Technische Universität Dresden
Faculty of Computer Science
Institute of Systems Architecture
Chair of Computer Networks

Automatic Parameter Optimization of Sensor Network MAC Protocols

Diploma Thesis of
Marco Zimmerling

Advisors:
Dr.-Ing. W. Dargie (TU Dresden)
Dr. T. Voigt (Swedish Institute of Computer Science)

Supervisor:
Prof. Dr. rer. nat. habil. Dr. h. c. A. Schill (TU Dresden)

Uppsala, August 2009

To Berit

Abstract

Wireless sensor networks are concerned with resource and performance constraints. In the light of these constraints, sensor network protocols must operate efficiently and effectively at all times. This applies in particular to the medium access control (MAC) protocol, which dictates to a large extent the energy consumption of a sensor node by controlling the use of the radio transceiver. Moreover, it determines the timing of communication among neighboring nodes and thus per-hop latency and per-hop reliability.

To achieve the best possible performance, the MAC protocol must be configured with appropriate parameters and automatically adapted to changes in network conditions and application requirements.

This thesis presents a novel system for automatic optimization and adaptation of sensor network MAC protocols. It adapts the protocol parameters at runtime to ensure optimal performance and compliance with the application requirements. The optimization approach is based on constraint programming and optimizes multiple objectives simultaneously. The approach is applied in a case study to X-MAC [9], a flexible and widely used MAC protocol for sensor networks. Experimental results obtained from a small-scale network of real sensor nodes show that the proposed system keeps MAC protocol performance close to the optimum under varying network conditions and different application requirements.

Acknowledgments

I am grateful to many people who helped me accomplish this thesis. Prof. Alexander Schill and Waltenegus Dargie gave me the freedom to pursue my thesis project in Sweden on a topic that I consider fascinating, and Prof. Per Gunningberg gave me the chance to join the Communications Research Group at Uppsala University for seven months. Christian Rohner and Lars-Åke Larzon helped me to get started and narrow the scope of my initial ambitions. I thank Prof. Pierre Flener for his encouragement to seriously look into constraint programming and helpful discussions in the early stages of this thesis.

If it had not been for Thiemo Voigt and Luca Mottola, this thesis would not have attained its present standard. I thank you for all your help developing the ideas in this thesis and for showing me how to present my work more clearly. I always gained fresh inspiration from our meetings at SICS, even if you were just listening.

During the writing of this thesis I received thorough input and valuable comments from Luca, Michael, Thiemo, Tino, and Ulf.

Throughout the entire thesis project, my family supported me whenever I was close to despair and put me in mind of the many other exciting things besides sensors.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter explains why it is desirable to automatically optimize and adapt the parameters of a sensor network MAC protocol at runtime, lists the contributions of this thesis, and presents the outline of the report.

1.1 Problem Statement

Wireless sensor networks are distributed collections of tiny devices. Each device is equipped with a microprocessor to execute small computer programs, a wireless radio transceiver to exchange messages with other devices, and several sensors to perceive the environment. Applications of sensor networks range from passive data gathering for habitat monitoring [59] to active control for home automation [66].

Sensor networks are confronted with resource and performance constraints. The nodes are powered by batteries, which presents a severe energy constraint for long-running applications if recharging or replacing of batteries is infeasible. A typical microprocessor is limited to a few megahertz processing speed, external flash memory provides roughly a megabyte data storage, and wireless communication speed is on the order of a few hundred kilobits per second. In the light of these constraints, sensor network protocols need to operate efficiently and effectively at all times.

The MAC protocol allows nodes to communicate over the same wireless channel by controlling the use of the radio transceiver. Energy-efficiency is the main concern of a sensor network MAC protocol, because the radio is the most power-consuming component of a typical sensor node [68]. Therefore, the radio should be turned off whenever possible to reduce the amount of energy wasted by idle listening and overhearing. At the same time, the radio needs to be turned on frequently enough at the right times to ensure low latency and high reliability of neighbor-to-neighbor communication. To achieve the best possible performance, the MAC protocol must be configured with optimal parameters.

However, it is not sufficient to configure the MAC protocol once and for all. Sensor networks operate and must respond to very dynamic environments [27]. They suffer substantial changes as nodes fail due to battery exhaustion or accident, nodes move or new nodes are added. User preferences also contribute to the dynamics as what is considered interesting changes. In an extreme case some nodes may be reprogrammed to adjust the sensing task of the network. Even in the absence of these external influences the conditions in the sensor network vary greatly. For example, Zhao and Govindan [99] observe 40 % variability in packet reception rate within a two-hour time window. Thus, the parameters of the MAC protocol

must be automatically adapted to changes in network conditions and application requirements.

1.2 Thesis Contributions

This thesis presents a novel system for automatic parameter optimization and runtime adaptation of sensor network MAC protocols. To this end, our contributions unfold as follows:

- We design an optimization approach based on constraint programming that enables simultaneous optimization of multiple conflicting performance metrics of sensor network MAC protocols.
- We implement our approach in the ECL^{i}PS^e constraint programming system using an analytical model of X-MAC—a flexible and widely used MAC protocol for wireless sensor networks—as a paradigmatic case study.
- We demonstrate the effectiveness of our approach by exploring the performance trade-offs in X-MAC, optimizing and adapting X-MAC to varying network conditions and application requirements, and quantitatively evaluating the accuracy of our analytical model.

1.3 Thesis Outline

The remainder of this thesis proceeds as follows:

Chapter 2 gives more background on parameter optimization in sensor networks and briefly reviews existing approaches. It introduces the basic concepts of multiobjective optimization and constraint programming, and discusses some works that apply constraint programming to problems in mobile ad-hoc and sensor networks. Chapter 2 also presents a taxonomy for sensor network MAC protocols and discusses prior work on MAC protocol optimization and adaptation.

Chapter 3 describes our novel system for automatic parameter optimization and runtime adaptation of sensor network MAC protocols. Furthermore, it presents within a case study the implemention of our optimization approach in the ECL^{i}PS^e constraint programming system using an analytical model of X-MAC.

Chapter 4 evaluates the proposed system. It validates the accuracy of the analytical model presented in Chapter 3 and explores the performance trade-offs in X-MAC using methods from multiobjective optimization. Experiments on a small-scale network of real sensor nodes are performed to show that the optimization approach determines optimal protocol parameters for different application requirements and characteristics, and to demonstrate its capability to adapt the protocol parameters to dynamic changes in network properties.

Chapter 5 presents conclusions and discusses possible future research directions.

Chapter 2

Background and Related Work

This chapter starts with an overview of existing approaches for parameter optimization in sensor networks. It gives a brief introduction to multiobjective optimization and constraint programming as the concepts developed in this thesis build upon established methods from both areas. Finally, it describes the different classes of a taxonomy for sensor network MAC protocols and discusses prior work on MAC protocol optimization and adaptation.

2.1 Parameter Optimization in Sensor Networks

Stringent resource constraints force us to optimize the operation of wireless sensor networks. Because the protocols deployed on individual nodes control the operation of the whole sensor network, we first have to design efficient protocols and then determine optimal parameters for these protocols. Important parameters are, for example, the fractions of packets routed to each neighbor, the transmit power levels of different nodes, and the lengths of listen and sleep period. He *et al.* [39] argue that we should yet design new protocols with ease of parameter optimization in mind.

We discuss mathematical optimization and machine learning, the most widely used methods for parameter optimization in sensor networks, and review some concrete applications of these methods reported in the literature.

2.1.1 Mathematical Optimization

Parameter optimization problems in sensor networks are typically *mathematical programming* problems. Most existing work aims at a *linear programming* formulation [7, 12, 36, 57, 97], mainly because there are a variety of very effective methods for solving them, including Dantzig's simplex method. An optimization problem is called a linear program if the objective and constraint functions are linear. Some resort to the more general class of *convex optimization* problems [8] if linearity is too difficult or impossible to attain [9, 62]. In this case, the objective and constraint functions are convex, and interiorpoint methods can be used to solve these problems efficiently. For many optimization problems it is possible to construct the equivalent Lagrange dual problem, which is a convex optimization problem whether or not the original problem is convex [8]. The Lagrange dual problem is particularly appealing for sensor networks, because it can be solved in a distributed way using the subgradient algorithm.

Linear programming has been used intensively to optimize routing decisions. For example, Chang and Tassiulas [12] as well as Madan and Lall [57] propose

distributed algorithms to compute an optimal routing scheme that maximizes the network lifetime. They define network lifetime as the time until the first node drains out of energy. The goal is then to find a maximum flow of packets from the source nodes to the base station, subject to the energy constraint of limited battery power at each node. While [12] and [57] assume that packets can be split into fractional parts at intermediate relay nodes, Bodlaender *et al.* [7] note that this may not be desirable nor practical and consider packets as units that cannot be split. The resulting *integer linear programming* problem is strongly NP-complete. Besides minimizing the energy consumption during data collection, Yuen *et al.* [97] consider the second objective of finding an optimal rate allocation. The idea is that nodes should transmit at a certain rate to aggregate redundant or correlated data along routing paths as much as possible. Ha *et al.* [36] provide an integer linear programming formulation to determine subtrees of nodes that take turns in sleeping. The goal is to wake up only a subset of nodes in a routing tree to minimize overhearing and packet collisions during data collection, which saves energy and increases reliability.

2.1.2 Machine Learning

A recent research direction is the application of *machine learning* techniques in sensor networks [31]. The goal of machine learning algorithms is to automatically learn the properties of the environment and to adapt their behavior accordingly. In particular, *reinforcement learning* is well suited because it has medium requirements for memory and computation, is easy to implement, and highly flexible to topology changes. More importantly, it converges in acceptable time to the optimum [31].

Using reinforcement learning, a sensor node (or agent) actively explores its environment by taking some possible action and receiving a reward from the environment [73]. The reward tells the agent whether something good or bad has happened as the result of taking that action. By trying many different actions and sequences of actions, the agent learns from its experience. The acquired knowledge is represented by the so-called *value function*. The value function defines the expected total reward when taking an action in some state, assuming that the agent behaves optimally from the next state on.

Q-learning [90] is a simple but powerful reinforcement learning algorithm. It learns the value function in an online fashion without needing a model of the environment [73]. Q-learning has been used for routing in sensor networks. For example, FROMS [32, 33] uses Q-learning to find routing paths that minimize hop count or energy and outperforms Directed Diffusion [43] on real sensor hardware.

2.2 Multiobjective Optimization

Most problems in nature involve several, possibly conflicting, objectives. A bee colony, for example, tries to satisfy conflicting objectives simultaneously [75]. Those objectives include maximizing the amount of collected honey, maintaining the temperature in a nest, and minimizing the number of dead drones. If bees focus only on foraging, they fail to ventilate their nest and remove dead drones. Similarly, applications of wireless sensor networks tend to have conflicting operational objectives. In data collection applications, for example, data delivery rate and latency conflict with each other. A higher data delivery rate allows human

operators to make better informed decision; a lower latency allows human operators to make more timely decisions. To improve data delivery rate, hop-by-hop recovery is often applied, but this degrades latency. In both nature and engineering we find so-called *multiobjective optimization problems*.

2.2.1 Concepts

According to Osyczka [64], multiobjective optimization is the problem of finding a vector of *decision variables* that satisfies constraints and optimizes a vector function whose elements represent the *objective functions*. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term *optimize* means finding such a solution that would give the values of all the objective functions acceptable to the decision maker.

Formally, we seek a vector $\mathbf{x}^*$ that satisfies the constraints

$$g_i(\mathbf{x}) \geq 0 \qquad (i = 1, 2, \ldots, r) \tag{2.1}$$

$$h_i(\mathbf{x}) = 0 \qquad (i = 1, 2, \ldots, s) \tag{2.2}$$

and optimizes the vector function

$$f(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})]^T, \tag{2.3}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables. In other words, we wish to determine from the set $\mathcal{F}$ of all numbers that satisfy (2.1) and (2.2) the particular numbers $x_1^*, x_2^*, \ldots, x_k^*$ that yield optimal values of all objective functions. Set $\mathcal{F}$ is called the *feasible region* of the problem.

Having several objective functions, we essentially try to find good compromises between the objectives rather than a single solution. This idea is captured by the concept of *Pareto optimality*. We say that a vector of decision variables is Pareto optimal if there exists no feasible vector that decreases some objective without causing a simultaneous increase in at least one other objective. Formally, $\mathbf{x}^* \in \mathcal{F}$ is Pareto optimal if there exists no $\mathbf{x} \in \mathcal{F}$ such that $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ for all $i = 1, \ldots, k$ and $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$ for at least one j. The concept of Pareto optimality gives usually a set of solutions, that is, several Pareto optimal vectors $\mathbf{x}^*$ of decision variables.[1] Plotting these solutions yields the *Pareto front*, where the point corresponding to solution $\mathbf{x}^*$ is given by $f(\mathbf{x}^*)$. The shape of the Pareto front provides valuable information about the trade-offs between the objectives.

Figure 2.1 depicts the Pareto front as it may look for a data collection problem with the two conflicting objectives data delivery rate and latency. To achieve a high data delivery rate, we have to accept a longer latency. The optimal solutions, or rather the good trade-offs, lie on the Pareto font. In general, it is difficult to find an analytic expression of the line or surface that contains these solutions.

2.2.2 Solving Methods

There exists a range of approaches for solving multiobjective optimization problems [80, 87]. In this thesis, we use two approaches that combine the different objectives into a single function. For small-scale problems with a limited number of

[1] The solution vectors $\mathbf{x}^*$ are *non-dominated*. Vector $\mathbf{u} = (u_1, \ldots, u_k)^T$ is said to dominate vector $\mathbf{v} = (v_1, \ldots, v_k)^T$ if and only if $\mathbf{u}$ is partially less than $\mathbf{v}$, that is, $\forall i \in \{1, \ldots, k\}$: $u_i \leq v_i$ and $\exists i \in \{1, \ldots, k\}$: $u_i < v_i$.

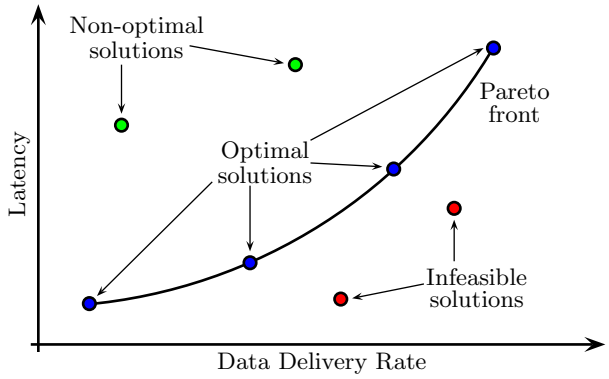


Figure 2.1: Pareto front of an optimization problem with two objectives.

objectives and decision variables, the combination of objectives is one of the simplest but also one of the most efficient methods [17]. For large-scale problems, approaches based on evolutionary algorithms are superior, because they are able to find an entire set of Pareto optimal solutions in a single run [103].

