *a priori* the number of total nodes, the position of the clusters and how many nodes are in each cluster;
- each node knows to which cluster it belongs.

This means that the Controller has a good estimation of the amount of data generated by each cluster and the cluster structure is global information shared in the nodes. From a protocol definition perspective, these are very useful to simplify the analysis. Moreover, these hypotheses are acceptable in an industrial monitoring application.

3.2 Routing

The routing solution of SERAN is based on a semi-random scheme.
The routing layer in the protocol stack can be hierarchically subdivided in

two parts:

- A **static** route scheduling performed at cluster level;
- A **dynamical** routing algorithm at node level.

In this way a transmitter has knowledge of the region to which the packet will be forwarded, but the actual choice of forwarding node is made at random. This random choice is not performed at the network layer, but it is a result of an acknowledgment contention scheme performed at the MAC layer by all the candidate receivers. The overhead of purely random approaches is so reduced.

The first step of the SERAN routing algorithm consists of calculating the shortest path from every cluster to the Controller and generating the minimum spanning tree. In the presented topology (Fig. 3.1) this is represented by the bold single arrows. Then, packets are forwarded to a randomly chosen node within the next-hop cluster in their fixed path to the Controller.
We can observe that these operations are done without need of a cluster head node within clusters; nodes need to be aware only of the next-hop cluster connectivity and do not need a neighbor list of next hop nodes.

3.3 Hybrid MAC

A two-level semi-random scheme is implemented at MAC layer (see Fig. 3.2):

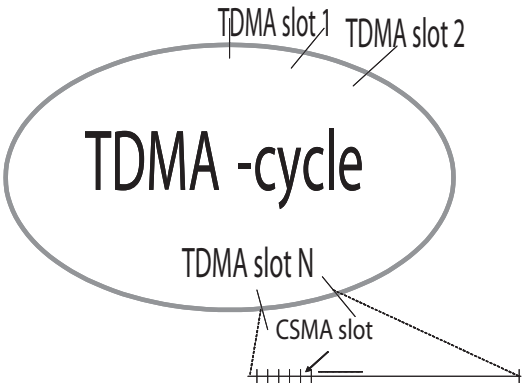

Figure 3.2: Hybrid MAC representation

- *A deterministic MAC with a weighted TDMA Scheme*: it regulates channel access among clusters. The main advantages of using this approach are the robustness to collision and the reduced energy consumption. During a TDMA-cycle, each cluster is allowed to transmit for a number of TDMA-slots that is proportional to the amount of traffic it has to forward. A node has to be awake only when it is in its listening TDMA-slot or its transmitting TDMA-slot if it has a packet to send.

- *A random based MAC with a* p*-persistent CSMA Scheme within a single TDMA-slot*: it manages the communication between the nodes of the transmitting cluster and the nodes of the receiving cluster within a single TDMA-slot. It offers flexibility to the introduction of new nodes and robustness to node failures. In SERAN the flexibility is obtained by having the transmitting nodes access the channel in a p-persistent slotted CSMA fashion [16]. The time granularity of this level is the CSMA-slot. Furthermore, the CSMA scheme has to support the node random selection procedure introduced in Section 3.2. The packet is sent in multi-cast over all nodes of the receiving cluster; then the receiving nodes implement a random acknowledgment contention scheme to prevent duplication of the packets. The algorithm is the following:

  1. Each of the nodes in the transmitting cluster that has a packet to send senses the channel at the first CSMA-slot with probability p. If the channel is clean, the node tries to multi-cast the packet to the nodes of the receiving cluster. If clear channel assessment (CCA) is supported, a node performs collision avoidance (CA) with a random back off time. If another transmission is detected, the node aborts the current trial to avoid collisions.

  2. At the receiving cluster, if a node has successfully received a single packet, it starts a back-off time T_{ack} before transmitting an acknowledgment. The back-off time is a random variable uniformly distributed between 0 and a maximum value called T_{maxack}. If in the interval between 0 and T_{ack}, it hears an acknowledgment coming from another node of the same cluster, the node discards the packet and does not send the acknowledgment.

  3. At the transmitting side, if no acknowledgment is received, the node assumes the packet transmission was not successful and it

multi-casts the packet at the next CSMA-slot again with probability p. The procedure is repeated until transmission succeeds or the TDMA-slot ends.

3.4 Mathematical Analysis

In this Section, a mathematical formulation of SERAN in proposed, explaining how access probability and slot duration are determined to satisfy application requirements (successful transmission probability and maximum delay), and to optimize for power consumption.

Recalling k the number of packets that the cluster has to evacuate at the beginning of a transmitting TDMA-slot, we consider the worst case scenario for collisions, when the k packets are distributed over k different nodes.

According to the p-persistent slotted CSMA scheme, a node successfully transmits a packet in the first CSMA-slot if the node accesses the channel and get it clean, while all other nodes in the same situation sense its transmission and abort the attempt. The channel can be modelled as a Bernoulli variable with parameter c.

In paper [2], a simplified analysis is presented. It is assumed that, when more than one node accesses the channel, nobody listens to the other transmissions and all packets are lost. This is comparable to a classical slotted ALOHA system and derive an upper bound for the packet loss probability due to the channel access. Under these assumptions, the probability of having a successful transmission at the first CSMA-slot is given by:

$$P_k = ckp(1-p)^{k-1} \tag{3.2}$$

while its one's complement $P_k^* = 1 - ckp(1-p)^{k-1}$ represents the probability to have again k packets to transmit in the next CSMA-slot.
Once a transmission succeeds, the cluster has $K-1$ packets to forward. Hence, the probability of successful transmission in the following CSMA-slot is $P_{k-1} = c(k-1)p(1-p)^{(k-2)}$. This allows representing the cluster behavior as a Discrete Time Markov Chain (DTMC), where the state is the number of nodes that still need to forward a packet (Fig. 3.3). The state 0 is the steady state solution of the chain.

According to the CSMA fashion, a lower bound for the packet loss probability can be found considering that all nodes are able to sense ongoing

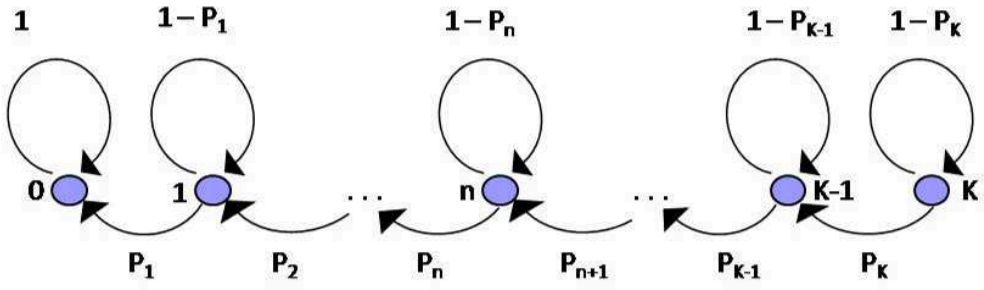

Figure 3.3: Markov Chain

transmissions avoiding to access and to collide. With this hypothesis the probability of successful transmission can be associated to the probability to have at least one node attempting to transmit the packet. Hence,

$$P_k = c[1 - (1 - p)^k] \tag{3.3}$$

Depending on the implementation of the CSMA and network parameters (e.g. network size), the real performance lays between these two bounds.

Possible failures in the sensing procedure can happen when two sensing procedures are simultaneous or a node start a transmission between the posting and the execution of a sending task of another node. To take into account possible collisions between packets we can consider the latter derived expression, introducing a factor Φ, that represents the probability of a wrong sensing when two nodes are involved.

$$P_k = c[1 - (1 - p)^k](1 - \Phi)^{[p(k-1)]} \tag{3.4}$$

Considering a transmitting node, $p(k-1)$ indicates the expected number of additional accesses to the channel in the same CSMA-slot.

Introducing a CSMA/CA mechanism, the parameter Φ is much closer to 1 and the approximation with the lower bound is satisfactory.

3.4.1 Absorption Time

In this section we determine the expected time (in number of steps) to reach the absorbing state starting from a given state between 1 and k. This is equivalent to determining the average number of CSMA-slots required for forwarding a number of packets between 1 and k.

Since expectation is a linear operator and considering that the chain can advance only one step at a time, the expected time to absorption starting from a state k is equivalent to the sum of the expected time to transition from state k to state $(k-1)$ plus the expected time to transition from state $(k-1)$ to $(k-2)$ and so on until state 0 is reached. The distribution of the required steps in the transition from the state j to the state $j-1$ is geometric of parameter $(1-P_j)$. Consequently, the expected time to transition from state j to state $(j-1)$ is bounded by:

$$\tau(j) = \frac{1}{P_j} = \frac{1}{cjp(1-p)^{j-1}} \tag{3.5}$$

The expected number of steps to reach the absorption starting from state k is:

$$\tau_k = \sum_{j=1}^{k} \tau(j) = \sum_{j=1}^{k} \frac{1}{cjp(1-p)^{j-1}} \tag{3.6}$$

Using Equation 3.4 for P_j, the expected absorption time is:

$$\tau_k = \sum_{j=1}^{k} \frac{1}{c[1-(1-p)^j](1-\Phi)^{[p(j-1)]}} \tag{3.7}$$

3.4.2 Access Probability

The access probability p is a critical parameter for the protocol performance. Recalling the Equation 3.6, it can be easily found that, for each transition from state j to $(j-1)$, the access probability that minimizes the transition time is

$$p_j = \frac{1}{j} \tag{3.8}$$

With this choice, the expected number of transmission attempts for each slot is exactly one. It maximizes channel utilization keeping a low probability of collision. A negative aspect is that the channel access probability depends on the entire network's behavior. It is not easy to implement this choice in a distributed fashion because nodes may not be aware of the fact that other nodes completed a successful transmission. Moreover there is no way to tell it to them without incurring into major overhead costs. A strategy is that each node automatically updates its access probability evaluating the expected time to complete a transition in the chain, but it is heavy to compute. A

simpler and useful choice is to fix a constant value that remains the same during the whole TDMA-slot duration for each node.

It is possible to show that finding a closed form expression for p that minimizes τ_k in Equation 3.6 is a non-trivial problem [4].
In paper [2], Bonivento et al. propose a suboptimal choice, fixing the access probability

$$p = \frac{1}{k} \tag{3.9}$$

for the whole duration of the slot, which is not optimal for the expected forwarding time, but it ensures that at the beginning of the TDMA-slot the expected number of transmission attempts for each CSMA-slot is one. Initially the channel is high utilized, while as the time advances, it will be less and less utilized.

The expressions of the absorption time become:

$$\tau_k = \frac{k}{c} \sum_{j=1}^{k} \frac{1}{j \left(1 - \frac{1}{k}\right)^{j-1}} \tag{3.10}$$

and

$$\tau_k = \sum_{j=1}^{k} \frac{1}{c \left[1 - \left(1 - \frac{1}{k}\right)^{j}\right] (1 - \Phi)^{\left[\frac{(j-1)}{k}\right]}} \tag{3.11}$$

A closed form solution for τ_k is not easy to calculate, but some useful upper and lower bounds are proved in reference [2]. In particular an upper bound for both of the expressions is:

$$\tau_k \leq \alpha k \ln(k) \tag{3.12}$$

where α is a constant.

Fig.3.4 reports a comparison between the expected absorption time against the number of packets k obtained from Equation 3.6 and the upper bound in Equation 3.12. Even if the upper bound is much higher than the real expected time, it will be useful in Section 3.4.4 to establish relations with other protocol parameters.