Weighted Sum Approach

This method adds all objective functions together using different weighting coefficients for each of them. The resulting optimization problem is of the form

$$\min \sum_{i=1}^{k} w_i f_i(\mathbf{x}), \qquad \text{subject to } \mathbf{x} \in \mathcal{F}, \tag{2.4}$$

where the weights $w_i \geq 0$ represent the relative importance of the objectives, and $\mathcal{F}$ is the feasible region. It is usually assumed that $\sum_{i=1}^{k} w_i = 1$. If we want the weights to proportionally reflect the importance of the objectives, all functions should be expressed in units of approximately the same numerical values. Therefore, we transform (2.4) into

$$\min \sum_{i=1}^{k} w_i f_i(\mathbf{x}) c_i, \qquad \text{subject to } \mathbf{x} \in \mathcal{F}, \tag{2.5}$$

where the constant multipliers c_i properly scale the objectives. We set $c_i = 1/f_i^0$, where f_i^0 is the solution of the optimization problem with f_i as the single objective.

The main strength of this method is its computational efficiency. Moreover, it is simple and easy to implement. However. the approach misses concave portions of the Pareto front [72].

Goal Programming

Using this method, we have to assign goals that we wish to achieve for each objective. The goals are incorporated into the optimization problem as additional constraints. The objective function minimizes the absolute deviations from the goals to the objectives. The optimization problem is of the form

$$\min \sum_{i=1}^{k} |f_i(\mathbf{x}) - G_i|, \qquad \text{subject to } \mathbf{x} \in \mathcal{F}, \tag{2.6}$$

where G_i denotes the goal set for the ith objective function $f_i(\mathbf{x})$.

If we know the desired goals and these are in the feasible region, this method yields a dominated solution in a computationally efficient way.

2.3 Constraint Programming

Constraints pervade many areas of human endeavor. They formalize dependencies in the physical world. More precisely, a *constraint* is a logical relation among several variables, each taking a value in a given domain. Constraints thus restrict the possible values that variables can take. Constraints are declarative, that is, they specify which relationships must hold without specifying the computational procedure to enforce those relationships [5].

Constraint programming is a programming paradigm in which a problem is modeled as a set of constraints, and a solution of the problem is found using general or domain specific methods [1]. It is used with great success for solving combinatorial and numerical problems in areas such as planning and scheduling [89]. We discuss some approaches that apply constraint programming to problems in wireless ad-hoc and sensor networks after introducing the basic concepts and some of the general solving methods.

2.3.1 Concepts

We distinguish between *constraint satisfaction problems (CSPs)* and *constraint optimization problems (COPs)*. By a CSP, we mean a finite sequence of variables, each ranging over a possibly different domain, and a finite set of constraints, each on a subsequence of the considered variables. The variables used in a CSP are called *decision variables*. A solution to a CSP is an assignment of values to its decision variables such that all constraints are satisfied. A *feasible* CSP has one or more solutions; an *infeasible* CSP has no solution. A COP is essentially a CSP together with an objective function, which is a mapping from the decision variables to the set of real numbers. The goal is then to find an optimal solution, that is, an assignment of values to the decision variables such that all constraints are satisfied *and* the value of the objective function is minimal (or maximal).

Constraint programming languages provide substantial support for modeling CSPs and COPs. This includes built-in facilities to represent the decision variables and their domains, as well as to generate constraints. Figure 2.2 shows how CSPs and COPs are modeled in ECL^{i}PS^e [88], a constraint programming language based on Prolog. The goal of the CSP is to assign integer values (`1` or `2`) to the decision variables (`X` and `Y`) such that the sum of the variables is less than `4`. The `labeling` predicate determines three solutions: `[X=1,Y=1]`, `[X=1,Y=2]`, and `[X=2,Y=1]`. The COP goes one step further and seeks to find an assignment that minimizes the sum of the variables. The `minimize` predicate determines the optimal solution: `[X=1,Y=1]`. This simple example demonstrates the specification of decision variables, finite domains, and arithmetic equality and inequality constraints. The underlying solving mechanisms encoded by the predicates `labeling` and `minimize` are briefly discussed in the following section.

```
/* Constraint satisfaction problem */
csp(X,Y,Sum) :-
    X :: [1..2], Y :: [1..2],
    Sum #= X + Y, Sum #< 4,
    labeling([X,Y]).

/* Constraint optimization problem */
cop(X,Y,Sum) :-
    X :: [1..2], Y :: [1..2],
    Sum #= X + Y, Sum #< 4,
    minimize(labeling([X,Y]),Sum).
```

Figure 2.2: Constraint satisfaction and constraint optimization.

2.3.2 General Solving Methods

General methods to solve problems formulated as CSPs or COPs are based on search. *Local search* starts with a random assignment of all variables and tries to improve the initial assignment iteratively by small, local changes. The quality of an assignment is measured by a cost function, for example, the number of violated constraints. Local search is, in general, incomplete because the final assignment is either a solution or indicates only that no solution has been found so far.

The solving methods used in this thesis are based on *backtracking search*, a form of complete top-down search that explores all possible assignments. Top-down search takes place on a tree, where the leaves of the tree are CSPs that are either infeasible or solved. Backtracking search, in particular, starts at the root of the tree and traverses downwards as long as a node is not a leaf. If a leaf is encountered, the search proceeds by moving back to the closest ancestor that has another descendant. This process continues until all descendants of the root have been visited.

Backtracking search is combined with *branching* and *constraint propagation*. Branching splits a given CSP into two or more CSPs, the union of which is equivalent to the initial CSP. An example of branching is splitting the domain of a variable in two halves by choosing an appropriate mid-point. Which variable domain is splitted is determined by a heuristic. In this thesis, the variable with the smallest domain size is selected. Constraint propagation transforms a given CSP into one that is equivalent but simpler by removing values from the domains that do not participate in any solution [49]. For example, the constraint $x < y$, where the domain of x is [10..100] and the domain of y is [1..50], allows constraint propagation to reduce the domain of x to [10..49] and the domain of y to [11..50]. Constraint propagation is alternated with branching.

For solving COPs, a modification of backtracking search, namely *branch and bound search*, is used. In case of minimization, the search maintains the currently minimal value of the objective function as the current *bound*. The bound is initialized to positive infinity and updated each time a solution with a smaller value of the objective function is found. At the same time, an appropriate inequality constraint on the objective function is maintained that triggers pruning of the search tree by identifying nodes under which no better solution can be present. Implementations

of branch and bound search are typically parameterized, allowing the programmer to control the improvement of subsequent solutions. This essentially abandons completeness—the solution returned is only guaranteed to be within a given bound from the true optimum—but speeds up the search.

2.3.3 Application in Wireless Networks

Frühwirth and Brisset [30] use constraint solving to compute the minimal number of base stations and their locations. Given a construction plan of the building and information about the materials used for walls and ceilings, the propagation of radio waves is simulated to compute so-called radio cells. A radio cell is the space where a base station must be placed to cover a possible receiver position. After computing an initial solution, the number of base stations is minimized using a branch and bound method. The system is implemented in ECL^{i}PS^e and produces placements comparable to those of a human expert.

Kotecha *et al.* [48] use constraint programming for computing the optimal placement of sensors based on different criteria such as cost and reliability. They solve this multiobjective optimization problem with lexicographic optimization. In this approach, the objectives are given a precedence ordering, and the optimization is performed by solving a series of optimization problems. At each step, the optimization of one objective is considered, and a new equality constraint is added to ensure that the optimal values of the previous objectives are maintained.

Guettier *et al.* [35] propose an approach to sensor network management that combines constraint solving techniques and distributed agreement algorithms. The goal is to find a subnet of active sensor routers such that the quality of the collected sensor data is maximized. Each sensor proposes a solution using a constraint solving algorithm. Afterwards, global agreement is reached by distributed consensus and coordination. The authors use ECL^{i}PS^e to implement the problem model and the search algorithms.

Frei and Faltings [29] formulate the problem of bandwidth allocation in communication networks as a CSP. Using information about available link capacities and expected traffic profile, their system computes offline an allocation of known demands to communication links such that the bandwidth requirements of the demands are satisfied. The focus of this work is on finding a compact representation of the problem to enable effective application of various CSP techniques.

2.4 Medium Access Control in Sensor Networks

The medium access control (MAC) protocol allows several sensor nodes to communicate with each other over the same wireless channel. It controls when and for how long a node may use its radio for transmitting or receiving data. Given that the radio is the most power-consuming component of a typical sensor node [68], a sensor network MAC protocol must turn off the radio as often as possible to conserve energy. At the same time, nodes must turn on their radios long enough to ensure responsiveness and network connectivity. Some MAC protocol designs abandon energy efficiency as the primary objective and instead focus on reliable data delivery with hard latency constraints [16, 62].

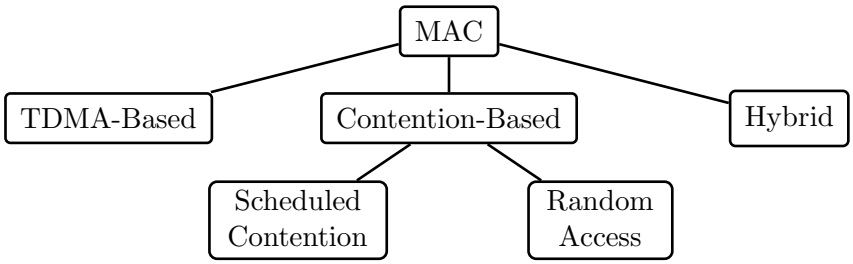


Figure 2.3: Taxonomy of sensor network MAC protocols.

2.4.1 Taxonomy

A multitude of MAC protocols for sensor networks have been developed [18, 50]. As shown in Figure 2.3, existing approaches can be classified as TDMA-based or contention-based. Two classes of contention-based protocols are those that add schedules and those that access the channel randomly. Hybrid protocols have been proposed aiming at combining the benefits of random access (flexible) and TDMA (collision-free). All protocols control the radio duty cycle to reduce energy waste due to idle listening; a few also take approaches to avoid collisions and overhearing. We discuss each class in turn in the following, but limit ourselves to single-channel MAC protocols.[2]

TDMA-Based Protocols

TDMA-based protocols schedule channel access in detail. Time is divided into frames, and in each frame either senders (sender-based scheduling) or communication links (link-based scheduling) are allocated in separate slots. Since nodes know exactly when they are supposed to send or listen, overhearing is mostly avoided and idle listening is reduced to a simple check whether the slot is actually used.

Early approaches require centralized control and global time synchronization. In LEACH [40] and BMA [55], for example, nodes form clusters and cluster heads schedule transmissions. To extend flexibility, TRAMA [69] and LMAC [85] contain a distributed slot assignment mechanism that allows nodes to select a slot based on information about their two-hop neighbors. Dozer [10] circumvents the burden of network-wide time synchronization by constructing single-hop schedules. Nodes form a routing tree and maintain two schedules: one provided by their parent and one self-determined for communication with their children.

Scheduled Contention

Scheduled, contention-based protocols organize time into slots. Nodes wake up at the beginning of each slot to synchronize with their neighbors and to exchange pending messages. As a result, communication is grouped into a small active period, which increases the probability of collisions. However, this approach allows the nodes to sleep for most of the time within a slot and greatly reduces idle listening.

The original S-MAC [94] protocol synchronizes the wake-up schedule of nodes with a fixed active period; a later refinement [95] varies the length of the active

[2] Recently, MAC protocols have been proposed that exploit multiple channels available at modern radio transceivers [47, 51]. The CC2420 radio, for example, features 16 different channels. This may alleviate some issues commonly encountered in single-channel solutions.

period to reduce multi-hop latency. T-MAC [86] adapts the active period to traffic fluctuations. RMAC [19] exploits cross-layer routing information to setup a multi-hop schedule along a forwarding path. Then, nodes along this path wake up at the same time and forward data packets with reduced latency. To increase effective channel capacity under high traffic loads, DW-MAC [77] allows nodes to wake up on demand, even during the sleep period of a slot.

Random Access

Protocols in this class access the channel in an uncoordinated fashion using low power listening (LPL) or low power probing (LPP). In LPL [41], nodes periodically sample the channel for signs of activity. A sender transmits a long preamble to generate such activity and transmits the data packet upon receiving an acknowledgment from the intended receiver. In LPP [61], nodes periodically broadcast short beacon packets. A sender awaits a beacon from the intended receiver and replies immediately (or after sending an acknowledgment) with the data packet. Both variants free the nodes to synchronize their clocks. Moreover, as channel access is unrestricted, they provide a lot of flexibility to handle different node densities and traffic loads. The primary concern is keeping idle listening and collisions to a minimum.

Aloha with Preamble Sampling [24], WiseMAC [25], and B-MAC [67] propose LPL in the context of bitstream radios. STEM [74] uses a second low power radio for signaling upcoming transmissions. CSMA-MPS [58] and X-MAC [9] adopt LPL to packet-based radios by replacing the continuous preamble with a sequence of short packets. Koala [61] and RI-MAC [78] independently propose LPP. Koala uses LPP to enable data collection with duty cycles of 0.2 %. RI-MAC shows that LPP achieves better performance than LPL in X-MAC under a wide range of traffic loads.

Hybrid Protocols

Some protocols try to strike a balance between the flexibility of random access and the collision-free nature of TDMA. In Z-MAC [71], for example, nodes are assigned to the slots of a conflict-free schedule. However, even slot owners must contend for access using LPL when they want to send a message. If packet loss reaches a threshold, the protocol switches for 10 seconds into a contention-free mode to prevent further collisions. Crankshaft [38] and PMAC [102] schedule receivers, which may cause neighbors to share the same slot. Both protocols use different schemes to resolve contention in multi-allocated slots.

2.4.2 Optimization and Adaptation

There are some existing approaches for MAC protocol optimization and adaptation. Polaste *et al.* [67] present an analytical model of node lifetime for B-MAC. They note that this model can be used to recompute check interval and preamble length for varying traffic volume, but they do not implement adaptation functionality. Buettner *et al.* [9] derive models of energy consumption and per-hop latency for X-MAC but consider only energy for optimization. They demonstrate the benefit of adaptation in a two-node experiment by varying the sleep period for a given traffic volume. They do not address the issue of signaling the change to other nodes, which is important because *all* nodes in the network need to have the same sleep period so that X-MAC is reliable. Namboodiri and Keshavarzian [62] design an

adaptive algorithm for Alert that runs on the nodes. The objective is to minimize latency. Ye *et al.* [96] present a model of energy consumption to find optimal protocol parameters for SCP. Their approach is limited to configuration at compile time.