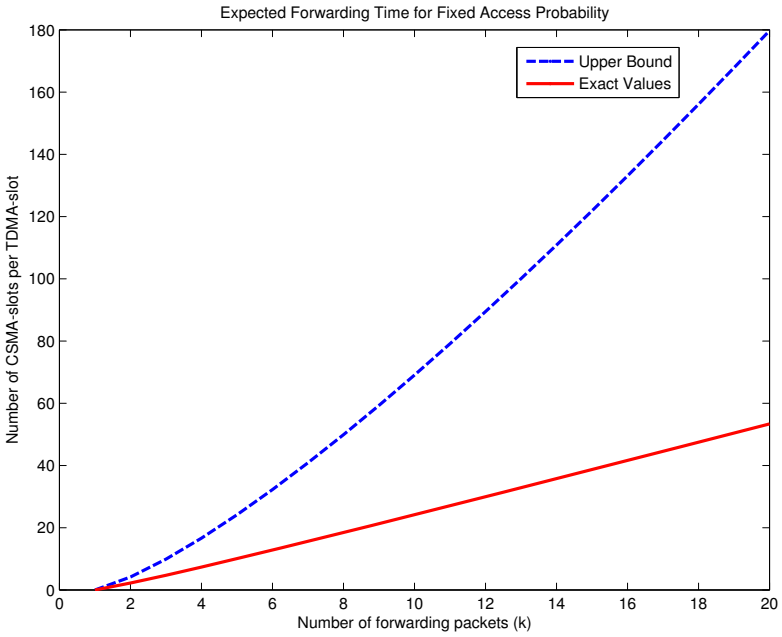

Figure 3.4: Expected forwarding time in number of CSMA-slot for $p = 1/k$

3.4.3 Scheduling Policy

The scheduling policy must consider the different traffic intensity in the network; in general it is opportune to evacuate the clusters close to the Controller first, to minimize the storage requirement in the network.
The organization of the TDMA-cycle has to refer to the same consideration. For instance, clusters closer to the Controller experience more traffic intensity and so more than one transmitting TDMA-slot can be assigned to them. Assuming the same average traffic for every cluster and referring again to the scenario in Fig. 3.1, a good scheduling policy would be to assign one transmitting TDMA-slot per TDMA-cycle to cluster 1, two transmitting TDMA-slots to cluster 2 and three transmitting TDMA-slots to cluster 4; in the same way in the other path one and two TDMA-slots allocated respectively for cluster 3 and 5.
The number of TDMA-slots in a TDMA-cycle is 9. A suitable scheduling table for such a policy is presented in Fig. 3.5.

In the general case, assuming P paths in the network and calling B_i the number of cluster in the i^{th} path, the number of TDMA-slots in a TDMA-

C	RX		RX			RX	RX		RX
5							TX	RX	TX
4	TX	RX	TX		RX	TX			
3								TX	
2		TX		RX	TX				
1				TX					
	TDMA Slot 1	TDMA Slot 2	TDMA Slot 3	TDMA Slot 4	TDMA Slot 5	TDMA Slot 6	TDMA Slot 7	TDMA Slot 8	TDMA Slot 9

Figure 3.5: Example of Scheduling Table

cycle is:

$$T_f = \frac{1}{2}\sum_{i=1}^{P} B_i(B_i + 1) \tag{3.13}$$

3.4.4 Sustainable Traffic

Because of the interleaved schedule, each cluster evacuates all the locally generated packets before receiving packets to forward.
It is necessary to ensure that the expected time for the evacuation of all the packets in a cluster is less then or equal to the duration of a TDMA-slot. If it does not happen, packets can not be disposed with catastrophic consequences on performance [4].

Consider:

- S: duration of a TDMA-slot,
- Δ: duration of a TDMA-cycle,
- λ: packet generation rate for each cluster

the number of generated packet during a TDMA-slot is:

$$k = \Delta\lambda \tag{3.14}$$

Using the upper bound of the evacuation time presented in Equation 3.12, the condition on the sustainable traffic can be expressed as:

$$S \geq \alpha\Delta\lambda \ln(\Delta\lambda) \tag{3.15}$$

Recalling $\Delta = ST_f$ Equation 3.15 can be simplified in:

$$S \leq S_{max,s} = \frac{\exp(\alpha T_f \lambda)^{-1}}{T_f \lambda} \tag{3.16}$$

Hence, the TDMA-slot duration S is upper bounded, depending on the topology (through T_f) and the packet generation rate λ.

Rewriting Equation 3.15, we can find an expression in terms of maximum sustainable traffic λ_{max} for the network, once fixed the topology and the TDMA-slot duration:

$$\lambda_{max} \ln(\lambda_{max} S T_f) = \frac{1}{\alpha T_f} \tag{3.17}$$

3.5 Energy Consumption

The total energy consumed by the network over a period of time is the combination of five components:

- energy spent for sensing the channel
- energy spent during the transmission stage
- energy spent to listen during the listening TDMA-slots
- energy spent during the receptions stage
- energy spent for the collision avoidance procedure if it is supported

For a simplified analysis, the energy consumption for receiving can be considered together with the consumption for listening, while the contributes for sensing the channel and avoiding collision together with the energy for transmission.

Listening Cost E_{ls}

Given a listening time t, the energy consumption is the sum of two costs:

- a fixed wake up cost R,
- the listening cost W.

Therefore, the listening cost is:

$$E_{\text{ls}}(t) = R + Wt \tag{3.18}$$

Now it is important to determine the number of wake-ups and the duration of the listening time during a TDMA-cycle.

Considering the reference topology, the number of wake-ups is 4. In fact, nodes in cluster 1 and 3 never wake up for listening, nodes in cluster 2 and 5 wake up once and nodes in cluster 4 wake up twice.
Referring to a general topology with N nodes per cluster and assuming that all nodes wake up in their listening slot, the total number of wake-ups N_{wu} during a TDMA-cycle is:

$$N_{\text{wu}} = \frac{N}{2} \sum_{i=1}^{P} B_i(B_i - 1) \tag{3.19}$$

Once awake, a node can keep on listening for at most the entire TDMA-slot duration S.

The total listening cost in a time $T \gg \Delta$ can be expressed by:

$$E_{\text{ls}} = \frac{T}{\Delta} N_{\text{wu}}\, N\, [R + WS] \tag{3.20}$$

Transmitting Cost E_{tx}

The energy consumption for transmissions has two components:

- packet transmission cost E_{pkt},
- acknowledgement transmission cost E_{ack}.

The global contribute depends on the average number of attempted transmissions during a TDMA-cycle.

For a transition from a state j to the state $j-1$ the number of attempted transmissions $N(j)$ is the average number of nodes attempting to transmit in a CSMA-slot multiplied by the average number of slots required for the transition.

$$N(j) = pj \frac{1}{cpj(1-p)^{(j-1)}} \tag{3.21}$$

During a TDMA-cycle the average number of attempted packet transmissions is:

$$N_{\text{pkt}} = T_f \sum_{j=1}^{k} \frac{1}{c(1-p)^{(j-1)}} \tag{3.22}$$

Since $p = 1/k$, Equation 3.22 can be simplified as:

$$N_{\text{pkt}} = T_f \frac{k-1}{c} \left[\left(1 - \frac{1}{k}\right)^{-k} - 1 \right] \tag{3.23}$$

For high values of k,

- $\left(1 - \frac{1}{k}\right)^{-k} - 1 \approx (e-1)$
- $(k-1) \approx k$

the expected number of attempted transmission is a linear function of the number of packets to transmit. Considering the relation $k = \lambda\Delta$:

$$N_{\text{pkt}} \approx \frac{T_f}{c}(e-1)\lambda\Delta = T_f A \lambda\Delta \tag{3.24}$$

The constant $A = \frac{(e-1)}{c}$ denote the average number of attempted transmissions for a single packet in a TDMA-slot.

The number of acknowledgement transmissions is linked to the number of transmitted packets by:

$$N_{\text{ack}} = T_f \lambda\Delta \tag{3.25}$$

Consequently, the transmitting cost in a time $T \gg \Delta$ is:

$$E_{\text{tx}} = \frac{T}{\Delta}[N_{\text{pkt}}\, E_{\text{pkt}} + N_{\text{ack}}\, E_{\text{ack}}] = T\, T_f \lambda [A\, E_{\text{pkt}} + E_{\text{ack}}] \tag{3.26}$$

Including a CA mechanism, the expression is slightly different. The actual number of attempted transmissions is reduced and it can be approximated by the number of successful transmissions:

$$N_{\text{tx,ca}} \approx T_f \lambda\Delta \tag{3.27}$$

The cost for collision avoidance is:

$$E_{\text{ca}} = N_{\text{ca}}(R + Wt) \tag{3.28}$$

N_{ca} and t are the number and the duration of clear channel assessments. Considering a simplified model, the number of channel assessments is:

$$N_{\text{ca}} = A_{\text{ca}} \lambda\Delta \tag{3.29}$$

Total Energy Cost

The total consumption in a time $T \gg \Delta$ with access probability $p = 1/k$ can be expressed by:

$$E_{\text{tot}} = \frac{T}{\Delta}[N_{\text{pkt}}\, E_{\text{pkt}} + N_{\text{ack}}\, E_{\text{ack}} + N_{\text{wu}}\, N(R + WS)] = \\ = T\,\lambda[A\, E_{\text{pkt}} + T_f E_{\text{ack}}] + \frac{T}{T_f} N_{\text{wu}} \left[\frac{R}{S} + W\right] \tag{3.30}$$

and with CSMA/CA:

$$E_{\text{tot,ca}} = \frac{T}{\Delta}[N_{\text{pkt}} E_{\text{pkt}} + N_{\text{ack}} E_{\text{ack}} + N_{\text{ca}}\, N(R + Wt) + N_{\text{wu}} N(R + WS)] = \\ = T\lambda[A\, E_{\text{pkt}} + T_f E_{\text{ack}} + A_{\text{ca}}(R + Wt)] + \frac{T}{T_f} N_{\text{wu}} \left[\frac{R}{S} + W\right] \tag{3.31}$$

where E_{pkt}, E_{ack}, R and W are parameters that characterize the physical layer, λ is given by the application, T_f and N depend on the network topology, so, the only protocol parameter in Equation 3.30 and 3.31 is S. Moreover, $E_{\text{tot}}(S)$ is a monotonically decreasing function of S. Hence, the problem of minimization of the energy consumption can be viewed as a problem of maximization of the TDMA-slot duration.

3.6 Latency Requirement

The clusters that experience the highest delay are the furthest from the Controller. The aim is to have the delay of packets coming from those clusters less than or equal to a given D_{max}, the requirement set by the application.

In this model we consider a time-driven data reporting method (see Section 2.4). A packet is generated only when a node wakes up in its transmitting TDMA-slot.

The idea of a uniform distribution of the generating packet rate inside a TDMA-cycle does not fit with the assumed scheduling policy (Section 3.4.3). In fact, in prefixed slots a node is in sleeping state and the sensing functionality is assumed to be turned off. The application requirement in terms of packet generating rate is kept constant not in a single TDMA-cycle but in a longer term temporal average.

Considering a packet generated from cluster 1, with the discussed scheduling policy the worst case of delay is when it is generated at the beginning of the

transmitting TDMA-slot and cluster 4 forwards it to the Controller at the end of its own transmitting TDMA-slot, that is $3S$.

Generalizing to the case of P path with B_i clusters per path and defining

$$B = \max_{1..P} B_i \tag{3.32}$$

the worst case delay is:

$$D = BS \tag{3.33}$$

Consequently, the requirement on the TDMA-slot duration S is:

$$S \leq S_{\max,\mathrm{d}} = \frac{D_{\max}}{B} \tag{3.34}$$

If during a TDMA-slot not all the packets are forwarded, latency over the deadline is observed. It can be useful to model this phenomenon of outage referring to the Discrete Time Markov Chain (DTMC) presented in Section 3.4.