Compared with existing work on MAC protocol optimization and adaptation, our optimization approach takes into account *multiple performance metrics*; we find protocol parameters that optimize node lifetime, per-hop reliability, and per-hop latency simultaneously. Prior approaches do not address reliability; we adapt the MAC protocol to the current *packet loss rate* (and traffic volume) in the network and consider *per-hop reliability as an optimization objective*. Existing work seldom mentions the relevance of the application requirements; our approach *incorporates the application requirements* explicitly and ensures both optimal performance and compliance with the application requirements at runtime.

Chapter 3

Optimization Approach

This chapter describes our novel system for automatic parameter optimization and runtime adaptation of sensor network MAC protocols, followed by a case study where we implement our optimization approach in the ECL^{i}PS^e constraint programming system using an analytical model of X-MAC.

3.1 Runtime Adaptation

Wireless sensor networks exhibit a great diversity along different dimensions. First, the wide range of existing and envisioned applications of sensor networks gives rise to different requirements. In the medical domain [4], for example, a few mobile sensors may collaborate in an ad-hoc fashion for the duration of a hospital stay, whereas habitat monitoring [44] may require hundreds of static sensors to operate for more than a year. Second, available sensor hardware platforms vary in size, communication range, storage, and power consumption [6]. Each platform imposes different constraints that affect application performance. Third, sensor networks operate in diverse physical environments. Ranging from indoor to underwater, completely different radio propagation and durability conditions arise. Therefore, networking protocols need to be tuned on a per-application basis to work efficiently.

Some variations are more or less constant with respect to time, such as application requirements and hardware characteristics. An application designer should consider them carefully before putting the system into action. However, there are many aspects the application designer cannot be entirely aware of beforehand.

During the lifespan of an application various network properties change dynamically at unpredictable temporal and quantitative scales. For example, the instantaneous volume of data traffic in the network can change depending on whether the nodes sense regular or exceptional phenomena. This in turn affects the level of interference in the network and thus the packet reception rate at specific nodes. In fact, Zhao and Govindan [99] report variations in packet reception rate of 20 % to 60 % within a two-hour time window.

The sensor network must *adapt* to these dynamic changes to ensure optimal performance and compliance with the application requirements at all times. Moreover, the adaption procedure must be *automatic* to realize the vision of unattended and self-organizing wireless sensor networks [27, 45].

3.1.1 System Overview

In this thesis, we propose a system for automatic adaptation of sensor network MAC protocols. Using techniques from multiobjective optimization and constraint programming, we adjust the parameters of the MAC protocol based on the current network state such that the application requirements are satisfied and the performance of the MAC protocol is optimized.

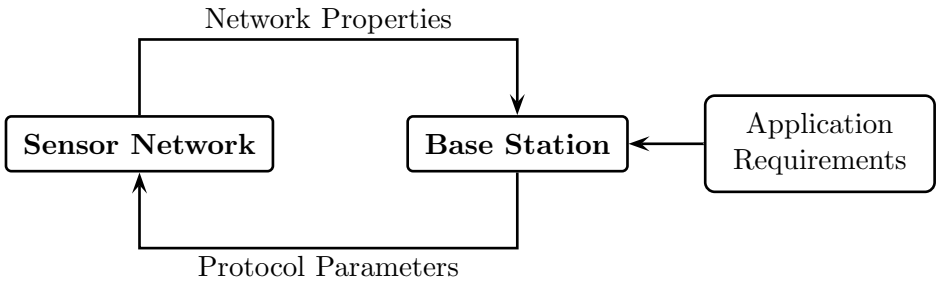


Figure 3.1: Base station and sensor network form a closed control loop.

Figure 3.1 illustrates our approach. The main architectural components are the sensor network and the base station. The base station serves as the central entity that collects information about the current state of the network. Based on these information and the application requirements, the base station computes appropriate MAC protocol parameters. These parameters are then disseminated back to the sensor nodes, which reconfigure their MAC layer with the newly received values. Sensor network and base station form a closed-loop control system [37, 63, 98], where the base station controlles the sensor network with respect to the application requirements and the current network state.

In the following, we describe each element of the control loop using Figure 3.2 as reference. Assume the sensor nodes execute some function and communicate using a pre-configured protocol stack. Besides their sensing tasks, nodes continuously keep track of packet loss rates and traffic volumes. These network properties are piggybacked on ordinary node-to-sink packets and collected by the base station. The base station feeds these data as well as the current protocol parameters into a model of the MAC protocol to compute an estimate of its performance. If the application requirements are not satisfied, the base station triggers the execution of an optimization component. The optimization component builds upon the protocol model and takes as input the current network properties and the application requirements. It computes a vector of MAC protocol parameters such that the expected performance of the MAC protocol is optimized and satisfies the application requirements. Finally, the new parameters are disseminated to the sensor nodes, which adapt their local parameters dynamically through a software interface provided by the MAC layer.

Many MAC protocols require that the protocol parameters are the same for all nodes in the network to guarantee reliable communication. For example, if nodes run the same low power listening MAC protocol but with different lengths of the sleep period, it may happen that the intended receiver never turns on its radio while the sender transmits a wake-up signal. Therefore, it is sufficient to collect aggregated views of network properties rather than from individual nodes. Efficient and robust

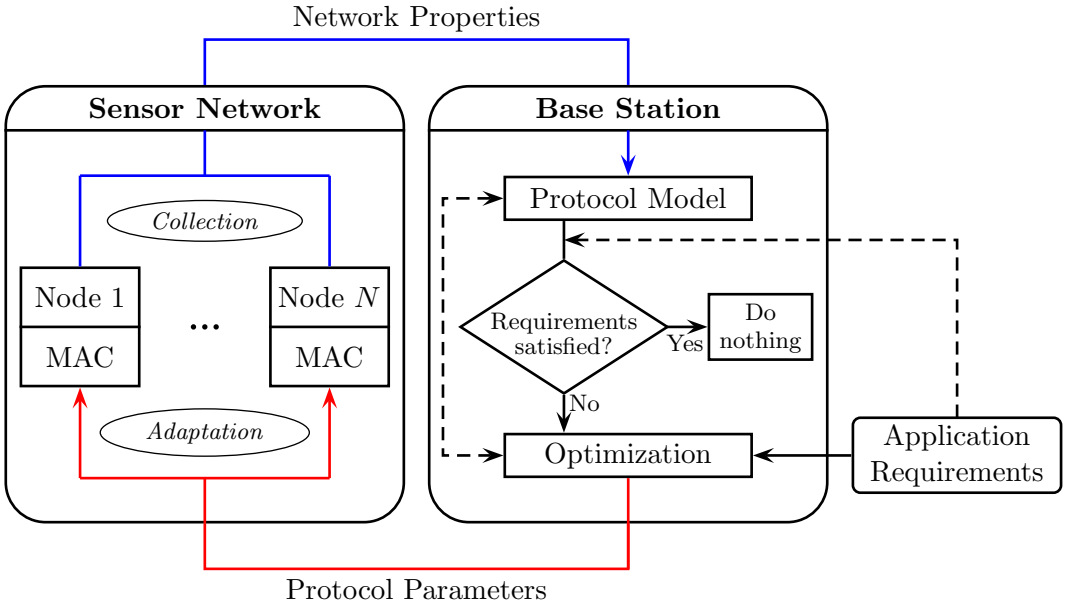


Figure 3.2: The base station collects the current network properties and decides, using the protocol model, whether the application requirements are satisfied. If this is not the case, the optimization component computes new parameters that are disseminated to the sensor nodes which adapt the MAC protocol accordingly.

methods exist for in-networking computation of such aggregates [13, 76]. In fact, the collection of network properties shows some resemblance with sensor network health monitoring [100, 101], where so-called network digests are computed to indicate system failures and resource depletion. Many deployed systems [34, 56, 59, 84] implement such a monitoring infrastructure to enable timely maintenance and to detect outliers in the data due to battery failures.

3.1.2 Network Properties

In this work we consider two network properties: *packet loss rate* and *traffic volume*. Packet loss rate is the fraction of packets that are transmitted within a time window, but not received. Traffic volume is the number of packets successfully received within a time window. The packet loss rate, or rather its complement, the packet reception rate, can be measured by analyzing the sender-generated sequence numbers embedded in packets. The traffic volume can be measured by counting the number of received packets.

We take into account *that* packets are lost, but we ignore *why* packets are lost. Many different factors affect packet loss over a wireless communication channel. The signal strength fading effect leads to low signal noise ratio over long distances, where the relation between transmission distance and packet loss is reported to be irregular [92, 101]. Environmental interference, which may be sporadic or constant, also contributes to packet loss. Packet collision between multiple transmitters, particularly the hidden terminal problem [82], is another factor. Heavy peaks in traffic may lead to buffer overflows and render a node unable to handle incoming packets.

Modeling all these effects, their mutual dependencies and influences on packet loss, is inherently complex. In fact, as we adapt the MAC protocol to the current network conditions, it is irrelevant what created these conditions in the first place. Iterative execution of the control loop aims at compensating for changes in the network properties caused by any phenomena, even for those caused by the protocol adaptation itself.

3.1.3 Protocol Model

The protocol model is used at the base station to determine the current performance of the MAC protocol. It is also used by the optimization component to compute optimal MAC protocol parameters. As depicted in Figure 3.3, the protocol model takes several input parameters and outputs MAC protocol performance with respect to a set of metrics. We identify three different performance metrics of a MAC protocol: *node lifetime*, *per-hop latency*, and *per-hop reliability*. Node lifetime refers to the time until a node exhausts its energy resources; per-hop latency is the time until a packet is successfully delivered over one hop; and per-hop reliability refers to the probability that a packet is successfully delivered over one hop.

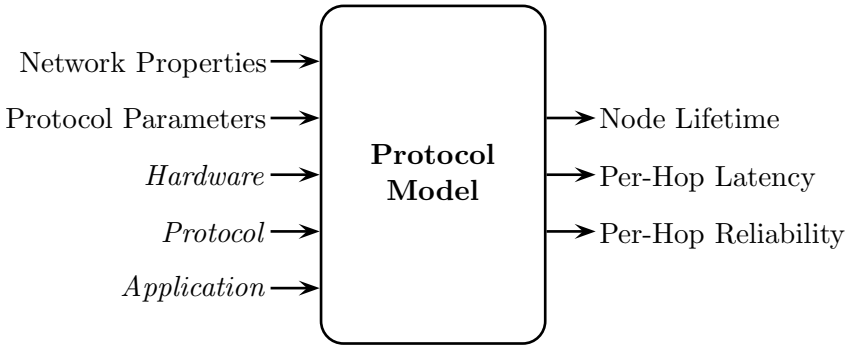


Figure 3.3: The protocol model maps variable and constant (printed in italics) input parameters onto the performance metrics node lifetime, per-hop latency, and per-hop reliability.

Our particular choice of performance metrics is motivated by the fact that the MAC protocol determines when neighboring nodes communicate. Since it does this by controlling the radio, the most power-consuming component of a typical sensor node [68], it is natural to consider node lifetime as a performance metric. Moreover, the timing of neighbor-to-neighbor communication affects per-hop latency and per-hop reliability, which clearly define the baseline of end-to-end network performance.

The set of protocol model parameters can be broadly divided into *variable* and *constant* parameters. Variable parameters may change value from one control loop iteration to the next, whereas constant input parameters are fixed for a reasonable number of iterations, and possibly throughout the entire lifespan of an application. Variable model parameters are the network properties (packet loss rate and traffic volume) and the protocol parameters. We identify three different classes of constant model parameters: *hardware-dependent*, *protocol-dependent*, and *application-dependent*. Typical hardware-dependent constants are battery capacity, supply voltage, current consumptions of the radio in different operational modes (sleep, idle, receive, and listen), and transition times among the modes. Protocol-dependent constants are, for example, the header sizes of different protocols in

the communication stack, which are required to compute packet transmission and reception times. For the same reason, the application-dependent size of the actual payload of a packet is needed.

Node Lifetime

The lifetime of a node is determined by its overall energy consumption. If the lifetime is maximized, then the energy consumption must be minimized.

A node consumes energy by sensing, processing, and communicating. Sensing energy is application-specific and independent of the MAC protocol. Energy used by MAC protocol processing is, by and large, a fixed cost—adapting the parameters of the MAC protocol leaves its processing overhead nearly unchanged. Conversely, energy used by communicating depends on the MAC timing parameters that determine whether the radio is in power-saving or power-intensive mode. This makes the MAC protocol most often the key factor in a node's overall energy consumption [23]. We consider the communication-related energy consumption.

The energy used by a node, E, consists of the energy consumed by receiving, transmitting, and sleeping.

$$E = E_{rx} + E_{tx} + E_s \tag{3.1}$$

The terms in (3.1) are expressed in units of joules per second, that is, E denotes the energy consumed within a second. Calculating the total energy usage can be done by multiplying E by the node lifetime. We give concrete expressions for the individual energy consumptions in our case study (see Section 3.2).

The lifetime of a node, T_l, is dependent on its energy consumption, E, the battery capacity, Q, and the supply voltage, U.

$$T_l = \frac{Q \times U}{P} \tag{3.2}$$

Using (3.2), we can calculate the lifetime of a node in seconds.

Per-Hop Latency and Per-Hop Reliability

Per-hop latency and per-hop reliability depend on the type of MAC protocol.

In TDMA-based protocols, for example, the number of slots within a frame and the duration of a slot determine the per-hop latency. After receiving a packet, a node has to wait until its next scheduled slot in the following frame to forward the packet. The number of slots in a frame is given by the number of sender-receiver pairs (link-based scheduling) or the number of senders (sender-based scheduling). Since both variants of TDMA are free of collisions, the per-hop reliability amounts to 100 % with ideal radio propagation.

In contention-based protocols, the per-hop reliability is strictly less than 100 %. Nodes contend for the channel, and packet loss is unavoidable. The per-hop latency depends on the packet loss rate—lost packets must be retransmitted—and the length of the receiver sleep period. In sender-initiated data transmission (low power listening), the sender transmits a wake-up signal until the receiver recovers from sleep and sends an acknowledgment. In receiver-initiated data transmission (low power probing), the sender stays active silently waiting for a beacon from the receiver. In either case, the receiver sleep period determines the average waiting time before the actual packet transmission and thus the per-hop latency. Moreover, in contention-based protocols, the per-hop latency is much more variable than in TDMA-based protocols.

3.1.4 Optimization

The optimization component computes MAC protocol parameters such that the performance of the MAC protocol is optimized and the application requirements are satisfied. This amounts to solving a multiobjective optimization problem, where the performance metrics (node lifetime, per-hop latency, and per-hop reliability) are the objectives and the protocol parameters are the decision variables. The choice of protocol parameters depends on the particular MAC protocol.

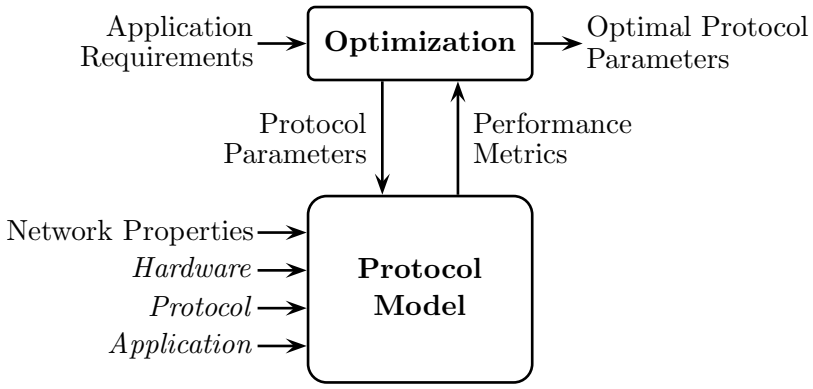


Figure 3.4: Given the application requirements, the optimization component interacts with the protocol model while computing the optimal protocol parameters.

As shown in Figure 3.4, the optimization component takes as input the application requirements and uses the protocol model during the optimization process to calculate the performance metrics for the protocol parameters currently selected. If the problem is solvable, that is, there exist parameters for which the application requirements are satisfied, the optimization component outputs the optimal MAC protocol parameters. The application requirements may be changed between successive executions of the optimization component.

To perform the optimization step, we adopt an approach based on goal programming (see Section 2.2.2). Using this method, we have to assign goals that we want to achieve for each objective. We consider the application requirements as goals and incorporate these values as additional constraints into the optimization problem.

Objective	Requirement	Description
$T_l(\mathbf{x})$	T_l^*	Minimal required node lifetime
$L(\mathbf{x})$	L^*	Maximal tolerable per-hop latency
$P(\mathbf{x})$	P^*	Minimal required per-hop reliability

Table 3.1: Application requirements are upper and lower bounds on the performance metrics.

Assume the application requirements are specified as lower and upper bounds on the objectives. As shown in Table 3.1, these are the minimal node lifetime, T_l^*, the maximal per-hop latency, L^*, and the minimal per-hop reliability, P^*. Here, $\mathbf{x}$ denotes the vector of decision variables of the optimization problem, which are the

protocol parameters that we seek to find. We add the following three constraints to the optimization problem.

$$T_l(\mathbf{x}) \geq T_l^* \tag{3.3}$$

$$L(\mathbf{x}) \leq L^* \tag{3.4}$$

$$P(\mathbf{x}) \geq P^* \tag{3.5}$$

In this way, we ensure that the computed protocol parameters yield a performance that satisfies the application requirements. The objective function tries to minimize the absolute deviations from the targets to the objectives.

$$\min \left(|T_l(\mathbf{x}) - T_l^*| + |L(\mathbf{x}) - L^*| + |P(\mathbf{x}) - P^*| \right) \tag{3.6}$$

We use constraint programming (see Section 2.3) to solve this optimization problem. We have to specify the decision variables (protocol parameters) and their respective domains, the individual objectives (performance metrics), the constraints, and the objective function. The resulting constraint optimization problem is given to a constraint solver that seeks to find a solution, that is, an assignment of values to the decision variables such that all constraints are satisfied and the objective function is minimized.

3.1.5 MAC Layer Interface

In addition to the standard message interfaces, the MAC layer must provide at least two interfaces that allow upper layer services to query and adjust the parameters of the MAC protocol. In their most generic form, these interfaces may look as follows.

```
struct mac_config { ... };

const struct mac_config * getMACConfig();
void setMACConfig(const struct mac_config * config);
```

The data type `mac_config` encapsulates the adjustable parameters of the MAC protocol. Upper layer services call `getMACConfig` to retrieve the current protocol parameters and `setMACConfig` to adapt the protocol parameters.

3.1.6 Collection and Dissemination

Our system for automatic adaptation of sensor network MAC protocols relies on two core services: a collection service to enable continuous acquisition of network properties and a dissemination service to enable on-demand updates of protocol parameters. These services are not specific to automatic protocol adaptation, but have been part of many sensor network deployments. Experience has shown that a human manager should be able to quickly determine whether a deployed network is functioning and to remotely reprogram the sensor nodes [11, 59, 84]. Several protocols have been developed with these objectives in mind that can be leveraged by our proposed system.

MintRoute [91] and CTP [28] are data collection protocols that organize nodes into a tree rooted at the base station. Individual nodes estimate link quality by observing packet success and loss events. These estimates are then used to form a tree that minimizes the expected number of transmissions necessary to forward a message to the root.

Trickle [53] addresses single packet dissemination and uses periodic retransmissions to ensure the delivery of a packet to every node in the network. To limit retransmissions among neighboring nodes, a node suppresses its own broadcast if it recently overhears a similar message. Deluge [42] builds upon Trickle and supports dissemination of large data objects. A three-phase handshaking protocol helps to ensure that a bidirectional link exists before transferring data, and spatial multiplexing enables parallel transfers of data.

The Sensor Network Management System (SNMS) [83] combines collection and dissemination mechanisms in an interactive management system for wireless sensor networks. Most notably, SNMS includes an instrumented radio stack that provides a set of remotely queryable counters, such as outgoing message notifications and bidirectional message deliveries. While being a data gathering protocol for periodic monitoring applications, Dozer [10] also disseminates commands injected at the data sink by repeatedly piggybacking on beacon messages.

3.2 Case Study

We apply our optimization approach to a concrete sensor network MAC protocol.

3.2.1 X-MAC Overview

X-MAC [9] is a power-saving MAC protocol for low traffic applications. It belongs to the class of contention-based protocols with channel sampling. That is, there exists no common schedule among the nodes that coordinates contention periods. Instead, nodes run an asynchronous duty cycle and briefly sample the channel for activity. This avoids synchronization overhead, provides flexibility to accommodate dynamic changes (for example, nodes joining or leaving the network), and makes the overall protocol rather simple.

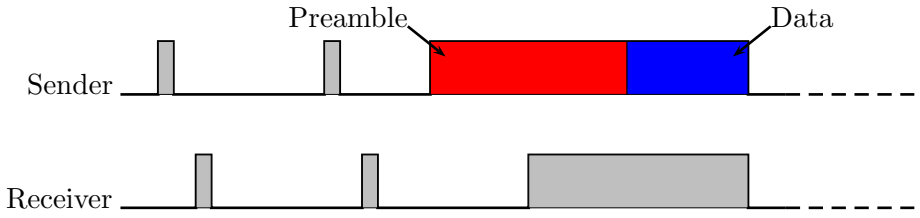


Figure 3.5: Periodic sampling in LPL. A long preamble before the data packet signals an incoming transmission, allowing the receiver to sleep most of the time.

Low power listening (LPL) is the underlying mechanism of X-MAC, which was independently developed by Hill and Culler [41] and El-Hoiydi [24].[1] To let receivers sleep most of the time, nodes wake up periodically and check for activity on the channel. If the channel is idle, the receiver goes back to sleep. Otherwise, the receiver keeps listening until data transmission is finished. Data transmissions are announced by sending a long preamble before sending the data packet. This effectively shifts the burden from the receiver to the sender. A lot of energy is saved due to reduced

[1] Polastre *et al.* [67] coined the term *low power listening* in a follow-up paper introducing the B-MAC protocol.

idle listening at the receiver if there are many more receivers than senders and data traffic is low. Figure 3.5 illustrates periodic sampling in LPL.

X-MAC adapts LPL to packet-based radios. Instead of sending a long, continuous preamble in front of the data packet, a node transmits a sequence of short radio packets, called *strobes*. X-MAC uses strobes to improve unicast communication. Each strobe packet contains the address of the target receiver. The sender inserts short pauses into the sequence of strobes, which allows the target receiver to respond in between by sending an *early acknowledgment*. This shortens the strobed preamble, thereby achieving energy savings at both the sender and the receiver. Moreover, non-target receivers can immediately go back to sleep when they overhear a strobe. For broadcast communication, however, the entire strobed preamble must be transmitted to wake up all neighbors.

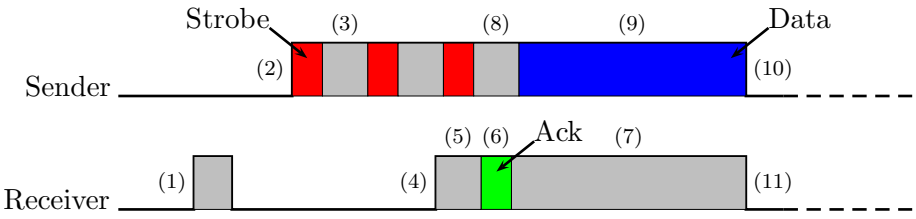


Figure 3.6: Unicast transaction in X-MAC.

Figure 3.6 depicts a unicast transaction in X-MAC. Nodes turn on their radio at regular intervals to listen for strobes (1). To send a data packet, the sender starts transmitting a sequence of strobes (2). Between successive strobes, the sender listens for an acknowledgment from the intended receiver (3). The receiver wakes up (4), detects a strobe tagged with its own address (5), acknowledges the strobe (6), and keeps its radio on awaiting the data packet (7). Upon receiving the acknowledgment (8), the sender terminates the strobe sequence, transmits the data packet (9), and goes back to sleep (10). The receiver turns off its radio after hearing the data packet (11).

We choose X-MAC for our case study because it is both simple and flexible. Large-scale application deployments [79, 84] used LPL, the underlying mechanism of X-MAC, primarily because of its simplicity and robustness [70]. X-MAC's asynchronous operation reduces configuration efforts and allows the protocol to handle dynamic changes. X-MAC is the default MAC protocol in Contiki [20]; one of its predecessors, B-MAC [67], is the default MAC protocol in TinyOS [52]. Moreover, we believe that our work on X-MAC is representative for the whole class of asynchronous LPL protocols, including CSMA-MPS [58] and B-MAC [67].

3.2.2 Protocol Parameters

The parameters of a MAC protocol control its execution on individual nodes and the timing of communication among neighboring nodes. We can improve the protocol's performance by deliberately changing its parameters. The optimization component (see Figure 3.4) tells us how we should set the parameters to achieve optimal performance and to satisfy the application requirements. The protocol parameters determined by the optimization component are the *decision variables* of the optimization problem. The set of decision variables may be different from the

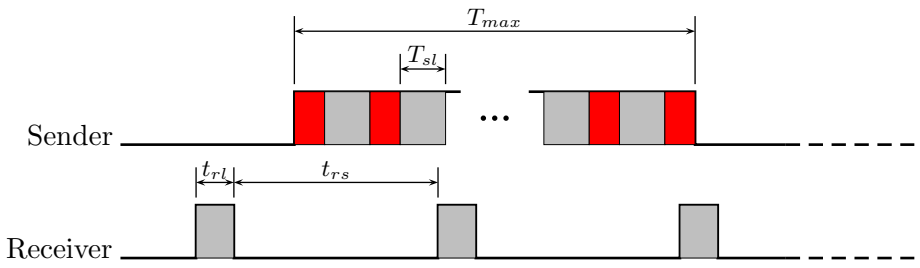


Figure 3.7: Protocol parameters of X-MAC.

set of protocol parameters. Some protocol parameters may be predefined by other system parameters or deliberately left out to simplify the optimization problem.

X-MAC has four protocol parameters, as shown in Figure 3.7:

- The length of the receiver listen period, t_{rl}.
- The length of the receiver sleep period, t_{rs}.
- The length of the sender listen period waiting for an acknowledgment, T_{sl}.
- The maximal length of the sender strobe sequence, T_{max}.

The receiver listen and sleep periods, t_{rl} and t_{rs}, determine a node's duty cycle. A low duty cycle conserves energy but leads to longer per-hop latency. Note that the period of the duty cycle, $t_{rl} + t_{rs}$, determines how long it takes, on average, for the receiver to wake up and acknowledge a strobe packet. The duty cycle period also affects the maximal length of the sender strobe sequence, T_{max}, because it must be guaranteed that the receiver wakes up at least once while the sender transmits strobe packets. Otherwise, X-MAC would be unreliable. The sender listen period, T_{sl}, must be long enough for the sender to be able to receive the acknowledgment packet. Similarly, the receiver listen period, t_{rl}, must be long enough for the receiver to be able to receive a strobe packet. In Section 3.2.4, we formally state these and other constraints on the protocol parameters.

We consider three decision variables for the parameter optimization of X-MAC:

- The length of the receiver listen period, t_{rl}.
- The length of the receiver sleep period, t_{rs}.
- The number of transmissions per packet, n_t.

The sender should listen as short as possible for an acknowledgment [9]. Buettner *et al.* [9] report that they could not schedule listen periods shorter than 20 ms. In Contiki, the sender listens for about 4 ms, which is a major improvement. We see that the sender listen period, T_{sl}, is predetermined by the operating system and the radio hardware. Therefore, we consider T_{sl} as a constant and exclude it from the set of decision variables. We also exclude the maximal length of the sender strobe sequence, T_{max}, since this protocol parameter is determined by the decision variables t_{rl} and t_{rs}. We can say that T_{max} is a *derived* protocol parameter; we discuss its relation to the decision variables in more detail in Section 3.2.3.

Table 3.2: Terms in the protocol model of X-MAC.