Using the Central Limit Theorem, the distribution of the time to forward $\lambda\Delta$ packets is a normal variable whose mean and variance is given by the sum of the expected times and variances to advance a step in the chain.
Let τ_{ev} be the time to evacuate $\lambda\Delta$ packets and m_{ev} and var_{ev} its mean and variance. Consequently, τ_{ev} can be modelled as $\tau_{ev} \sim N(m_{ev}, \sigma^2_{ev})$. In case there is no carrier sense and collision avoidance:

$$m_{ev} = \sum_{j=1}^{\lambda\Delta} \frac{1}{cpj(1-p)^{j-1}} \tag{3.35}$$

$$\sigma^2_{ev} = \sum_{j=1}^{\lambda\Delta} \frac{cpj(1-p)^{j-1}}{[1-cpj(1-p)^{j-1}]^2} \tag{3.36}$$

while in the general case:

$$m_{ev} = \sum_{j=1}^{\lambda\Delta} \frac{1}{c[1-(1-p)^j](1-\Phi)^{[p(j-1)]}} \tag{3.37}$$

$$\sigma^2_{ev} = \sum_{j=1}^{\lambda\Delta} \frac{c[1-(1-p)^j](1-\Phi)^{[p(j-1)]}}{[1-c[1-(1-p)^j](1-\Phi)^{[p(j-1)]}]^2} \tag{3.38}$$

Consequently, the probability of outage in a given TDMA-slot can be approximated by:

$$P_r[\tau_{ev} \geq S] \approx \frac{1}{2} erfc\left(\frac{S-m_{ev}}{\sqrt{\sigma^2_{ev}}}\right) \tag{3.39}$$

where $erfc()$ is the complementary error function defined as:

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2}\, dt \tag{3.40}$$

3.7 Error Rate Requirement

In the following a method to establish the constraint $PRR \geq PRR_{min}$ is discussed.

3.7.1 Problem Formulation

The reference model is the same Discrete Time Markov Chain (DTMC) presented in Section 3.4, where the state is the number of packets that still need to be forwarded.

We recall the parameters:

- S: TDMA-slot duration (in number of CSMA-slots)
- k: number of packets to send in the cluster
- P_n: probability of transition from the state n to the state $n-1$

P_n denotes the probability of successful transmission when there are n packets to transmit.

We define $P(n, S, k)$ the probability to be in the state n after a number S of steps in the DTMC. In other words, $P(n, S, k)$ represents the probability of losing n of k packets. If there are still n packets left after S CSMA-slots, this packets are discarded.
Consequently, the PRR can be written as:

$$PRR = \sum_{n=0}^{k} P(n, S, k) w(n) \tag{3.41}$$

where

$$w(n) = \frac{k-n}{k}$$

3.7.2 P(n,S,k) Evaluation

It is hard to evaluate easily $P(n, S, k)$ for a TDMA/CSMA with ACK and retransmission.

We can consider a specific example to understand a general law. Let k be 3 and S equal to 4, we can try to determinate $P(1, 4, 3)$. The associated Markov Chain is in Figure 3.6.

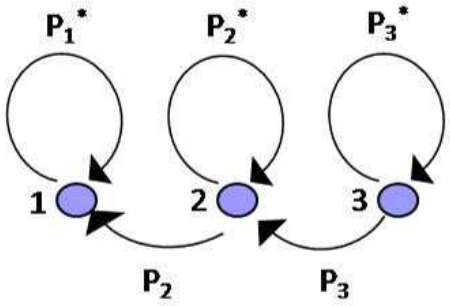

Figure 3.6: Markov Chain for $n = 1, S = 4, k = 3$

$P(1, 4, 3)$ is given by the sum of the probabilities of all the possible paths that start from the state 3 and end in 1, in exactly 4 steps. So:

$$P(1,4,3) = P_2P_3[P_1^*P_1^* + P_2^*P_2^* + P_3^*P_3^* + P_1^*P_2^* + P_1^*P_3^* + P_2^*P_3^*] \quad (3.42)$$

The product P_2P_3 is present in all the paths, while, inside the brackets, there are all the *combinations with repetition* of the elements P_1^*, P_2^*, P_3^*, taken 2 at a time.

It is possible to generalize this approach, defining a set

$$V(n) = \{P_n^*, P_{n+1}^*, ..., P_k^*\} \quad (3.43)$$

and a matrix that contains all the N_c combinations with repetition of the elements in $V(n)$, taken in groups of $n + S - k$.

$$A(n) = [a_{h,j}]_{N_c}^{S-k+n} \quad (3.44)$$

In the previous example,

$$V(1) = \{P_1^*, P_2^*, P_3^*\} \quad (3.45)$$

$$A(1) = \begin{bmatrix} P_1^* & P_1^* \\ P_2^* & P_2^* \\ P_3^* & P_3^* \\ P_1^* & P_2^* \\ P_1^* & P_3^* \\ P_2^* & P_3^* \end{bmatrix}$$

The general expression for $P(n, S, k)$ is given by:

$$P(n, S, k) = \prod_{m=n+1}^{k} P_m \left[\sum_{j=1}^{N_c} \prod_{h=i}^{(S-k)+n} a_{h,j} \right] \tag{3.46}$$

3.7.3 Packet Reception Rate

Implementing Equations 3.41 and 3.7.3 in Matlab, it is possible to evaluate the PRR of SERAN protocol, and to obtain bounds for the protocol parameters.

We implement the simplified model for P_n, using the expression:

$$P_n = cnp(1-p)^{n-1} \tag{3.47}$$

As already said this gives us a worst case scenario for the CSMA, but the analysis holds for the other models as well.

The Fig. 3.7 represents the PRR as function of S, varying the number of packets k.

In Fig. 3.8, the relation between the number of packets K and the value of S that gives a fixed PRR is presented. For instance, it shows that 3 packets require a TDMA-slot of about 12 CSMA-slots, to guarantee $PRR = 95\%$, 10 slots for $PRR = 90\%$ and 8 slots for $PRR = 85\%$.
The computation for the Equation with high values of k is heavy and long. Hence, the Fig. 3.7 and 3.8 can not be easily drawn for $k > 10$. On the other side, the relation between S and k for fixed PRR is almost linear and the behavior for $k > 10$ can be estimated.

An upper bound for the PRR can be determined in a CSMA scenario with perfect collision avoidance, where:

$$P_k = c[1 - (1-p)^k] \tag{3.48}$$

In Fig. 3.9, a comparison between upper and lower bound is shown for $k = 3$.

These results will be validated in Section 5.2.

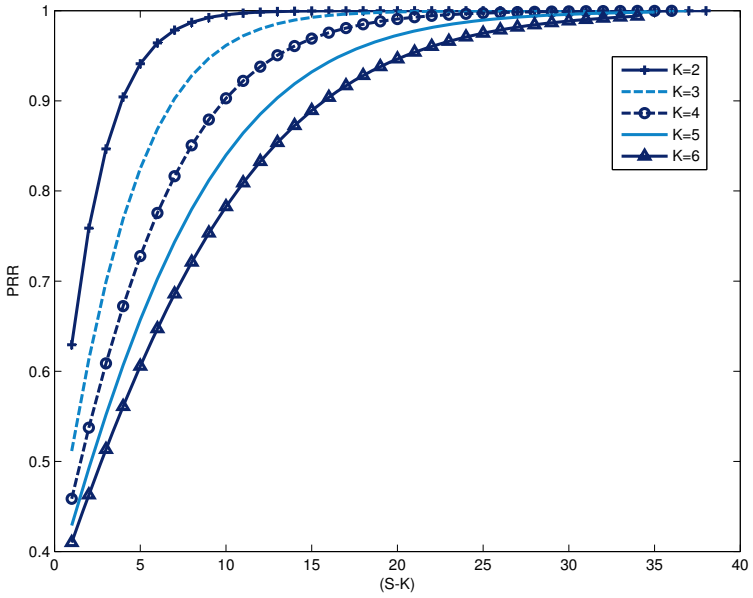

Figure 3.7: PRR vs.$(S-k)$, for different values of k

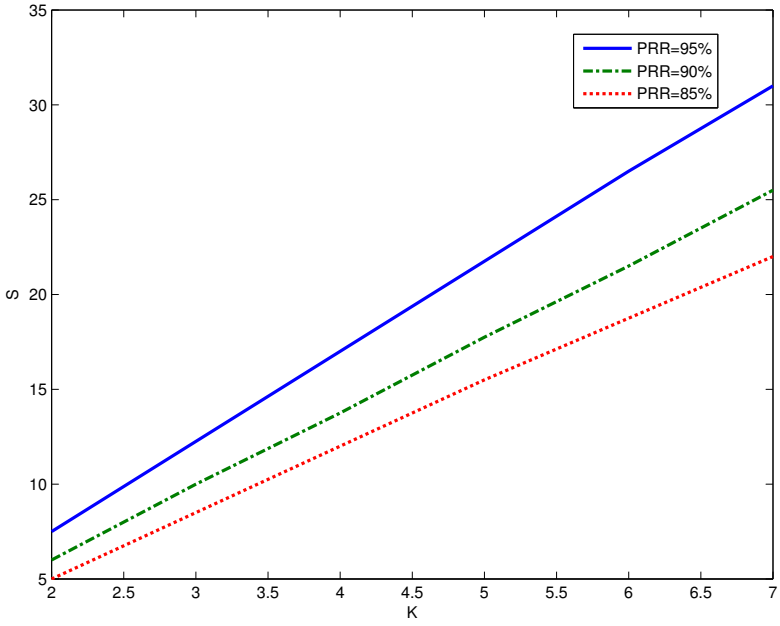

Figure 3.8: S vs.k, for fixed values of PRR

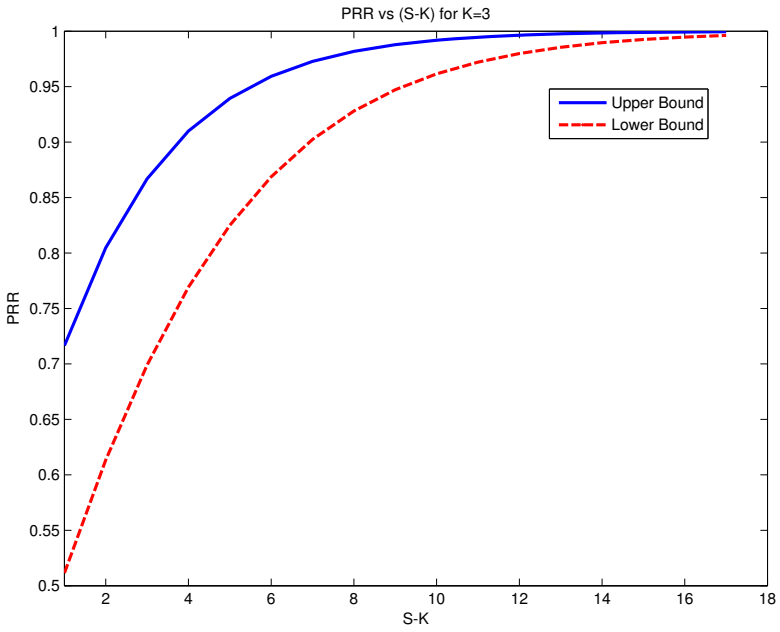

Figure 3.9: PRR vs.$(S-k)$, for $k=3$ - upper and lower bounds

3.8 Optimization Problem

Once established mathematical relations in terms of energy cost function, latency and error rate requirements, it is possible to define the choice of the parameters that optimize the performance of SERAN. We recall the reference problem:

$$\begin{aligned} \text{minimize} \quad & E_{\text{tot}} \\ \text{subject to} \quad & PRR \geq PRR_{\min} \\ & D \leq D_{\max} \end{aligned} \tag{3.49}$$

In Appendix A, we report the complete list of parameters in the protocol. Here, we recall the parameters that are directly influenced by MAC & Routing layers of SERAN:

- access probability p,
- TDMA-slot duration S.

The optimization algorithm works in two phases: an *off-line* and an *on-line* optimization.

Off-line optimization

According to the analysis in Section 3.4.2, an appropriate choice of the access probability is:

$$p = \frac{1}{k} \tag{3.50}$$

where k is the average number of packets that a cluster sends in a TDMA-slot. Recalling $k = \lambda\Delta$, and $\Delta = ST_f$, we have:

$$p = \frac{1}{S\,\lambda\,T_f} \tag{3.51}$$

The only variable is S.

From Section 3.5, it is known that the energy cost is a monotonically decreasing function of S. Without considering any application constraint, S can be increased until it reaches the maximum sustainable traffic $S_{max,s}$ derived in Section 3.4.4.

As shown in Section 3.6, the maximum delay requirement D_{max} provides an upper bound for S, given by: $S_{max,d} = \frac{D_{max}}{B}$
Hence, we will fix:

$$S = min\{S_{max,s}, S_{max,d}\} \tag{3.52}$$

This choice has to be compared with the error rate requirement, considering the analysis in Section 3.7. If the value of S is in compliance with the constraints, it is fixed as initial TDMA-slot duration.