Term	Type	Unit	Description
P_{tx}	H	W	Power in transmit mode
P_{rx}	H	W	Power in receive mode
P_s	H	W	Power in sleep mode
U	H	V	Supply voltage
Q	H	C	Battery capacity
T_{dtx}	H	s	Pre-transmission delay (Rx/Tx turnaround)
T_{drx}	H	s	Pre-reception delay (Tx/Rx turnaround)
T_{str}	H/P	s	Strobe transmission
T_{ack}	H/P	s	Acknowledgment transmission
T_{data}	H/P/A	s	Data packet transmission
T_{sl}	P	s	Sender acknowledgment listen period
T_{wait}	P	s	Receiver waiting for data packet
P_{str}	N	–	Prob. of successful strobe transmission
P_{ack}	N	–	Prob. of successful acknowledgment transmission
P_{data}	N	–	Prob. of successful data packet transmission
R_0	N	packets/s	Traffic volume
T_{max}	Pa	s	Maximal length of strobe sequence
t_{rl}	Pa/D	s	Receiver listen period
t_{rs}	Pa/D	s	Receiver sleep period
n_t	Pa/D	-	Transmissions per packet

We introduce the number of transmissions per packet, n_t, as a decision variable, although it is not a true parameter of X-MAC. Rather, n_t is a system parameter that indicates how many times a packet must be sent to guarantee its reception *with a given probability*. In a realistic environment, the packet reception rate is strictly less than 100 % due to packet loss. Therefore, it may be necessary to transmit a packet more than once to satisfy the application's reliability requirements. We account for this fact and consider n_t in our analysis.

3.2.3 Protocol Model of X-MAC

Given a set of input parameters, the protocol model determines the performance of the MAC protocol. In this section, we derive expressions for per-hop reliability, node lifetime, and per-hop latency of X-MAC.

Table 3.2 lists the terms that appear throughout the presentation. Variable terms are network properties (N), protocol parameters (Pa), and decision variables (D). Constant terms are hardware-dependent (H), protocol-dependent (P), or application-dependent (A) constants. For example, the length of a data packet transmission, T_{data}, depends on the transmit bit rate of the radio device, the header sizes of different protocols in the communication stack, and the size of the application payload. Note that we distinguish packet reception probabilities based on packet type (strobe, acknowledgment, data packet).

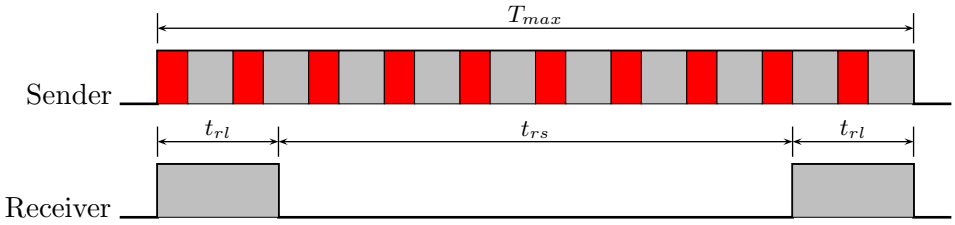


Figure 3.8: Maximal length of strobe sequence.

Per-Hop Reliability

Per-hop reliability, P, refers to the probability that a packet is successfully delivered over one hop within n_t transmissions.

A *successful* unicast transmission consists of three phases (see Figure 3.6). First, at least one strobe packet arrives at the target receiver; this happens with probability $P(\text{one strobe})$. Second, the corresponding acknowledgment arrives at the sender; this happens with probability P_{ack}. Third, the data packet arrives at the target receiver; this happens with probability P_{data}. Therefore, a single transmission is successful with probability

$$P(\text{one strobe}) \times P_{ack} \times P_{data}. \tag{3.7}$$

The sender transmits each packet n_t times. The probability that at least one of n_t transmissions succeeds—the per-hop reliability—is given by

$$P = 1 - [1 - P(\text{one strobe}) \times P_{ack} \times P_{data}]^{n_t}. \tag{3.8}$$

In the following we derive $P(\text{one strobe})$, the probability that at least one strobe arrives at the target receiver.

The receiver is only able to hear a strobe if the sender keeps on sending strobes until the receiver wakes up and starts to listen. In fact, as shown in Figure 3.8, the maximal length of the strobe sequence, T_{max}, must be at least twice the length of the receiver listen period, t_{rl}, plus the length of the receiver sleep period, t_{rs}. If the strobe sequence is even longer, it may happen that the receiver wakes up twice for the entire listen period during the strobe sequence. Since this would waste energy at the sender, we set

$$T_{max} = 2\,t_{rl} + t_{rs}. \tag{3.9}$$

Furthermore, we have to ensure that the receiver listens long enough so it is able to receive a complete strobe packet. When the sender wants to transmit a strobe, it first switches within T_{dtx} from receive (or sleep) mode into transmit mode; transmits the strobe packet, which takes T_{str} of time; switches within T_{drx} back into receive mode; and waits T_{sl} for an acknowledgment. We call this sequence a *strobe iteration*. Thus, to be able to receive a complete strobe packet, the receiver must listen for at least the time from the start of a strobe transmission to the end of the subsequent strobe transmission.

$$t_{rl} \geq 2\,T_{str} + T_{drx} + T_{dtx} + T_{sl} \tag{3.10}$$

Figure 3.9 illustrates this lower bound on t_{rl}.

The transmission of a single strobe succeeds with probability P_{str}. In harsh environments with high packet loss, it is likely that a strobe transmission fails.

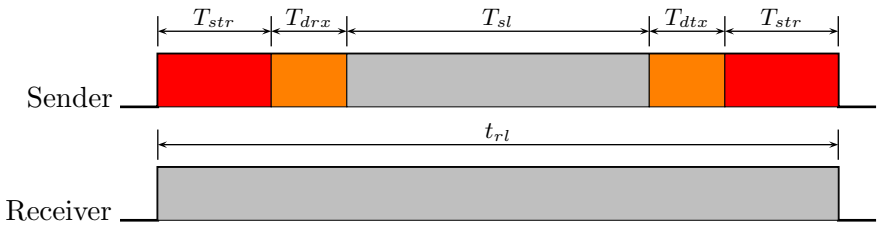


Figure 3.9: Lower bound on receiver listen period.

Therefore, we want the receiver to be able to receive two strobes within a listen period. At the same time, we want to preserve the idea of low power listening, namely short wake-up periods. We have to set an upper bound on t_{rl} as well.

Whether the receiver is able to hear a complete second strobe depends on the time when the receiver wakes up and starts to listen. As shown in Figure 3.10, the receiver may start to listen at any time with respect to a strobe iteration at the sender of length

$$a + b = T_{dtx} + T_{str} + T_{drx} + T_{sl}. \tag{3.11}$$

However, the receiver hears a complete second strobe only if it starts to listen within time period

$$b = t_{rl} - 2\,T_{str} - T_{dtx} - T_{drx} - T_{sl}. \tag{3.12}$$

Therefore, the probability that the receiver is able to receive a second strobe, $P(\text{2nd strobe})$, is given by

$$P(\text{2nd strobe}) = \frac{b}{a+b} = \frac{t_{rl} - 2\,T_{str} - T_{dtx} - T_{drx} - T_{sl}}{T_{dtx} + T_{str} + T_{drx} + T_{sl}}. \tag{3.13}$$

The upper bound on t_{rl},

$$t_{rl} \leq 3\,T_{str} + 2\,(T_{dtx} + T_{drx} + T_{sl}), \tag{3.14}$$

guarantees that the receiver cannot hear more than two complete strobes in the same listen period. Moreover, constraints (3.10) and (3.14) ensure that (3.13) is sound, that is, $0 \leq P(\text{2nd strobe}) \leq 1$ holds.

We can now deduce $P(\text{one strobe})$, the probability that at least one strobe arrives at the target receiver. When the receiver starts to listen, it receives the first strobe with probability P_{str}. Otherwise, it receives the second strobe with probability $P(\text{2nd strobe}) \times P_{str}$. Thus, we have

$$P(\text{one strobe}) = P_{str} + [1 - P_{str}] \times P(\text{2nd strobe}) \times P_{str}. \tag{3.15}$$

Plugging (3.15) in (3.8) yields the per-hop reliability of X-MAC. Note that the per-hop reliability, P, is independent of the length of the receiver sleep period, t_{rs}.

Node Lifetime

We adapt (3.1) to X-MAC and model the energy consumption of a node as follows.

$$E = E_{rx} + E_{tx} + E_{cycle} \tag{3.16}$$

The term E_{rx} refers to the energy used by receiving packets, and E_{tx} refers to the energy used by sending packets. With X-MAC, a node switches the radio on and off at regular intervals when it is not receiving or transmitting. The term E_{cycle} accounts for this duty cycle energy.

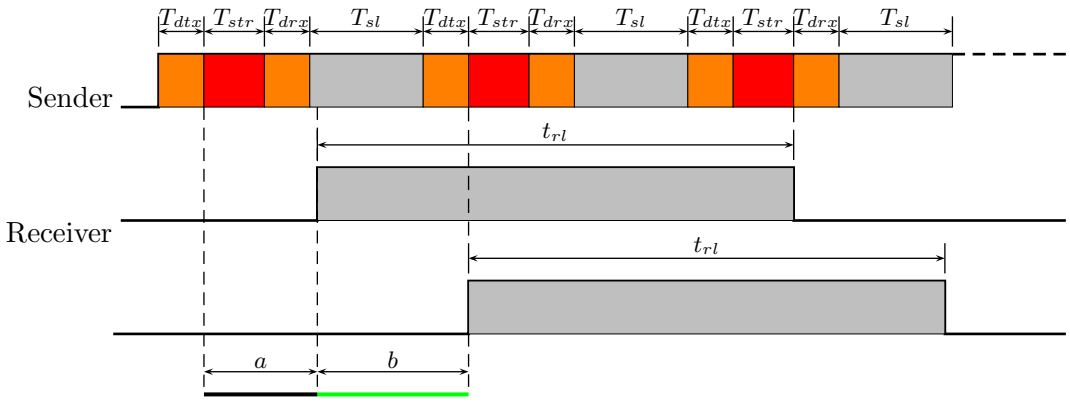


Figure 3.10: Probability of hearing a complete second strobe in a receiver listen period.

Energy Consumed by Receiving The energy consumed by receiving packets is the energy used by trying to receive a single packet, $E_{rxpacket}$, times the number of packets that a node tries to receive, R_{rx}.

$$E_{rx} = E_{rxpacket} \times R_{rx} \tag{3.17}$$

The number of packets that a node tries to receive is usually less than the number of incoming packets. A sender stops sending strobes if it receives an acknowledgment, but at the latest after the maximal length of the strobe sequence, T_{max}. That is, if a node does not receive a strobe within T_{max}, it never attempts to receive the incoming packet. Therefore, the number of packets that a node tries to receive is the number of incoming packets times the probability that at least one strobe arrives, $P(\text{one strobe})$. The number of incoming packets is $n_t R_0$, since each packet is sent n_t times. Thus, we have

$$R_{rx} = n_t R_0 \times P(\text{one strobe}), \tag{3.18}$$

where $P(\text{one strobe})$ is given by (3.15).

We now want to deduce, $E_{rxpacket}$, the energy consumed by trying to receive a packet. In this case, a node sends an acknowledgment back to the sender and awaits the reception of the data packet.

$$E_{rxpacket} = E_{ack} + E_{rxdata} \tag{3.19}$$

Sending an acknowledgment consumes

$$E_{ack} = P_{tx}(T_{dtx} + T_{ack}) \tag{3.20}$$

of energy. A node goes back to sleep immediately after receiving the data packet. Otherwise, if the data packet is not received, a node keeps listening for a duration of T_{wait} before it switches into sleep mode. Therefore, the energy used by receiving the data packet is

$$E_{rxdata} = E_{rxdata1} + E_{rxdata2} \tag{3.21}$$

$$E_{rxdata1} = P_{rx}(T_{drx} + T_{data}) \times P(\text{rx data} \,|\, \text{one strobe}) \tag{3.22}$$

$$E_{rxdata2} = P_{rx} T_{wait} \times [1 - P(\text{rx data} \,|\, \text{one strobe})]. \tag{3.23}$$

Here, $P(\text{rx data} \,|\, \text{one strobe})$ denotes the conditional probability that the data packet is successfully received, given that at least one strobe arrives. The event of receiving at least one strobe and the event of successfully transmitting acknowledgment and data packet are statistically independent—getting a strobe makes it neither more nor less likely that acknowledgment and data packet are successfully transmitted, and vice versa. Therefore, by the definition of conditional probability, we have

$$P(\text{rx data} \,|\, \text{one strobe}) = \frac{P(\text{one strobe}) \times P(\text{rx data})}{P(\text{one strobe})} = P(\text{rx data}), \tag{3.24}$$

where $P(\text{one strobe})$, the *prior* probability that at least one strobe arrives, is given by (3.15). The *prior* probability that the data packet is successfully received, $P(\text{rx data})$, is given by

$$P(\text{rx data}) = P_{ack} \times P_{data}, \tag{3.25}$$

because both the transmission of the acknowledgment and the transmission of the data packet must be successful.