On-line optimization

The network starts operating with the selected optimal parameters. Real-time, the Controller determines the actual values of packet delay and error rate and modifies the value of S, to enhance the performance. There are various possible situations:

1. The measured delay can be lower than the worst-case scenario used for the initial optimization. In this case, if the error rate constraint allows it, the protocol is entitled to increase the TDMA-slot duration.

2. The delay is on the boundary, but PRR can be higher than the minimum threshold. However, S can be slightly increased and the generated lost packets rate due to the overtaken delay constraint, can be supported by the network.

Chapter 4

Protocol Implementation

In this chapter, we introduce the hardware and software technologies used to implement the WSN test-bed environment. We report some implementation tricks and procedure to guarantee the correct behavior of SERAN protocol and also difficulties and drawbacks. Eventually, we describe a specific model to evaluate the network lifetime for the presented platform.

4.1 Hardware Technologies

A sensor network is an *embedded* system, or rather a digital system committed to specific duties. Each node consists of a **sensor board** and a **programming board** [8]. The sensor board could be differentiated by the specific kind of sensor: light, temperature, humidity, but also distance tracking or GPS receiver. The programming board supplies wireless communication capabilities between nodes or wired between a node and a base station (PC). The benchmark is the IEEE 802.15.4 radio standard, with low data rate (around 250 kbps). A node is equipped with a microcontroller (8-16 bit) and low storage memories.

4.1.1 Tmote Sky platform

Tmote Sky is a node platform for low power and high data-rate sensor network applications designed with the dual goal of fault tolerance and development ease [29]. Designed at the University of California, Berkeley, it is the successor of the popular TelosA and TelosB research platforms. The Tmote Sky platform offers vertical integration between the hardware and

the TinyOS operating system. The Tmote Sky module has integrated sensors, radio, antenna, microcontroller and programming capabilities. The low power operation of the module is due to the low power TI MSP430 microcontroller. This 16-bit RISC processor features low active and sleep current consumption. In order to minimize power consumption, the processor in sleep mode during majority of the time, wakes up as fast as possible to process, then returns to sleep mode again. Tmote Sky provides an easy-to-use USB protocol from FTDI to communicate with the host computer for programming, debugging and data collection. It features the Chipcon CC2420 radio for reliable wireless communications [32], which is high configurable for many applications with the default radio setting providing IEEE 802.15.4 [22] compliance. The radio provides fast data rate and robust signal. It is controlled by the microcontroller through the SPI port and can be shut off for low power duty cycled operation. Tmote Sky's internal antenna is an Inverted-F microstrip design, with a pseudo omnidirectional pattern that may attain 50 meter range indoors and up to 125 meter range outdoors.
The picture in Fig. 4.1 shows a Tmote Sky platform, compared in size with a Swedish coin.

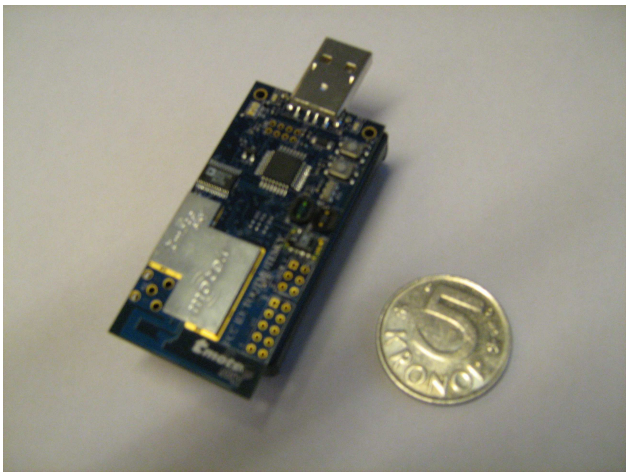

Figure 4.1: Tmote Sky platform

4.2 Software Technologies

The link between hardware platform and software equipment is stricter than the other technologies, because of the particular resource constraints (e.g. low power consumption, reduced memory). It follows the demand of specific *ad hoc* software technologies. Hence, operating systems for WSN nodes are typically less complex than general-purpose operating systems. In general operating systems in WSNs should fulfill these requirements:

- **Robustness**: once deployed, a sensor network must work unattended for months or years;
- **Low resource usage**: sensor network nodes include very small RAM, and run off batteries;
- **Multiple service implementation**: applications should be able to choose between various implementations;
- **Adaptability to evolutions**: mote hardware is in constant evolution; applications and most system services must be portable across hardware generations;
- **Adaptability to application requirements**: applications have very different requirements in terms of lifetime, communication, sensing, etc. On the other side, they do not require interactivity in the same way as applications for PCs and the operating system does not need to include support for user interfaces.

4.2.1 TinyOS

TinyOS is an embedded operating system expressly designed for WSNs. The basic concepts behind TinyOS are:

- application compiled and used for programming a single node.
- *Hurry Up and Sleep Philosophy*; so when a node wakes up for an event, it has to execute the associated action as fast as possible, then go back to sleep.

Because of the extremely limited resources of the hardware platforms, it is difficult to virtualize system operation to create the kinds of system abstractions that are available in more resource rich systems. The concurrency model and abstractions provided by operating system therefore significantly impact the design and development process.

The TinyOS 2.x family is the latest stable branch of the operating system and is used in this section to describe the basic design principles. The TinyOS development environment directly supports a variety of device programmers and permits programming each device with a unique address attribute without having to compile the source code each time. The TinyOS system, libraries and applications are written in nesC, a version of C that was designed for programming embedded systems.

The characteristics of TinyOS 2.x are listed as [24]:

- **Resource constrained concurrency**

  Concurrency is the main important software challenge. The system manages several components, as sensors, ADCs, radio and flash memory. Generally, an operation is started on a device, which runs concurrently with the main processor until generating a response. Meanwhile, other devices may also need service, requiring the system to manage several event streams. A conventional OS uses multiple threads, each with its own stack. The thread dedicated to a device issues a command and then sleeps or polls until the operation completes. The OS switches among threads by saving and restoring their registers, and threads coordinate with others by using shared variables as flags and semaphores. This is problematic for embedded designs because multiple stacks must be kept in memory and each thread can potentially interact with any other whenever it accesses a shared variable. This can lead to deadlocks, requiring complex schedulers to meet real-time requirements and deadlines. TinyOS attacks the problem by offering different levels of concurrency, in a structured event-driven execution.

- **Structured event-driven execution**

  TinyOS provides a structured event-driven model. A complete system configuration is formed by 'wiring' together a set of components for a target platform and application domain. Components are restricted objects with well-defined interfaces, internal state, and internal concurrency. Primitive components encapsulate hardware elements (radio,

ADC, timer, bus ...). Their interface reflects the hardware operations and interrupts; the state and concurrency is that of the physical device. Higher-level components encapsulate software functionality, but with a similar abstraction. They provide commands, signal events, and have internal handlers, task threads, and state variables. This approach accommodates hardware evolution, including major changes in the hardware/software boundary, by component replacements. Its memory footprint is small, despite supporting extensive concurrency, requiring only a single stack and a small task queue. However, the modular construction provides flexibility, robustness, and ease of programming. A restricted form of thread, called a task, is available within each component, but interactions across components are through explicit command/event interfaces. The wiring of components and the higher priority of asynchronous events over tasks permit the use of simple schedulers, and in TinyOS 2.0 even the scheduler is replaceable.

- **Components and bidirectional interfaces**

  TinyOS supports component composition, system-wide analysis, and network data types. A component has a set of bidirectional command and event interfaces implemented either directly or by wiring a collection of subcomponents. The compiler optimizes the entire hierarchical graph, validates that it is free of race conditions and deadlocks, and sizes resources. The TinyOS community has developed plug-ins for integrated solutions and several visual programming environments for this component-based programming style. Network data types simplify protocol implementation. While network packets have a particular specified format, data representation in a computer program depends on word width and addressing of the host processor, so most protocol code contains machine-dependent bit-twiddling and run-time parsing. Because TinyOS uses network types with a completely specified representation, the compiler provides efficient access to packet fields on embedded nodes, as well as Java and XML methods for packet handling on conventional computers and gateways.

- **Split-phase operations**

  Split-phase operations are a typical use of bidirectional interfaces. TinyOS has a non-preemptive nature and does not support blocking opera-

tions. This means that all long-latency operations have to be realized in a split-phase fashion, by separating the operation-request and the signaling of the completion. The client component requests the execution of an operation using command calls which execute a command handler in the server component. The server component signals the completion of the operation by calling an event handler in the client component. In this way, each component involved in the interaction is responsible for implementing part of the split-phase operation.

- **Sensing**

  TinyOS 2.0 provides sensor drivers, which scale from low-rate, low-power sampling of environmental factors to high-rate, low-jitter sampling of vibration or acceleration. Drivers handle warm-up, acquisition, arbitration, and interface specifics. Since sensor selection is closely tied to the application and mechanical design, drivers must be easily configured and tested, rather than dynamically loaded after market.

- **Communication and networking**

  Communications and networking have driven the TinyOS design as available radios and microcontroller interfaces evolved. The communication subsystem has to meet real-time requirements and respond to asynchronous events while other devices and processes are serviced. The TinyOS 2.0 radio component provides a uniform interface to the full capabilities of the radio chip. The link-level component provides rich media access control (MAC) capabilities, including channel activity detection, collision avoidance, data transfer, scheduling, and power management, allowing networking components to optimize for specific protocols. The TinyOS community has developed networking layers targeted at different applications with different techniques for discovery, routing, power reduction, reliability, and congestion control. These networking layers provide higher-level communication services to embedded applications with a TinyOS programming interface. The most widely used services for WSNs are reliable dissemination, aggregate data collection, and directed routing. To collect and aggregate data from the WSN, nodes cooperate in building and maintaining one or more collection trees, rooted at a gateway or data sink. Each node gathers link-quality statistics and routing cost estimates through all

kinds of communication, including routing beacons, packet forwarding, and passive traffic monitoring.

- **Storage**

  Nonvolatile storage is used in WSNs for logging, configuration parameters, files, and program images. Several different kinds of Flash memory are used with different interfaces and different protocols. Lower-layer TinyOS components provide a block storage abstraction for various Flash chips and platform configurations, with one or more application-level data services provided on this substrate.

4.3 Time Synchronization and Network Initialization

Time synchronization is a critical issue in distributed WSN. The TDMA-based MAC component of SERAN imposes a strict synchronization between clusters, while the slotted CSMA component requires robustness against clock drift between nodes inside a cluster.

4.3.1 Token Passing Procedure

SERAN implements a token passing procedure that:

- ensures synchronization between nodes and clusters;
- allows initializing and self configuring to the optimal working point;
- allows for the addition of new nodes.

A token is a particular message that carries the information on the duration of a TDMA-slot and a TDMA-cycle, the transmitting and receiving schedule of a TDMA-cycle, a synchronization message carrying the current execution state of the TDMA-cycle.
The Controller has all the information to calculate the optimal set of parameters, consequently, it is able to generate a token before the network starts operating. The network initialization algorithm works as follows:

1. When the network starts, all nodes are awake and listening. Nodes remain in this state and cannot transmit before receiving the token.

2. The Controller multi-casts the token to all nodes of one of the connected clusters. In general this is the first in the scheduling table. In the reference example (Fig. 3.1), assume the selected cluster is the cluster 4.

3. Nodes of the selected cluster read the information on scheduling and duration of TDMA-slot and TDMA-cycle. Moreover, each node acquires the information about the global time and launches periodic timers for CSMA and TDMA slots. In the meantime, a random back-off timer starts for each node before sending an acknowledgement.

4. The first node that expires the back-off time sends the acknowledgement to the Controller and becomes *the token forwarder*. Then, all nodes in the cluster go to sleep.

5. At the beginning of the second TDMA-slot the token forwarder wakes up and immediately multi-casts the token to all nodes in the next cluster (i.e. cluster 2).

6. With the same random acknowledgement-based scheme, a node is elected token forwarder for nodes in the following cluster (i.e. cluster 1).

7. After the first branch of the routing tree is explored, the Controller sends a token to cluster 5, the new branch is explored, and so on.