Energy Consumed by Transmitting The energy consumed by transmitting is the energy used for sending a single packet, $E_{txpacket}$, times the number of packets a node has to send, R_{tx}.

$$E_{tx} = E_{txpacket} \times R_{tx} \tag{3.26}$$

We first consider R_{tx}, the number of packets to send. This is the number of *unique* packets that a node is able to receive successfully, R_{rxsucc}, times the number of transmissions per packet, n_t, because a node ignores duplicates.

$$R_{tx} = R_{rxsucc} \times n_t \tag{3.27}$$

The number of unique packets that a node receives successfully is the number of incoming packets, R_0, times the probability that a packet is successfully delivered, which is essentially the per-hop reliability P and given by (3.8).

$$R_{rxsucc} = R_0 \times P \tag{3.28}$$

As for the energy consumed by sending a single packet, $E_{txpacket}$, we have to distinguish between two cases: a node receives at least one acknowledgment and a node receives no acknowledgment from the target receiver within T_{max}. The former case occurs with probability $P(\text{one ack})$ and implies that the data packet is eventually transmitted. The latter case occurs with probability $1 - P(\text{one ack})$ and implies that the data packet is not transmitted. Also, the energies used for sending strobes and listening for acknowledgments are different in both cases. Thus, the energy consumed by sending a single packet is given by

$$E_{txpacket} = E_{txpacket1} + E_{txpacket2} \tag{3.29}$$

$$E_{txpacket1} = (E_{strobes1} + E_{txdata}) \times P(\text{one ack}) \tag{3.30}$$

$$E_{txpacket2} = E_{strobes2} \times [1 - P(\text{one ack})]. \tag{3.31}$$

The probability that a node receives at least one acknowledgment within T_{max}, the maximal length of the preamble sequence, is

$$P(\text{one ack}) = P(\text{one strobe}) \times P_{ack}, \tag{3.32}$$

where the probability that at least one strobe is successfully received, P(one strobe), is given by (3.15). If at least one acknowledgment arrives, a node consumes

$$E_{txdata} = P_{tx}(T_{dtx} + T_{data}) \tag{3.33}$$

of energy for sending the data packet and

$$E_{strobes1} = [P_{tx}(T_{dtx} + T_{str}) + P_{rx}(T_{drx} + T_{sl})] \times N_{exp} \tag{3.34}$$

of energy by sending strobes and listening for acknowledgments. Here, N_{exp} denotes the expected number of strobe iterations until an acknowledgment arrives. An acknowledgment can only arrive if a strobe arrives before on the receiver side, which in turn requires that the receiver is awake. Because the receiver wakes up exactly once while a node is strobing, the expected number of strobe iterations, N_{exp}, is the expected time until the receiver wakes up, T_{exp}, divided by the length of a strobe iteration.

$$N_{exp} = \frac{T_{exp}}{T_{dtx} + T_{str} + T_{drx} + T_{sl}} \tag{3.35}$$

In X-MAC, nodes know nothing about each others duty cycle, that is, there is no synchronization among the nodes. From the sender's point of view, the receiver can wake up after any time period t with $0 \leq t \leq t_{rl} + t_{rs}$. In the best case, the receiver wakes up immediately ($t = 0$); in the worst case, the receiver wakes up after a complete duty cycle ($t = t_{rl} + t_{rs}$). Since nodes are asynchronous, any time period t is equally likely, and on average the receiver wakes up after half the duty cycle.

$$T_{exp} = \frac{t_{rl} + t_{rs}}{2} \tag{3.36}$$

Otherwise, if no acknowledgment arrives, a node sends strobes for T_{max}; the data packet is not transmitted. The energy used by sending strobes is given by

$$E_{strobes2} = [P_{tx}(T_{dtx} + T_{str}) + P_{rx}(T_{drx} + T_{sl})] \times N_{max}. \tag{3.37}$$

The number of strobes sent in this case, N_{max}, is the maximum length of the strobed preamble, T_{max}, divided by the length of a strobe iteration.

$$N_{max} = \frac{T_{max}}{T_{dtx} + T_{str} + T_{drx} + T_{sl}} \tag{3.38}$$

Energy Consumed by Duty Cycling A node runs a periodic duty cycle whenever it is not receiving or transmitting. The time spent duty cycling, t_{cycle}, is the time remaining each second that is not consumed by other operations.

$$t_{cycle} = 1 - t_{rx} - t_{tx} \tag{3.39}$$

Then, the energy consumed by duty cycling, E_{cycle}, is the energy used for a single cycle times the number of cycles within t_{cycle}. Note that the duration of a single duty cycle is $t_{rl} + t_{rs}$.

$$E_{cycle} = (P_{rx}t_{rl} + P_s t_{rs}) \times \frac{t_{cycle}}{t_{rl} + t_{rs}} \tag{3.40}$$

In the following, we want to determine the time spent receiving, t_{rx}, and the time spent transmitting, t_{tx}, needed to solve (3.39).

A node tries to receive R_{rx} packets within a second, where trying to receive a single packet takes $t_{rxpacket}$ of time. Therefore, the time spent receiving is given by

$$t_{rx} = t_{rxpacket} \times R_{rx}. \tag{3.41}$$

Trying to receive a single packet involves sending an acknowledgment and subsequently awaiting the reception of the data packet, given that at least one strobe packet arrives. Therefore, the time spent trying to receive a single packet is

$$t_{rxpacket} = t_{rxpacket1} + t_{rxpacket2} \tag{3.42}$$
$$t_{rxpacket1} = T_{dtx} + T_{ack} + (T_{drx} + T_{data}) \times P(\text{rx data}) \tag{3.43}$$
$$t_{rxpacket2} = T_{wait} \times [1 - P(\text{rx data})], \tag{3.44}$$

where $P(\text{rx data})$ is given by (3.25).

A node transmits R_{tx} packets within a second, and transmitting a single packet takes $t_{txpacket}$ of time. Therefore, the time spent transmitting is given by

$$t_{tx} = t_{txpacket} \times R_{tx}. \tag{3.45}$$

The data packet is only sent if at least one acknowledgment arrives within T_{max}. This happens with probability $P(\text{one ack})$, given by (3.32). If no acknowledgment arrives, a node sends strobes for T_{max}. The time spent sending a single packet is therefore

$$t_{txpacket} = t_{txpacket1} + t_{txpacket2} \tag{3.46}$$
$$t_{txpacket1} = (T_{exp} + T_{dtx} + T_{data}) \times P(\text{one ack}) \tag{3.47}$$
$$t_{txpacket2} = T_{max} \times [1 - P(\text{one ack})], \tag{3.48}$$

where T_{exp} is the expected time until an acknowledgment arrives and $T_{dtx} + T_{data}$ is the time needed to send the data packet.

Per-Hop Latency

Per-hop latency, L, refers to the time needed to successfully deliver a packet over one hop to a neighboring node. That is, the packet arrives *for sure* at the receiver.

A single packet transmission takes $t_{txpacket}$ of time, given by (3.46). However, one transmission is only successful with probability $P(\text{one strobe}) \times P_{ack} \times P_{data}$. Therefore, the per-hop latency is

$$L = t_{txpacket} \times [P(\text{one strobe}) \times P_{ack} \times P_{data}]^{-1}, \tag{3.49}$$

where $P(\text{one strobe})$ is given by (3.15). Note that the per-hop latency is independent of the number of transmissions per packet, n_t.

3.2.4 Constraints

The following constraints describe the range of values for which our protocol model of X-MAC is reasonable.

- To guarantee that the target receiver listens exactly once for an entire listen period while the sender transmits strobes, the maximal length of the strobe

sequence, T_{max}, must be equal to the length of two receiver listen periods plus the length of the receiver sleep period.

$$T_{max} = 2\,t_{rl} + t_{rs} \tag{3.50}$$

This equality constraint is illustrated by Figure 3.8.

- For the receiver to be able to receive one strobe, the receiver listen period, t_{rl}, must be at least as long as a complete strobe iteration plus the duration of a strobe transmission.

$$t_{rl} \geq 2\,T_{str} + T_{dtx} + T_{drx} + T_{sl} \tag{3.51}$$

This lower bound on t_{rl} is illustrated by Figure 3.9.

- The receiver should not be able to hear more than two strobes within the same listen period. This is to preserve the idea of low power listening, where nodes are supposed to drastically reduce idle listening by periodically sampling the channel for a very short time. Therefore, the receiver listen period must be shorter than two strobe iterations plus the duration of a strobe transmission.

$$t_{rl} \leq 2\,(T_{str} + T_{dtx} + T_{drx} + T_{sl}) + T_{str} \tag{3.52}$$

This upper bound on t_{rl} is illustrated by Figure 3.10.

- For the sender to be able to receive an acknowledgment, the sender listen period, T_{sl}, must be longer than the time needed to switch into receive mode plus the duration of an acknowledgment transmission.

$$T_{sl} \geq T_{drx} + T_{ack} \tag{3.53}$$

- After sending an acknowledgment, the receiver keeps its radio on in anticipation of the data packet. To be able to receive the whole data packet, the receiver must be awake at least the time it takes to switch into receive mode plus the duration of a data packet transmission.

$$T_{wait} \geq T_{drx} + T_{data} \tag{3.54}$$

- A node is overloaded if it is unable to handle all incoming packets. To preclude this case in the protocol model, we require that a node duty cycles a positive amount of time per second.

$$t_{cycle} \geq 0 \tag{3.55}$$

3.3 Implementation

We use the ECL^{i}PS^e constraint programming system [2, 89] to implement the protocol model of a MAC protocol and the optimization component.[2] ECL^{i}PS^e realizes the paradigm of constraint logic programming and is based on Prolog. It consists of a runtime core, a rich set of libraries, and a modelling and control language. Furthermore, ECL^{i}PS^e provides interfaces to C/C++, Java, and Tcl/Tk, and interacts with the host environment via files, pipes, queues, or sockets. These features make ECL^{i}PS^e a suitable runtime platform for protocol model and optimization component that runs on the base station and communicates with the sensor network (see Figure 3.2).

[2] ECL^{i}PS^e is open source and available at `http://www.eclipse-clp.org`.

```
/* Per-hop reliability */
generateReliabilityCost(Cost, Vars, Scale) :-
    declareConstants(Consts),
    declareParameters(Params),
    Consts = [_,_,_,_,_,T_dtx,T_drx,T_str,_,_,T_sl,_],
    Params = [_,P_str,P_ack,P_data],
    Vars = [T_rl,_,N_t],
    P_rx_data is P_ack*P_data,
    T_iter is T_dtx + T_drx + T_str + T_sl,
    P_one_strobe $= P_str + (1.0 - P_str)*P_2nd_strobe*P_str,
    P_2nd_strobe $= (Scale*T_rl - T_iter - T_str)/(T_iter),
    Cost $= 1.0 - (1.0 - P_one_strobe*P_rx_data)^N_t.

/* Per-hop lifetime */
generateLifetimeCost(Cost, Vars, Scale) :-
    ...

/* Per-hop latency */
generateLatencyCost(Cost, Vars, Scale) :-
    ...
```

Figure 3.11: Performance metrics are implemented as separate predicates.

3.3.1 Implementation of Protocol Model

The protocol model may involve real numbers and non-linear expressions. Therefore, we use the Interval Constraint (IC) library of ECL^{i}PS^e, a hybrid integer and real interval constraint solver that handles non-linear constraints. Using built-in IC library predicates, we encode each performance metric as a separate Prolog predicate, as shown in Figure 3.11.

For example, we can determine the node lifetime in seconds for a given set of MAC protocol parameters by querying the `generateLifetimeCost` predicate.

```
[eclipse 2]: generateLifetimeCost(Lifetime, [10.29e-3,0.58656,1], 1).

Lifetime = 5707963.2580824653__5707963.2580824839
```

Because IC uses interval arithmetic with numbers represented by two floating bounds, ECL^{i}PS^e returns the node lifetime in `Cost` and prints its lower and upper bound. The true node lifetime is definitely known to lie between these two bounds.

The protocol parameters are passed as a list (`Vars`) to the predicates that encode the performance metrics. The constant input parameters of the protocol model, such as the radio's transmit bit rate and the battery capacity, are defined by the predicate `declareConstants`. The network properties, traffic volume and packet loss rates, are defined by the predicate `declareParameters`. In a running system, `declareParameters` retrieves the current network parameters from the data collection protocol, and the performance predicates pass their results to the control application (on the base station) that checks whether the application requirements

```
/* Declaration of goals (application requirements) */
generateGoals(Costs, Goals) :-
    Costs = [T_l,P,L],
    Goals = [GT_l,GP,GL],
    GT_l is 100*24*3600,  /* 100 days node lifetime */
    GP is 0.9,            /* 90 % per-hop reliability */
    GL is 0.5,            /* 500 ms per-hop latency */
    T_l $>= GT_l,
    P $>= GP,
    L $=< GL.

/* Optimization using goal programming */
goalProgramming(Vars) :-
    generateConstraints(Vars, Scale),
    generateCosts(Costs, Vars, Scale),
    generateGoals(Costs, Goals),
    Vars = [T_rl,T_rs,N_r],
    Costs = [T_l,P,L],
    Goals = [GT_l,GP,GL],
    Cost $= (T_l - GT_l)/GT_l + (P - GP)/GP + (GL - L)/GL,
    bb_min(
        search(Vars,0,most_constrained,indomain_split,complete,[]),
        (Cost),
        bb_options{strategy:continue,delta:0.0001,factor:1}
    ).
```

Figure 3.12: Core predicates of the optimization component. Predicate `generateGoals` defines the goals (application requirements) as lower and upper bounds on the objectives and posts three corresponding constraints. Predicate `goalProgramming` defines the objective function and performs branch and bound minimization on the decision variables.

are satisfied. Afterwards, if the application requirements are not satisfied, the control application triggers the execution of the optimization component.

3.3.2 Implementation of Optimization Component

The centerpieces of the optimization component, the predicates `generateGoals` and `goalProgramming`, are shown in Figure 3.12. Predicate `generateGoals` defines the application requirements—lower and upper bounds on the objectives node lifetime, per-hop latency, and per-hop reliability—and posts for each objective a corresponding inequality constraint. In this way, the application requirements translate into goals that we want to achieve for each objective. Predicate `goalProgramming` defines the objective function in `Cost` and queries the library predicate `bb_min`. `bb_min` performs branch and bound minimization. It computes, using a complete search on the decision variables, a solution to the optimization problem for which the value of the objective function is minimal. As a result, `Vars` contains the optimal parameters of the MAC protocol.

We can control the optimization process by selecting appropriate `bb_options`, the third argument of `bb_min`. For example, `delta` and `factor` are concerned with the improvement of the cost of each new solution, and `strategy` specifies the strategy used in the branch and bound search. These settings are important because they affect the completeness of the search and the time needed to find the optimal solution.

Additionally, we employ *discretization*, that is, the decision variables have integer domains instead of real domains. This is motivated by the following observations. First, optimization over finite domains is much faster. Second, the benefit of continuous domains, namely very high precision, is impractical. The highest reasonable precision for the timing parameters of a MAC protocol is defined by the resolution of the hardware timers. For example, the MSP430 microprocessor features a clock that oscillates with a 32,786-Hz watch crystal, and the resulting timer resolution is about 30 μs. It is therefore sufficient to calculate optimal solutions with a precision of 0.00003 s. If, for example, the maximal plausible value of a decision variable is 10 s, we define [0..333333] as its finite integer domain.

Chapter 4

Evaluation

This chapter quantitatively evaluates the effectiveness of our optimization and adaptation approach. We first validate the accuracy of the analytical model presented in Section 3.2.3 through a series of experiments. Based on this model, we explore the performance trade-offs in X-MAC using methods from multiobjective optimization. Then we show that our optimization approach determines optimal protocol parameters for different application requirements and characteristics. Finally, we demonstrate its capability to adapt the parameters of a MAC protocol to dynamic changes in network properties.

We use Tmote Sky sensor nodes [68] as the experimental platform. Each Tmote Sky has a 8 MHz MSP430 microcontroller (10 kB RAM and 48 kB flash), 1024 kB external flash memory, and a CC2420 low power 2.4 GHz radio [81] with a transmit bit rate of 250 Kbps. We run the Contiki operating system [20] on the nodes. Contiki is implemented in the C programming language and comes with an implementation of X-MAC and a communication stack [22] that provides a set of basic communication primitives, such as best-effort single-hop unicast. We use the unicast primitive on top of a slightly modified version of the X-MAC implementation and implement application-level code in Contiki to instrument the Tmote Sky nodes in our experiments. We assume a 20-byte application payload for data packets. We take all other values needed in the experiments, such as the power consumption of the CC2420 radio, from the datasheets or explicitly mention them in the text.