Information about routing and TDMA-slot duration needs also to be updated during the network operation. Hence, the Controller periodically performs a *token refreshing procedure*. It is a critical phase because the Controller needs to ensure that all nodes in the network receive the new token. First, all nodes should be in listening state and, moreover, when the token is forwarded along the network, the scheduled packet transmission has to be interrupted, to avoid collisions.

If the frequency of token refreshing is high, the response of the protocol to variation in the surrounding conditions is faster and the adaptability increases. On the other side, the procedure is costly in terms of energy consumption. In our implementation, we do not consider strict requirements on the protocol adaptability, while the minimum energy consumption is fundamental. We think that a suitable choice of the refreshing period is 20 TDMA-cycles.

Nodes are informed by the Controller about the refreshing period through the first token passing. We can also suppose that the Controller can modify this value and update it during each token refreshing, according to the network behavior.

4.4 MAC/Routing Implementation

SERAN MAC and Routing protocols are introduced in Section **??**. Here, we discuss some details referred to the specific implementation in TinyOS 2.x on Tmote Sky motes. In Fig. 4.2 we present a useful flow diagram, describing the node activities in transmission and reception.

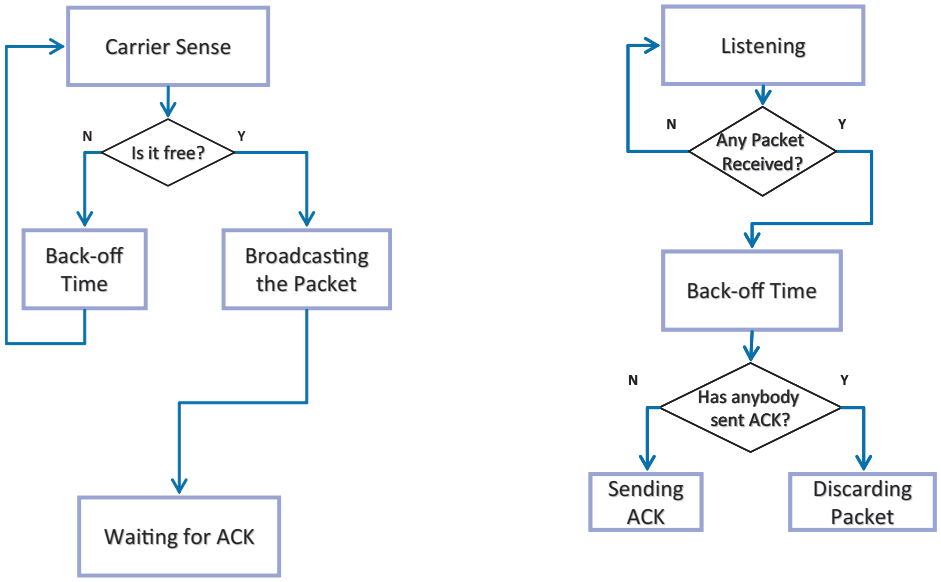

Figure 4.2: Flow diagram of SERAN Protocol: transmission side (left) and receiving side (right)

We adapted the SERAN protocol stack to the CC2420 Radio stack layers implemented in TinyOS 2.x (TEP 126 of Ref. [30]) and reported in Fig. 4.3:

- *ActiveMessageP*: This is the highest layer in the stack, responsible for filling in details in the packet header and providing information about the packet to the application level.
- *UniqueSend*: This layer generates a unique Data Sequence Number (DSN) byte for the packet header. This byte is incremented once per

outgoing packet, starting with a pseudo-randomly generated number. A receiver can detect duplicate packets by comparing the source and DSN byte of a received packet with previous packets. DSN is defined in the 802.15.4 specification [31].

- *PacketLink*: This layer provides automatic retransmission functionality and is responsible for retrying a packet transmission if no acknowledgement was heard from the receiver.

- *CC2420AckLplP / CC2420NoAckLplP*: These layers provide asynchronous low power listening implementations. *AckLplP* implementation supports acknowledgement gaps during the preamble. The *NoAckLplP* implementation does not support it.

- *UniqueReceive*: This layer maintains a history of the source address and DSN byte of the past few packets it has received, and helps filter out duplicate received packets.

- *TinyosNetworkC*: This layer allows the TinyOS 2.x radio stack to interoperate with other non-TinyOS networks.

- *CsmaC*: This layer is responsible for defining 802.15.4 FCF byte information in the outbound packet, providing default back-off times when the radio detects a channel in use, and defining the power-up/power-down procedure for the radio.

- *TransmitP/ReceiveP*: These layers are responsible for interacting directly with the radio through the SPI bus, interrupts, and GPIO lines.

The main part of SERAN code belongs to the Application Layer of the stack, but some parts are mixed with the other layers. Awake and sleeping states are governed by the *SplitControl* interface. It is used for switching between the on and off power states of the component providing it, with the functions *start()* and *stop()*. The implementation of the *SplitControl* interface is defined in the *CC2420CsmaP* component.

The TDMA and CSMA slotted structure is realized with the instances of the TinyOS 2.x *TimerMilliC* component, wired to the interface $Timer < Tmilli >$ that provides a set of commands to handle periodic events. It ensures a precision in order of milliseconds.

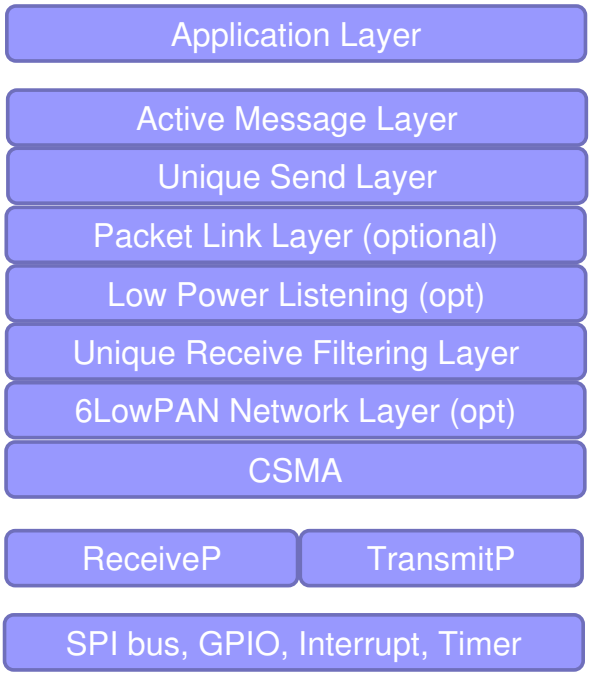

Figure 4.3: Architectural layers of CC2420 Radio stack

Back-off timers at MAC layer are managed by the TinyOS 2.x interface $Alarm < T32khz, uint32_t >$, with a 32 microseconds precision, wiring the component *AlarmMultiplexC*.

The random components are achieved by the *Random* interface, provided by the *RandomC* configuration, through the function *rand16()*, that returns a 16-bit pseudo-random number. The configuration *RandomC* maps the standard number generator to a specific algorithm called *multiplicative linear congruential generator* (MLCG) [33]. Once the code is compiled the sequence of the generated random values is the same for each repetition of the experiment. To increase the randomness, we combined random numbers with the current value of one of the running timers, obtained with the function *getNow()*. For instance, that procedure is used to generate the channel access probability p. The interfaces *Send* and *Receive* define commands and events in transmission and reception of data messages.

4.4.1 Acknowledgement Mechanism

The default MAC code of TinyOS 2.x implements two types of acknowledgements:

- Hardware ACK, directly implemented in the transceiver.
- Software ACK, described by the specific interface *PacketAcknowledgements*.

Originally, the CC2420 radio stack only used hardware generated auto acknowledgements provided by the CC2420 chip itself. This led to some issues, such as false acknowledgements where the radio chip would receive a packet and acknowledge its reception and the microcontroller would never actually receive the packet. The current CC2420 stack uses software acknowledgements, which have a higher drop percentage. When used with the *Send* and *Receive* interfaces, dropped acknowledgements are more desirable than false acknowledgements. Received packets are always acknowledged before being filtered as a duplicate.

The peculiarity of SERAN acknowledgement mechanism is that a node, during the random back-off time, should listen to possible ACK transmissions coming from other nodes, before sending its ACK. The implementation of this mechanism above the standard Software ACK is very tricky. After various tests, we decided to implement the acknowledgement mechanism using the configuration of *Send* interface, the same of the data packet transmissions.

We measured the minimum back-off granularity in the order of $\Delta_t = 2$ ms. It means that, if a node decides to send the acknowledgement, the other nodes in the same cluster require at least 2 milliseconds to receive and elaborate it, before sending their ACK.

4.5 Network Lifetime

Network lifetime is the time span from the deployment to the instant when the network is considered nonfunctional [27]. When a network should be considered nonfunctional is, however, application-specific. For instance, it can be the instant when the first sensor dies, a percentage of sensors die, the network partitions, or the loss of coverage occurs. In this analysis we refer to the battery lifetime of the sensor nodes in the various clusters.

To achieve a good estimation of the network lifetime, we should refer to a model for the instantaneous power consumption of the motes. In this study the platform is represented by Tmote Sky mote, with CC2420 Chipcon wireless transceiver.

4.5.1 Characterization of the CC2420 Transceiver

The consumption pattern can be determined considering different states of operation. In particular the CC2420 transceiver supports four states [25]:

- **Shutdown**: the chip is completely deactivated and the clock is switched off. It reacts only to a particular startup strobe.
- **Idle**: the chip can acquire commands and the clock is turned on.
- **Transmit**: chip, clock and radio are turned on in transmitting mode
- **Receive**: chip, clock and radio are turned on in receiving mode.

The data sheet of CC2420 component specifies typical, maximum and minimum current consumption. The state diagram in Fig. 4.4 reports typical values of current I and transition times.

The average power consumed is $P = VI$. Considering a voltage supply $V = 3V$ [1], the power consumption for each state is evaluated in Table 4.1.

Interesting parameters are the energy per bit $E' = P/R$, and the average energy per message $E = E'L$, where R is the transmission rate of the platform and L the average length of a message. These parameters are evaluated both for transmission and reception, as reported in Table 4.2. Notice that for Tmote Sky is $R = 250$ kbps and the average length of a message is supposed $L = 30$ bytes (equal to 240 bits).

[1] equivalent to the voltage of 2 full charged AA batteries

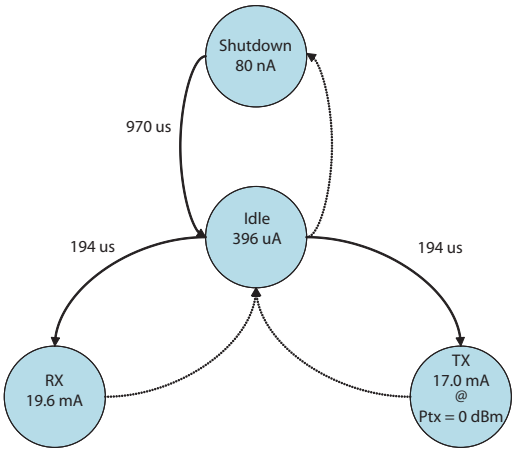

Figure 4.4: State diagram and typical current consumption and transition times for CC2420 transceiver

Table 4.1: Node power consumption

State	Power Consumption
Shutdown	$120\,nW$
Idle	$594\,\mu W$
Transmission	$26.1\,mW$
Reception	$29.1\,mW$

We can easily evaluate an average message transmission time $t \simeq 940\mu s$.

Furthermore, studies in wireless micro-sensor networks has shown that the transient energy when switching from one mode to another impacts the total power consumption [25]. The order of magnitude is not significant but with a high number of state transitions can not be neglected.

4.5.2 Battery Model

The power consumption analysis is necessary to determine the network durability. In order to convert the mean power consumed by a protocol into the lifetime of a node, a battery model needs to be defined. In this work it is considered the simplified model developed in [26] for alkaline batteries.