We validate the performance metrics of our analytical model of X-MAC in Section 4.1.3 using the Cooja/MSPSim simulator. MSPSim [26] is a Java-based instruction level simulator. It emulates complete sensor node platforms, such as the Tmote Sky, and executes the same code that can be deployed in real sensor networks. MSPSim provides detailed simulation with accurate timing, which enables test and evaluation of timing-sensitive software such as MAC protocols. Cooja [104] is a Java-based network simulator for simulating complete networks of sensor nodes. Cooja integrates MSPSim for executing Contiki programs compiled for MSP430-based sensor node platforms

4.1 Validation of Protocol Model

We take a bottom up approach to validate the accuracy of the protocol mode of X-MAC. Starting with transmission time measurements at the physical layer, we proceed by validating the duty cycle energy—an essential component of the node lifetime model—and finally analyze in detail the accuracy of the performance metrics with respect to the network properties.

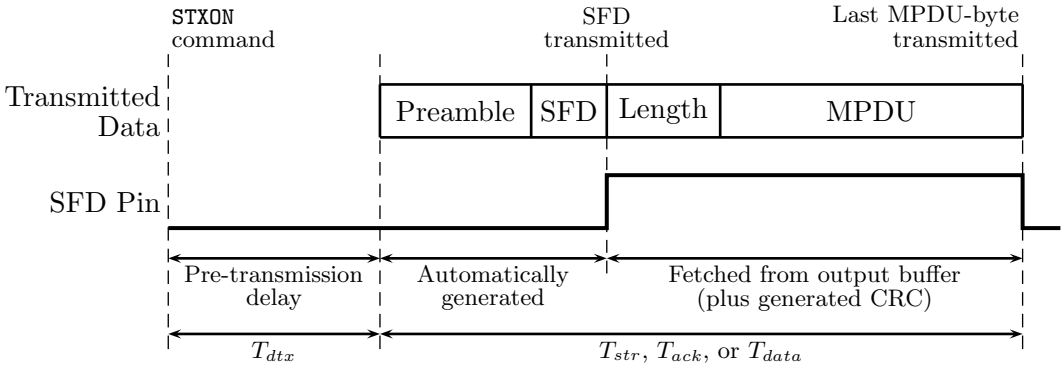


Figure 4.1: Start Frame Delimiter (SFD) pin activity during transmission. The SFD pin is active while frame length field and MAC Protocol Data Unit (MPDU) are transmitted. The corresponding terms in the protocol model of X-MAC are indicated below.

4.1.1 Transmission Times

The goal of this experiment is to study how well the theoretical length of a packet transmission conforms to the length observed in practice. Given packet size in bytes, S, and transmit bit rate, R, the protocol model calculates the transmission times of strobe, acknowledgment, and data packet as $8 \cdot \frac{S}{R}$. Therefore, this experiment indicates whether these times (T_{str}, T_{ack}, T_{data}) are properly represented in the protocol model. Additionally, we validate the length of the pre-transmission delay T_{dtx}, which amounts to 12 symbol (6 byte) periods in theory.

Setup and Method

We connect a Tmote Sky sensor node via USB to a PC. The node periodically sends out messages and we take time measurements at the CC2420 physical layer.

We use the Start Frame Delimiter (SFD) pin of the CC2420 radio to determine start and end of a data transmission. As shown in Figure 4.1, the SFD pin goes active when the SFD field of a frame has been completely transmitted. It goes inactive again when the complete MAC Protocol Data Unit (MPDU) has been transmitted, or if an underflow is detected. The SFD pin of the CC2420 radio is connected to the Serial Peripheral Interface Bus (SPI) of the MSP430 microprocessor, which can be accessed through C library routines. Using these routines we determine the rising and falling edges of the SFD pin during transmission of a frame.

Furthermore, we use the second 16-bit timer of the MSP430 microprocessor, called TimerB, to obtain accurate timestamps.[1] We configure TimerB so that it is sourced from the auxiliary clock which oscillates with a 32,786-Hz watch crystal.

Figure 4.2 shows how we record the SFD pin activity at the physical layer. The `TBR` routine returns the current value of TimerB. We also record when the `STXON` command is issued to enable transmission. This allows us to analyze also the pre-transmission delay and the time needed to transmit the automatically generated preamble and SFD fields (see Figure 4.1). Transmission is triggered by an application that periodically sends messages with a fixed-size application payload. We vary

[1] The first timer, TimerA, is already used in Contiki to schedule the execution of protothreads [21], for example.

```
int cc2420_send(const void * payload, unsigned short payload_len)
{
  ...                          /* Preprocessing */

  strobe(CC2420_STXON);        /* Enable transmission */
  log("STXON issued", TBR);

  while(!SFD_IS_1);            /* Wait until SFD pin goes active */
  log("SFD active", TBR);

  while(SFD_IS_1);             /* Wait until SFD pin goes inactive */
  log("SFD inactive", TBR);

  ...                          /* Postprocessing */
}
```

Figure 4.2: Measuring packet transmission time at the physical layer. The `TBR` routine returns the current value of the hardware timer.

the size of the application payload at compile time to take time measurements for different frame sizes.

Results and Discussion

We run the experiment for five different frame sizes: 14, 40, 70, 100, and 133 bytes. A 14-byte frame corresponds to the length of a X-MAC strobe packet, and 133 bytes is the maximum length of a frame. For each frame size we take 100 measurements and compute the average duration of the SFD pin activity.

Table 4.1: Duration of SFD pin activity. For varying frame size (in bytes), theoretical and measured SFD pin activity (in ms) is listed as well as the absolute error (in ms).

Frame Size	SFD Pin Activity		Absolute
Total (Length+MPDU)	Theoretical	Measured	Error
14 (9)	0.288	0.295	0.007
40 (35)	1.120	1.117	-0.083
70 (65)	2.080	2.077	-0.003
100 (95)	3.040	3.031	-0.009
133 (128)	4.096	4.106	0.010

Table 4.1 lists the duration of SFD pin activity for different packet sizes. Note that the SFD pin is active while frame length field and MPDU are transmitted. We see that theoretical and measured times correspond very well; the average relative error is less than 1 %. Also, our measurements reveal that the duration between the `STXON` command and the rising edge of the SFD pin is as expected. In theory this takes about 352 μs, and the measured values range from 345 μs to 356 μs. We therefore conclude that the pre-transmission delay (T_{dtx}) and the transmission

times of strobe, acknowledgment, and data packets (T_{str}, T_{ack}, T_{data}) are properly represented by their theoretical estimates in the protocol model.

4.1.2 Duty Cycle Energy

The baseline lifetime of a node is determined by the power consumption in the sleep state and the duty cycle. In the protocol model of X-MAC, these fundamental parameters are captured by the duty cycle energy. We want to validate the accuracy of the duty cycle energy, which is part of the node lifetime performance metric.

Setup and Method

We connect a Tmote Sky sensor node via USB to a PC. The node runs Contiki with X-MAC as the default MAC protocol. The node neither receives nor transmits any messages; it simply switches the radio on and off at a duty cycle of 1 %. The listen time is 5 ms, and the sleep time is 495 ms.

We measure the node's power consumption over a period of ten minutes using Contiki's built-in software-based power profiler [23]. The power profiler records the times that the radio spends in different states. This method has been shown to provide a good measure of the total energy consumption of the device [23].

Results and Discussion

We assume batteries that supply 2000 mAh at 3 volts. By plugging the measured power consumption in (3.2), we obtain a node lifetime of 136.1 days. To obtain the theoretical node lifetime, we use the implementation of the protocol model in ECL^{i}PS^e and query the `generateLifetimeCost` predicate with appropriate parameters, as demonstrated in Section 3.3.1. The protocol model calculates a theoretical node lifetime of 135.7 days.

The relative error is 0.3 %. We conclude that the duty cycle energy is accurately represented in the protocol model of X-MAC.

4.1.3 Performance Metrics

Our system for MAC protocol adaptation operates a control loop (see Figure 3.1). In each control loop iteration, the base station uses the protocol model to decide whether the application requirements are satisfied or new parameters have to be computed by the optimization component. The current network properties collected from the sensor nodes are the input parameters of the protocol model. To ensure the base station makes the right decisions and the optimization component computes appropriate parameters, we have to validate the accuracy of the performance metrics with respect to the network properties.

Setup and Method

We deliberately use a small-scale setup to have tight control over packet loss rate and traffic volume. Unpredictable packet loss in a large-scale setup would make it impossible for us to fairly validate the protocol model. We simulate three Tmote

Sky nodes in Cooja and run the Contiki operating system on them.[2] As shown in Figure 4.3, the first node sends data packets to the second node which retransmits any received packets to the third node. The duration of the experiments is one hour.

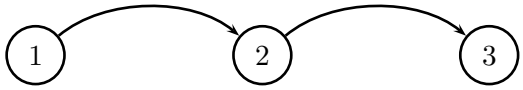


Figure 4.3: Experimental setup to validate the performance metrics.

X-MAC is configured to have a duty cycle of approximately 1.5 %. The receiver listen time is 8 ms, and the receiver sleep time is 495 ms. The maximal length of the strobe sequence is 511 ms. The sender listen period is 4 ms. Each packet is transmitted exactly once, that is, lost packets are not retransmitted. We base node lifetime calculations on the assumption that the batteries supply 2000 mAh at 3 volts. We control the network parameters—packet loss rate and traffic volume—and take measurements as described in the following paragraphs.

Packet Loss Rate We implement packet loss by controlling packet reception. A node drops incoming packets according to a Bernoulli process with probability of success p. For example, if $p = 0.9$, a node drops on average one out of ten incoming packets, which gives a packet reception rate of 90 %. We vary the packet reception rate of strobe, acknowledgment, and data packets between 60 % and 100 % in steps of 10 %. In a first series of experiments, we fix the reception rates for each two packet types (for example, acknowledgment and data packets) at 100 % and vary the reception rate of the third packet type (for example, strobe packet). This allows us to study the accuracy of the protocol model with respect to the packet reception rate on a per packet type basis. In a second series of experiments, we vary the reception rates for all packet types simultaneously. This is much closer to a realistic environment, where the reception rates are typically identical for all packets.

Traffic Volume In all experiments, the first node sends data packets according to a homogeneous Poisson process of rate 0.1, that is, on average one packet in ten seconds. Considering the traffic volumes of real-world applications that fall into the scope of X-MAC, such as environmental monitoring, one packet in ten seconds is a rather high traffic volume. For example, [59] and [84] report traffic volumes between one packet every five minutes and one packet every hour. We do not run experiments with lower traffic volumes because of the following reasons. First, the performance metrics per-hop latency and per-hop reliability are independent of the traffic volume. Second, we know already that the relative error in node lifetime decreases with lower traffic volumes. For lower traffic volumes, the overall energy consumption of a node is dominated by the duty cycle energy, which is accurately represented in the protocol model of X-MAC, as demonstrated in Section 4.1.2.

Measurements To validate node lifetime, we measure the power consumption of the second node using Contiki's built-in power profiler. To validate per-hop

[2] We ran part of the validation experiments also on real sensor nodes, leading essentially to the same results. To preclude even interference with other wireless communication, we decided to carry out all validation experiments in Cooja and to report these results in the thesis.

reliability, we measure the reception rate of *data packets* at the second node. To validate per-hop latency, we measure the time from the start of a unicast transmission at the first node to the reception of the data packet at the second node. We compute the average per-hop latency of *received* packets.

Results and Discussion

Figures 4.4(a) to 4.4(c) plot measured and theoretical **per-hop reliability** for different packet reception rates. The packet reception rate varies between 60 % and 100 % for *one* packet type, while it is 100 % for the other two packet types. As expected, the per-hop reliability decreases as the packet reception rate decreases.

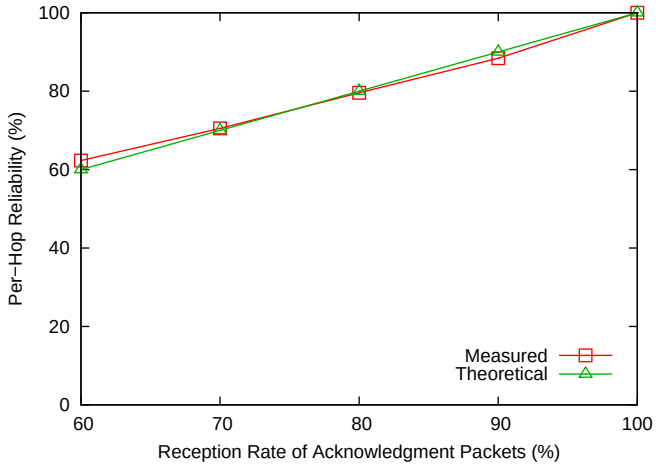
We see that the theoretical per-hop reliability corresponds very well with the measured per-hop reliability. When varying the reception rate of acknowledgment packets (see Figure 4.4(a)) and data packets (see Figure 4.4(b)), the largest deviations are 2.3 % and −0.6 %. Looking at Figure 4.4(c), we note a slight gap between the two curves for different reception rates of strobe packets. This results from the small time scale in the timing of strobe packets. To illustrate this, consider the probability that a node is able to hear two strobes within a listen period (see Section 3.2.3, Figure 3.10). If the transmission of the first strobe fails, the intended receiver may be able to receive the following strobe. This probability is crucial for per-hop reliability and depends on small-scale changes in the length of the listen period. Our protocol model computes with 10 μs precision. But the implementation of X-MAC in Contiki schedules listen periods using a timer with 250 μs resolution. This is very coarse, given that it takes about 450 μs to transmit a strobe. The difference in precision between implementation and protocol model explains the difference between measured and theoretical per-hop reliability for strobe packets.

Figures 4.5(a) to 4.5(c) show measured and theoretical **node lifetime** for different packet reception rates. Again, the packet reception rate varies between 60 % and 100 % for one packet type, while it is 100 % for the other two packet types. The node lifetime increases as the packet reception rate decreases. This behavior results from the fact that lost packets are not retransmitted. Thus, the more packets are lost, the fewer packets a node must handle. This saves energy and leads to an extended node lifetime.