Table 4.2: Node energy specification

Operation	Energy per bit	Energy per message
Tx mode	$102\,\mu J/bit$	$24.5\,mJ$
Rx mode	$114\,\mu J/bit$	$27.3\,mJ$

A single battery has a lifetime T given by:

$$T = \frac{E}{P + P_{leak}}(years) \tag{4.1}$$

where E is the total energy in Wh of a battery, P is the average power consumption and P_{leak} is the leakage power, assumed equal to 10% of the full energy E during one year:

$$P_{leak} = \frac{0.1E}{24365} \tag{4.2}$$

In the specific case of AA LR26 alkaline, $E = 3.12Wh$.

Hence, T is depending only on P according to the expression:

$$T = \frac{1}{7614P + 0.1}(years) \tag{4.3}$$

As immediate observation, even with zero power consumption, the model limits the lifetime of alkaline batteries to ten years.

4.6 Drawbacks

The protocol implementation presents two main drawbacks.

Collisions

Collisions are fundamentally related to failures in the CSMA mechanism. A collision can happen if two or more nodes listen to the channel at the same time and find a clean channel.

Duplicated Packets

There are two main reasons of duplication:

- A packet is correctly received but the ACK does not arrive to the sender (e.g. for bad channel condition). Hence, the sender retransmits the packet.

- A node in the receiving cluster does not hear the ACK transmitted from another node. Hence, in the receiving cluster more than one node forwards the same packet.

In conclusion, the presence of collisions and packets duplication depends on the contention-based MAC and it is a common problem with other cross-layer solutions.

Chapter 5

Experimental Results

This Chapter describes the experimental setup and some results obtained by running the SERAN protocol on a real scale test-bed.

5.1 Network Setup

The experiments have been conducted in a test-bed environment, realized along corridors of the building where the Automatic Control Laboratory is located[1](see Fig. 5.1).

The deployment is done in such a way to avoid obstacles between clusters and inside a cluster, this allows us to reproduce good operative conditions for the protocol, however, the surrounding space is typical of an indoor application.

We reproduced the reference topology as in Fig. 5.2. The clusters are placed at regular intervals of 5 meters. Each cluster is composed by $k = 3$ motes, deployed without a specific pattern inside a circle with a 1 meter radius.
The complete network is composed by 16 T-mote Sky sensor nodes, including a sink mote, connected to a personal computer acting as Controller.

5.2 Validation

As first step of the validation, the performance of the network are tested in terms of PRR, referring to the theoretical analysis proposed in Section 3.7.

[1] 8^{th} floor, Osquldas väg 10, KTH Main Campus, Stockholm

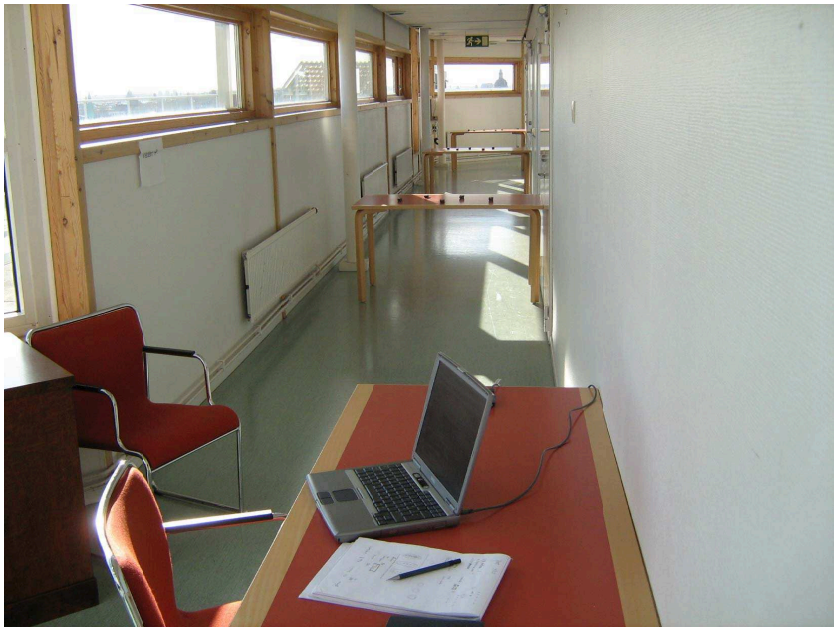

Figure 5.1: Test-bed

The CSMA-slot duration is $\delta = 100ms$, which we verified to be enough for the nodes to exchange a packet and randomly contend the acknowledgment (see also Section 4.4.1).

We evaluated some graphics of the PRR against the TDMA-slot duration S, taking as reference the upper and lower bound of the probability of successful transmission. In Fig 5.3 the comparison between ideal and measured values is shown for a number of packets $k = 3$.
As in the theoretical approach, the performance is evaluated only for single

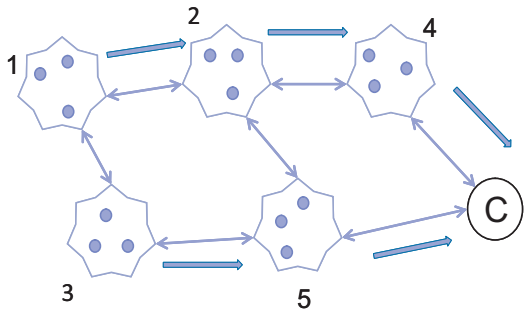

Figure 5.2: Network Topology

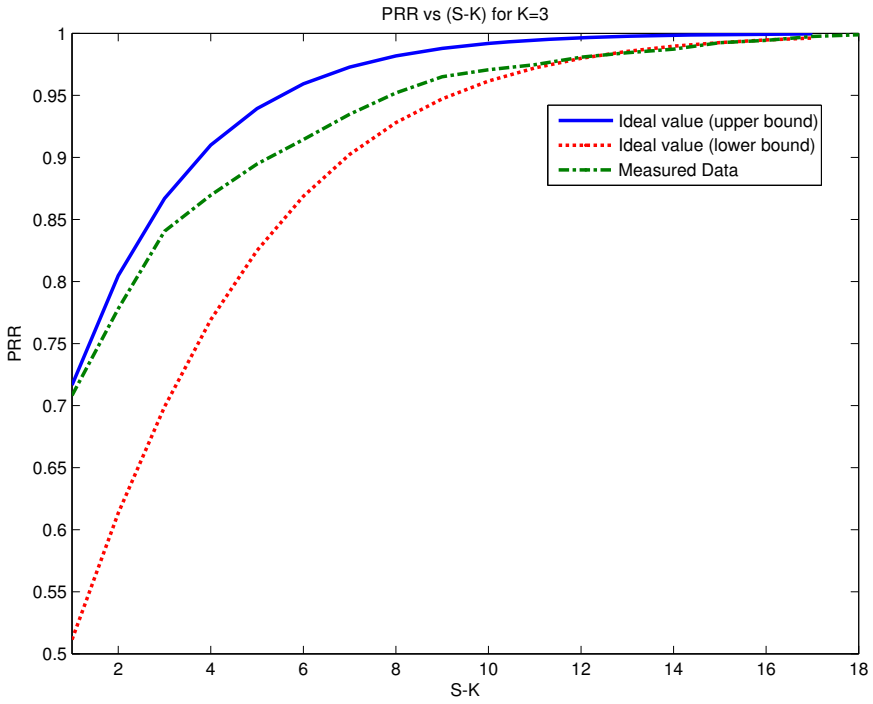

Figure 5.3: Packet Reception Rate vs. TDMA-slot duration for $k = 3$

hop, considering an average of all the transmissions in the network.

The measured PRR respects the theoretical behavior. Only a slower attainment of maximum PRR is noticed. This is due to various practical aspects of implementation, but the main reason is that the ideal model does not consider the acknowledgements. ACKs introduce problems in terms of collisions, channel losses and duplicated packet. However, the strength is that the network is able to achieve PRR values drawn near the 100%.

A second test consists in fixing relaxed constraints in terms of PRR and delay and evaluate the performance of the protocol. We considered a latency constraint on the maximum end-to-end delay of $D = 9s$ and a low cluster packet rate of $\lambda = 1pkt/10sec$.
Under these assumptions the off-line optimization gives a TDMA-slot duration $S = 3300ms$. The TDMA-cycle is $\Delta = 29,7s$. Consequently, there is an average of about 3 packets per TDMA-slot and each slot is composed by 33 CSMA-slots. According to the previous validation this setup should ensure 100% of packets delivered in the single hop.

The Table 5.1 presents the average values of PRR, delay and duty cycle registered during the experiment.

Table 5.1: Validation: average values of PRR, delay and duty cycle

Cluster Rate λ	TDMA Slot S	Cluster Number	PRR	Average Delay	Duty Cycle
1pkt/10sec	3300ms	1	99.2 %	7111ms	0.2%
		2	99.6 %	3640ms	1.9%
		3	99.5 %	3560ms	0.2%
		4	100 %	148ms	3.3%
		5	100 %	139ms	1.9%

Nodes of clusters 4 and 5 have optimal conditions of PRR and minimum delay. The worst performance in terms of PRR and delay is for cluster 1. Clusters 2 and 3 show intermediate values. The results indicate that PRR and delay depends mainly on the distance in number of hops to the Controller (3 hops for cluster 1, 2 for clusters 2 and 3, 1 for clusters 4 and 5).

Duty cycle is minimum for nodes in clusters 1 and 3, intermediate for clusters 2 and 5, maximum for cluster 4. According to the scheduling policy (Section 3.4.3) clusters 1 and 3 wake up for transmissions only once a TDMA-cycle, clusters 2 and 5 wake up twice for transmissions and once for reception and cluster 4 has three transmitting and two receiving TDMA-slots. This means that the duty cycle depends mainly on the number of times a cluster wakes up for the forwarding procedure. In particular, these results show that the major component of the duty cycle is the listening and receiving time. A node in a transmitting TDMA-slot of $3300ms$ is awake for about only $60ms$ ($\simeq$ 0.2% of a TDMA-cycle), while in the receiving TDMA-slot it is awake for an average time of $500ms$ ($\simeq$ 1.5% of a TDMA-cycle). It points out the benefits of power saving techniques in the receiving cluster. For instance, in the analyzed cases the number of nodes waking up for listening in the receiving cluster is equal to the expected number of packet to be transmitted in a TDMA-slot.

The Fig. 5.4 shows the evolution of the average delay, evaluated for the different distances in number of hops to the Controller. Nodes at the same distance to the Controller does not experience a big variance around the average values determined in Table 5.1, but the randomized approach make the nodes experience independent delays in two consecutive transmissions.

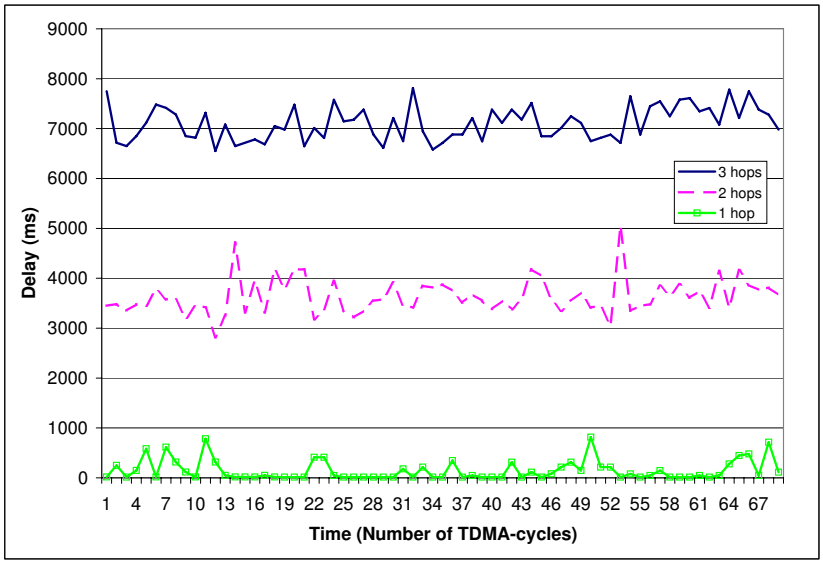

Figure 5.4: Average Delay, $\lambda = 1pkt/10s$

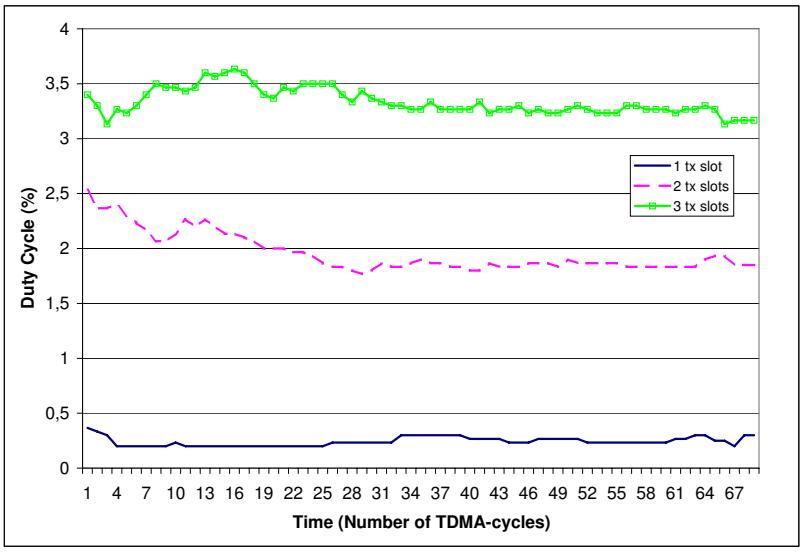

Figure 5.5: Average Duty Cycle, $\lambda = 1pkt/10s$

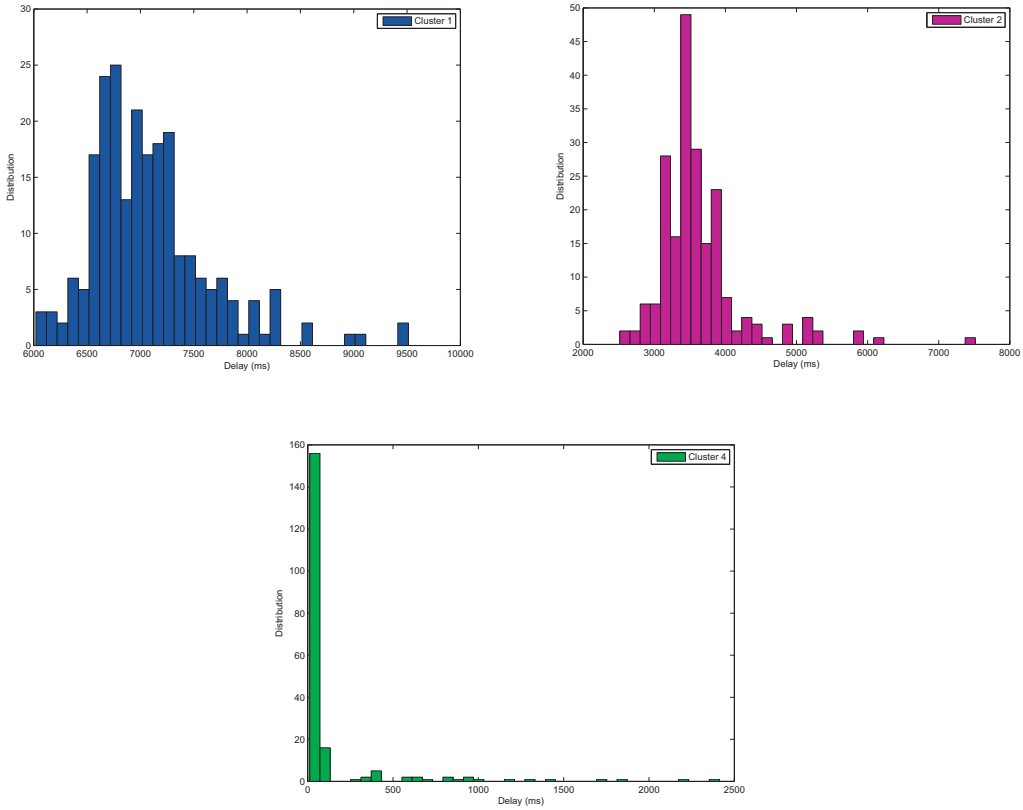

Figure 5.6: Delay distribution for clusters 1, 2 and 4 (3,2 and 1 hops to the Controller)

In Fig. 5.5 the average duty cycle is presented, considering averaged representative values for clusters with the same number of assigned TDMA-slot.

In Fig. 5.6, we show the distribution of the delays respectively for cluster 1, 2 and 4 (located at 3, 2 and 1 hop to the Controller). The furthest cluster presents a larger variance, due to the multi-hop.

This preliminary phase has been useful to verify the actual operation of the network and to have an idea of the behavior and the capabilities of the protocol.

5.3 Performance Analysis

In this Section, we present the results of the experiments with more stressed constraints, in order to evaluate the real performance and test the robustness of the protocol.

We considered a latency constraint on the end-to-end delay of: $D = 3s$ The lower bound for the cumulative PRR is fixed to: $PRR = 95\%$. Under this assumptions, the theoretical optimal value for the TDMA-slot duration is $S = 1100ms$. This value allows keeping the maximum delay in the worst case of communication within the required boundary. For this reason, it is fixed as start-point for the experiment (using off-line optimization).
The Controller initializes the network sending a token with a synchronization message and the TDMA-cycle schedule. It starts to acquire the actual values of delay, PRR and duty cycle. Furthermore, it is able to modify the TDMA-slot duration for an on-line optimization.
The slot duration is updated during the token refreshing procedure that, in this particular setup, is every 20 cycles. In Table 5.3, the experimental results are summarized, considering two different values of packet rate.

Table 5.2: Performance analysis: average values of PRR, delay and duty cycle

Cluster Rate λ	TDMA Slot S	Cluster Number	PRR	Average Delay	Duty Cycle
3pkt/10sec	1200ms	1	86.5 %	2553ms	0.5%
		2	94.6 %	1322ms	3.7%
		3	95.0 %	1355ms	0.5%
		4	97.3 %	43ms	6.1%
		5	97.0 %	38ms	3.6%
1pkt/5sec	1300ms	1	99.3 %	2719ms	0.4%
		2	99.7 %	1411ms	2.3%
		3	99.6 %	1401ms	0.4%
		4	100 %	66ms	4.1%
		5	100 %	59ms	2.3%

In both cases, the protocol increases the duration of the TDMA-slot due to the computed end-to-end delay values that do not reach the worst case

level in the normal execution. It enhances the performance of the network both in terms of PRR and duty cycle.
In the case $\lambda = 1pkt/5sec$, the optimal TDMA-slot is greater, and the latency constraint sometimes is not satisfied because of the better performance evaluated in terms of PRR. It allows accepting that some packets could be lost due to the end-to-end delay bound.

The Fig. 5.7 show the evolution of the average delay during the experiment, evaluated for the different distances in number of hops to the Controller. In Fig. 5.8 the average duty cycle is presented. As intuitively clear, the reference clusters for the end-to-end delay and PRR are the farthest from the Controller. However these clusters are also the less loaded in terms of traffic and have very low energy consumption. Otherwise the closest clusters present higher average node duty cycle, that is generated by the forwarding procedure.

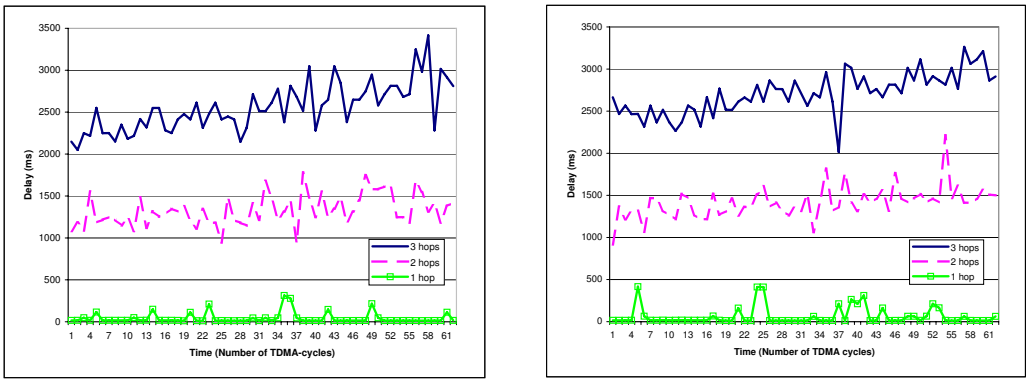

Figure 5.7: Average Delay, $\lambda = 3pkt/10s$ (left) $\lambda = 1pkt/5s$ (right)

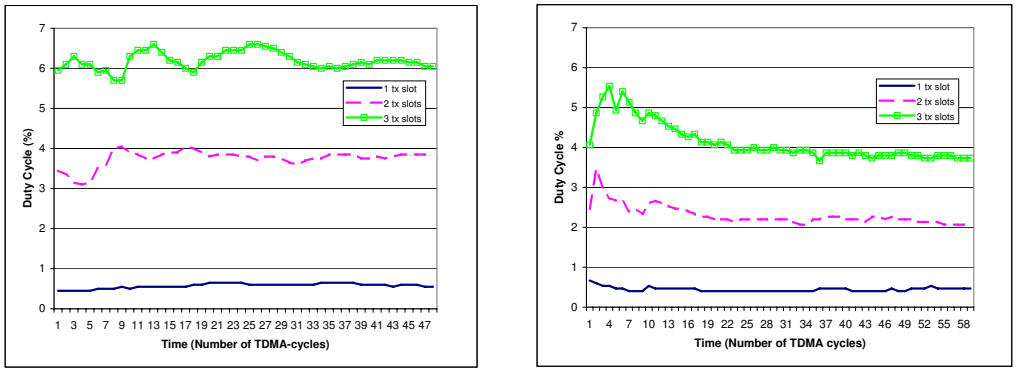

Figure 5.8: Average Duty Cycle, $\lambda = 3pkt/10s$ (left) $\lambda = 1pkt/5s$ (right)

5.4 Network Lifetime Estimation

We refer to data in Table 4.2 and the experimental results on the duty cycle to estimate the network lifetime. The cluster 4 has the highest duty cycle. Note that nodes in this cluster are the most loaded in terms of traffic to forward.

We focus on the experiment with $\lambda = 3pkt/10sec$ (Table 5.3). In Table 5.3, the time distribution of the different states during a TDMA-cycle ($\Delta = 10800ms$) is reported, with the relative energy consumption.

Table 5.3: Time distribution and energy consumption in a TDMA-cycle

State	Time	Percentage	Energy Consumption
Shutdown	10141ms	93.89%	1.2 μJ
Idle	646ms	5.98%	383 μJ
Transmission	6ms	0.05%	156 μJ
Reception	7ms	0.06%	203 μJ

Transmission and reception times include the time for transmitting and receiving packets and acknowledgements. Idle time is mainly composed of the time for listening and solving contentions.
The TDMA structure allows nodes to remain in shutdown state for an high percentage of time ($\simeq 94\%$), consuming a negligible amount of energy. Transmission and reception consumptions are balanced in order of magnitude, while the energy spent in idle state is slightly predominant. However, Fig. 5.9 shows that a node is in idle state for an average of 98% of the active time[2].

The total energy consumption in a TDMA-cycle is $E_\Delta = 743.2\mu J$, which corresponds to an average node power consumption of $P = 68.8\mu W$. Consequently, according to Equation 4.3 the estimated lifetime is:

$$T = \frac{2}{7414P + 0.1} \simeq 3.2\ years \tag{5.1}$$

For instance, in paper [28], the lifetime of a B-MAC protocol (Section 2.5.3) in indoor monitoring applications is estimated in 1.1 years.

[2]The active time is the sum of idle, transmission and reception time. It is related to the duty cycle

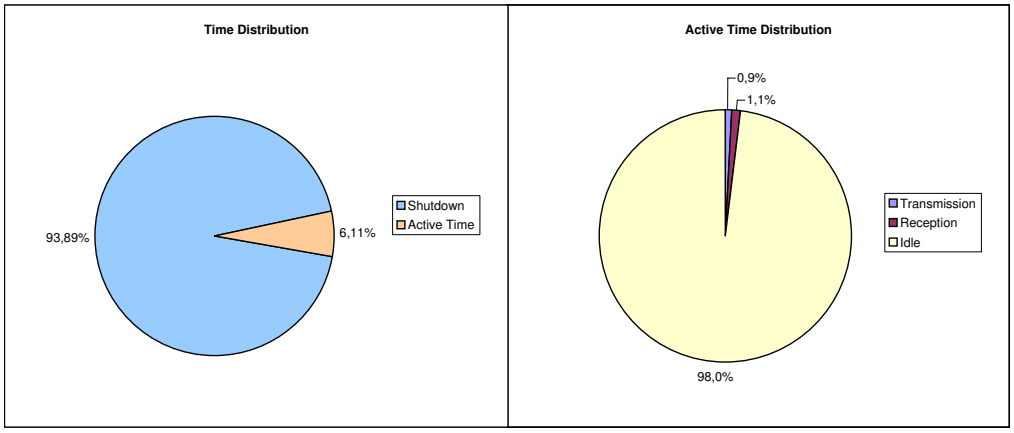

Figure 5.9: Time distribution

Chapter 6

Conclusions

6.1 Conclusions of the Work

The achieved objectives of this Thesis can be resumed, referring to different levels.

Regarding *theoretical aspects*, we provided a survey on the applications and the design of cross-layer protocols on WSNs. We focused the attention on SERAN, a semi-random protocol for clustered wireless sensor networks, proposed in its first formulation in paper [1]. SERAN satisfies system level requirements on packet delay and successful transmission probability while minimizing energy consumption. SERAN is characterized by a mathematical model that allows for cross-layer optimization without the need for extensive simulations.

We enhanced the optimization problem, including a mathematical analysis of the PRR. The main advantage is the possibility of an on-line optimization, which considers the actual dynamics in the network.

The major part of the work concerned the *implementation* of SERAN on Tmote Sky motes, using TinyOS.

The presented *experimental results* demonstrate the effectiveness of the implementation. The protocol allows the network to get excellent performance for low data rate transmissions, ensuring low average node duty cycle, which yields a long network lifetime.

6.2 Future Developments

An ongoing project is the definition of a GUI interface for SERAN, that allows showing the behavior of the network with different levels of abstraction.

Future work includes performing a set of experiments with a wide experimental evaluation of the SERAN protocol, in several scenarios of traffic load, node clustering, cluster size, channel conditions, and performance requirements.

An improvable part of the protocol is the acknowledgement-based contention scheme. We noticed it gives physical limits to the CSMA-slot duration. Different mechanisms, as the use of Beacon messages in paper [6], can be exploited, comparing SERAN with other solutions of cross-layer protocol. Also the interaction with the standard IEEE 802.15.4 [31] might be analyzed, both in the mathematical formulation and in the practical implementation.

Furthermore, aggregation algorithms can be included in the framework to increase the effective throughput at no power cost, whenever aggregate information is required.

Appendix A

Protocol Parameters

	Topology Parameters
P	Number of paths
B_i	Number of clusters in the path i
N	Number of nodes per cluster
	Channel Parameters
c	Channel gain
	Application Parameters
λ	Packet generation rate
$PRR_{\min}$	Minimum Packet Reception Rate
$D_{\max}$	Maximum end-to-end Delay
	MAC/Routing Parameters
p	Access probability
S	TDMA-slot duration
T_f	Number of TDMA-slots in a TDMA-cycle
δ	CSMA-slot duration
Δ	TDMA-cycle duration
k	Number of packets to send in the cluster
τ_k	Expected time to evacuate a cluster

	Physical Layer Parameters
E_{pkt}	Energy spent for a single data transmission
E_{ack}	Energy spent for a single ACK transmission
R	Energy spent for a single wake up
W	Energy (per unit time) spent during the listening
N_{pkt}	Average number of data transmissions
N_{ack}	Average number of ACK transmissions
N_{wu}	Total number of wake-ups

	TinyOS Parameters	**Value**
$DATA\_CHANNEL$	Data communication channel	15
$ACK\_CHANNEL$	Acknowledgement channel	14
$TOKEN\_CHANNEL$	Synchronization channel	15
$MESS\_LENGTH$	Length of data message	28 bytes
ACK_LENGTH	Length of ACK	24 bytes
$TOKEN\_LENGTH$	Length of synchronization message	28 bytes

Appendix B

Packet Structure

SERAN Payload

typedef nx_struct DataMsg {
nx_uint16_t seqnum; *Data Sequence Number*
nx_uint8_t sourceid; *Software ID of the source node*
nx_uint8_t forwardid; *Software ID of the node that forwards the packet*
nx_uint8_t destgroupid; *Software ID of the destination node*
nx_uint8_t trafficrate; *Traffic rate in pkt/s*
nx_uint8_t TDMAslotdur; *TDMA-slot (in number of CSMA-slots)*
nx_uint8_t slots; *CSMA-slot number*
nx_uint16_t dutycycle; *Evaluated node duty cycle*
nx_uint16_t delay; *Time-stamp for the evaluation of delays*
} DataMsg;

typedef nx_struct AckMsg {
nx_uint16_t seqnum; *Data Sequence Number*
nx_uint8_t sourceid; *Software ID of the ACK source node*
nx_uint8_t forwardid; *Software ID of the node that forwarded the data*
nx_uint8_t destgroupid; *Software ID of the data source node*
nx_uint8_t prr; *Evaluated PRR (from the sink node)*
nx_uint8_t TDMAslotdur; *TDMA-slot (in number of CSMA-slots)*
nx_uint8_t slots; *CSMA-slot number*
} ACKMsg;

```
typedef nx_struct SyncMsg {
    nx_uint16_t seqnum; Data Sequence Number
    nx_uint8_t sourceid; Software ID of the source node
    nx_uint8_t forwardid; Software ID of the node that forwards the token
    nx_uint8_t destgroupid; Software ID of the destination node
    nx_uint8_t trafficrate; Traffic rate in pkt/s
    nx_uint8_t TDMAslotdur; TDMA-slot (in number of CSMA-slots)
    nx_uint8_t slots; free
    nx_uint16_t dutycycle; free
    nx_uint16_t delay; Synchronization time
  } SyncMsg;
```

Bibliography

[1] A. Bonivento, C. Fischione, A. Sangiovanni-Vincentelli, F. Graziosi, F. Santucci: *SERAN: A Semi Random Protocol Solution for Clustered Wireless Sensor Networks*, MASS '05, November 2005.

[2] A. Bonivento, C. Fischione, F. Pianegiani, A. Sangiovanni-Vincentelli: *System Level Design for Clustered Wireless Sensor Networks*, IEEE Transactions on Industrial Informatics, Vol. 3, No. 3, August 2007

[3] A. Bonivento, C. Fischione, A. Sangiovanni-Vincentelli: *Randomized Protocol Stack for Ubiquitous Networks in Indoor Environment*. Proceedings of Consumer Communication and Network Conference (CCNC), Las Vegas, January 2006.

[4] A. Bonivento: *Platform Based Design for Wireless Sensor Networks*. PhD Thesis, UC Berkeley, 2007.

[5] P.G. Park: *Protocol Design of Sensor Networks for Wireless Automation*. MSc Thesis, KTH Stockholm, 2007.

[6] P.G. Park, C. Fischione, A. Bonivento, K.H. Johansson, A. Sangiovanni-Vincentelli: *Breath: a Self-Adapting Protocol for Wireless Sensor Networks*. Accepted paper at IEEE SECON 2008, San Francisco US, June 2008.

[7] D. Culler, D. Estrin, M. Srivastava: *Overview of Sensor Networks*, IEEE Computer Society, August 2004.

[8] L. Pomante: *Wireless Sensor Networks*, Seminar in Wireless Communications - University of L'Aquila, March 2007.

[9] J.N. Al-Karari, A.E. Kamal: *Routing Techniques in Wireless Sensor Networks: A Survey*, IEEE Wireless Communications, December 2004.

[10] W. Heinzelman et al.: *Energy-efficient Communication Protocol for Wireless Microsensor Networks*, Proc. 33rd Hawaii Int'l Conf.Sys.Sci., January 2000.

[11] *nesC: A Programming Language for Deeply Networked Systems*, UC Berkeley WEBS Project, December 2004.

[12] C.Y. Chong: *Sensor Networks: Evolution, Opportunities, and Challenges*, Proceedings of IEEE Vol 91, No 8, August 2003.

[13] I.F. Akyildiz , W. Su , Y. Sankarasubramaniam , E. Cayirci: *Wireless sensor networks: a survey*, Computer Networks: The International Journal of Computer and Telecommunications Networking, v.38 n.4, p.393-422, 15 March 2002

[14] A. Sangiovanni-Vincentelli, M. Sgroi, A.Wolisz and J. M. Rabaey: *A service-based universal application interface for ad-hoc wireless sensor networks*, Whitepaper, U.C.Berkeley, 2004.

[15] W. Ye and J. Heidemann: *Medium access control in wireless sensor networks*, Norwell, MA, USA: Kluwer Academic Publishers, pp. 73 – 91, 2004.

[16] R. Rom, M. Sidi: *Multiple Access Protocols - Performance and analysis*, Springer-Verlag, New York, 1990.

[17] J. Heidemann W. Ye and D. Estrin: *Medium access control with coordinated adaptive sleeping for wireless sensor networks*, IEEE/ACM Transactions on Networking, 12, n. 3 pp. 493-506, 2004.

[18] T.V. Dam and K. Langendoen: *An adaptive energy-efficient MAC protocol for Wireless Sensor Networks*, in SenSys '03. New York, NY, USA: ACM Press, pp. 171–180, 2003.

[19] J. Hill, J. Polastre and D. Culler: *Versatile low power media access for wireless sensor networks*, Proceedings of SenSys 2003.

[20] I. Rhee, A. Warrier, M. Aia and J. Min: *ZMAC: a Hybrid MAC for Wireless Sensor Networks*, New York, NY, USA: ACM Press, 2005, pp. 90 – 101.

[21] *IEEE 802.11 – Wireless LAN media access control (MAC) and physical layer (PHY) specifications*. 1999

[22] T. Melodia, M.C. Vuran, and D. Pompili: *The State of the Art in Cross-Layer Design for Wireless Sensor Networks*, Broadband and Wireless Networking Laboratory, School of Electrical & Computer Engineering, Georgia Institute of Technology, Atlanta, GA.

[23] X. Liu, A. GoldSmith: *Wireless Network Design for Distributed Control*, IEEE Conference on Decision and Control, Atlantis, December 2004.

[24] D. Gay, P. Levis, and D. Culler: *Software design patterns for tinyos*, in Proc LCTES '05, vol. 31, no. 15. New York, NY, USA: ACM Press, 2005, pp. 40–49.

[25] B. Bougard, F. Catthoor, D.C. Daly, A. Chandrakasan,W. Dehaene: *Energy Efficiency of the IEEE* 802.15.4 *Standard in Dense Wireless Microsensor Networks:Modeling and Improvement Perspectives*, IMEC, Leuven, Belgium, March 2005.

[26] El-Hoiydi: *A.Spatial TDMA and CSMA with preamble sampling for low power ad hoc wireless sensor networks*, Proceedings, International Symposium on Computers and Communications, ISCC, 2002.

[27] El-Hoiydi: *On the Lifetime of Wireless Sensor Networks*, IEEE Communications Letters, Vol. 9, No. 11, November 2005.

[28] B. Lazzerini, F. Marcelloni, M. Vecchio, S. Croce, E. Monaldi: *A Fuzzy Approach to Data Aggregation to Reduce Power Consumption in Wireless Sensor Networks*, Annual meeting of the North American Fuzzy Information Processing Society, 2006. NAFIPS 2006.

Web References

[29] *Tmote Sky Data Sheet*, Moteiv, San Francisco, CA, 2006.
http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf

[30] *The TinyOS community forum*.
http://www.tinyos.net

[31] *IEEE 802.15.4 Specification*.
http://standards.ieee.org/getieee802/802.15.html

[32] *CC2420 Data Sheet*, Chipcon, Oslo, Norway, 2005.
http://www.chipcon.com/files/CC2420-Data-Sheet-1-3.pdf

[33] P. Levis: *TinyOS Programming*.
http://csl.stanford.edu/ pal/pubs/tinyos-programming.pdf