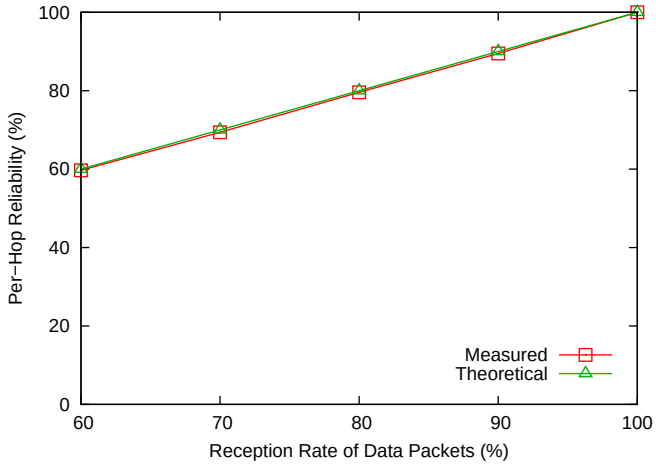
We observe that the theoretical node lifetime corresponds well with the measured node lifetime. The maximal deviation ranges between 1 day and 2.5 days when varying the reception rates of data and acknowledgment packets. For strobe packets, we note again a slightly larger deviation of maximal 5 days.

We now turn to a more realistic setting, where the reception rates are the *same* for strobe, acknowledgment, and data packets. Figure 4.6(a) plots per-hop reliability and Figure 4.6(b) plots node lifetime for different packet reception rates. For both performance metrics, theoretical and measured values correspond very well. In fact, the slight inaccuracy with regard to strobe packets is compensated for by the dominance of acknowledgment and data packets. Overall, we observe less than 4 % deviation in per-hop reliability and less than 2.5 days deviation in node lifetime.

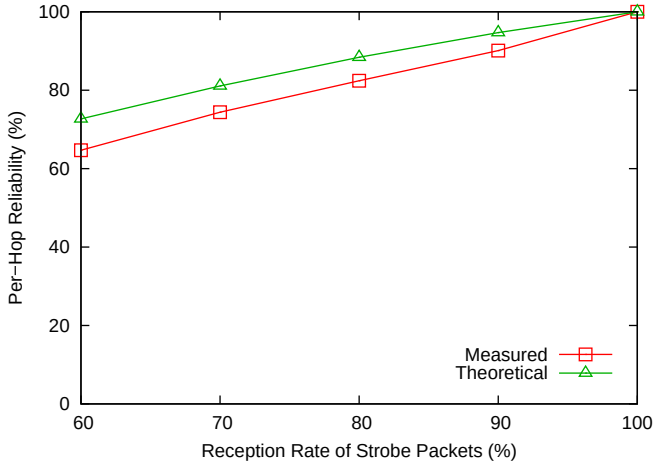
The theoretical **per-hop latency** corresponds up to 1 ms with the measured per-hop latency for 100 % packet reception rate. If packets are lost, we are only able to measure per-hop latency for received packets, because we miss the time of reception for lost packets. Therefore, we have to omit lost packets when computing the average, measured per-hop latency. But the theoretical per-hop latency in the protocol model does *not* omit lost packets. Instead, the theoretical per-hop



(a) Varying reception rate of acknowledgment packets.

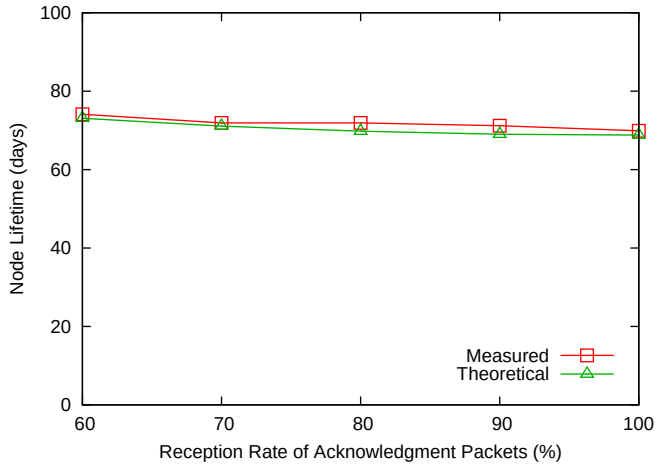


(b) Varying reception rate of data packets.

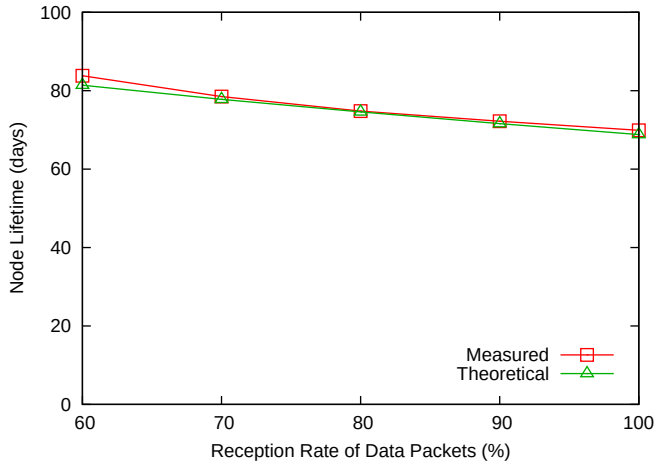


(c) Varying reception rate of strobe packets.

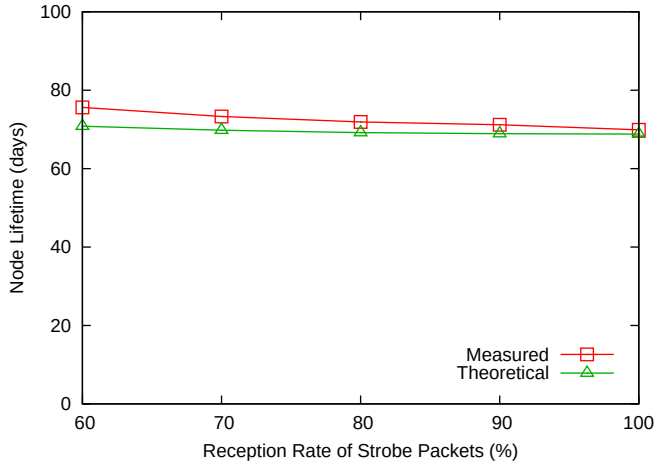
Figure 4.4: Measured and theoretical per-hop reliability. The packet reception rate varies between 60 % and 100 % for one packet type, while it is 100 % for the other two packet types.



(a) Varying reception rate of acknowledgment packets.

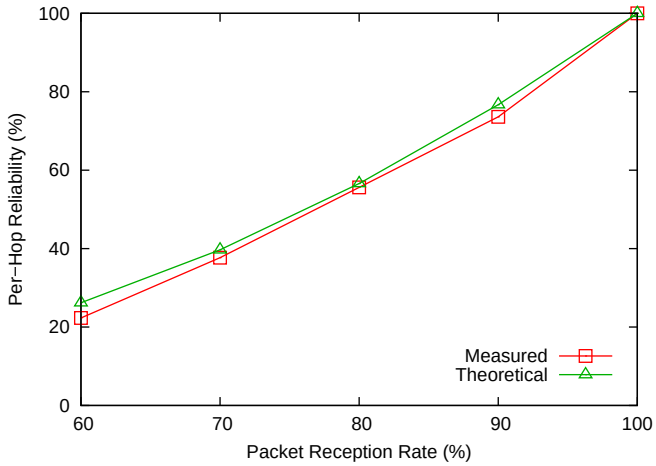


(b) Varying reception rate of data packets.

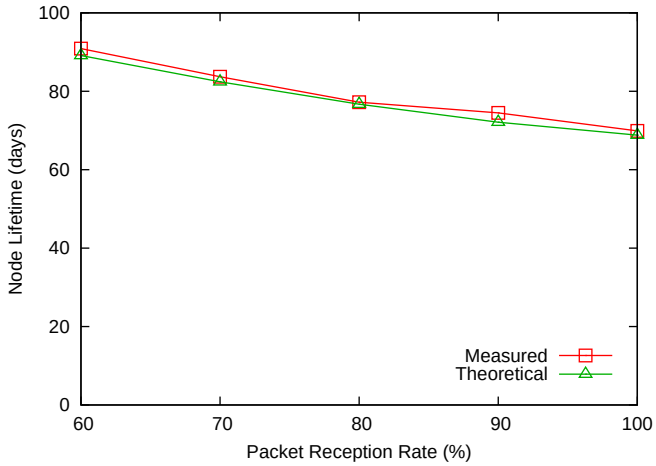


(c) Varying reception rate of strobe packets.

Figure 4.5: Measured and theoretical node lifetime. The packet reception rate varies between 60 % and 100 % for one packet type, while it is 100 % for the other two packet types.



(a) Per-hop reliability.



(b) Node lifetime.

Figure 4.6: Measured and theoretical performance. The packet reception rate varies between 60 % and 100 % for all packet types simultaneously.

latency assumes that every packet is eventually delivered after a finite number of transmissions. This leads inevitably to an error between measured and theoretical per-hop latency. The error grows with lower packet reception rates, because the number of lost packets, for which we miss the time of reception, increases. We could set the number of transmissions per packet artificially high to guarantee reception of each packet. Theoretically, this requires sending each packet infinitely many times. Practically, we have to send each packet 24 times to guarantee delivery with 99.99 % probability for 70 % packet reception rate. We refrain from such an approach and note that per-hop latency is indeed accurate for 100 % packet reception rate, and moreover, the model of per-hop latency builds upon expressions from the per-hop reliability and node lifetime models, which are known to be accurate for varying packet reception rates.

Table 4.2: Optimal values of the objectives for different packet reception rates.

Reception Rate (%)	Node Lifetime (days)	Per-Hop Reliability (%)	Per-Hop Latency (ms)
100	**121.9**	100.0	407.9
	4.2	**100.0**	4.2
	4.2	100.0	**4.2**
90	**133.5**	72.9	589.3
	4.2	**99.9**	7.4
	4.2	72.9	**4.7**
70	**136.2**	34.3	2497.3
	4.2	**99.9**	26.5
	4.2	34.3	**22.1**

4.2 Configuration Space Exploration

Intuitively speaking, finding an optimal set of parameters (or configuration) for a MAC protocol means finding a good trade-off among the conflicting objectives node lifetime, per-hop reliability, and per-hop latency. A graphical representation of the good trade-offs is the Pareto front (see Section 2.2.1). For an optimization problem with three objectives, the Pareto front is some shape in three-dimensional space. Each point of that shape corresponds to a good trade-off, that is, to a Pareto optimal vector of decision variables. In our case, each Pareto optimal vector is an optimal configuration of the MAC protocol. Generating the Pareto front can be considered exploring the configuration space of the MAC protocol and memorizing the optimal configurations.

The shape of the Pareto front reveals how the MAC protocol realizes the various performance trade-offs. This provides valuable feedback for the protocol designer who may want to modify the protocol to emphasize a specific trade-off. More importantly, it allows potential users of the MAC protocol to assess whether that protocol is suitable for their sensor network application. We generate the Pareto front of X-MAC and discuss some insights that we gain from its shape.

4.2.1 Generating the Pareto Front

We use the weighted sum approach (see Section 2.2.2) and the protocol model to generate 600 points of the Pareto front of X-MAC. Using the constraint programming system ECL^{i}PS^e, we solve the optimization problem

$$\min \left(w_1 \frac{\hat{T}_l}{T_l(\mathbf{x})} + w_2 \frac{\hat{P}}{P(\mathbf{x})} + w_3 \frac{L(\mathbf{x})}{\hat{L}} \right) \tag{4.1}$$

for 600 different combinations of the weighting coefficients w_i, where $0 \leq w_i \leq 1$ and $\sum_i w_1 = 1$. The terms $\hat{T}_l$, $\hat{P}$, and $\hat{L}$ denote the maximum node lifetime, the maximum per-hop reliability, and the minimal per-hop latency when optimizing each objective separately. Table 4.2 displays these optimal values (in boldface) for different packet reception rates. For example, at 90 % packet reception rate the maximal node lifetime is 133.5 days, while at the same time per-hop reliability

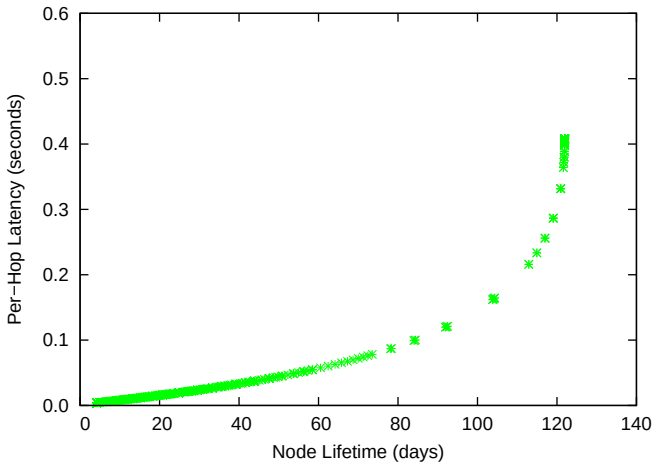


Figure 4.7: Trade-off between node lifetime and per-hop latency in X-MAC for 100 % packet reception rate and a traffic volume of one packet per minute. In this case, the per-hop reliability is 100 % for all feasible configurations of X-MAC.

is 72.9 % and per-hop latency is 589.3 ms. We use these values to scale the objective functions so that the weights reflect proportionally the importance of the objectives.

Each generated point of the Pareto front corresponds to a triple

$$[T_l(\mathbf{x}^*), P(\mathbf{x}^*), L(\mathbf{x}^*)] \tag{4.2}$$

of values of the objective functions, where $\mathbf{x}^*$ is an optimal configuration of X-MAC.

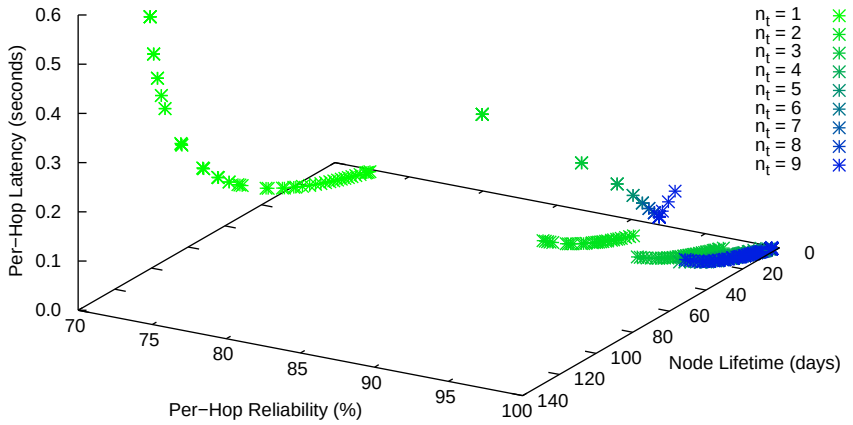
We have to select network properties for which we want to generate the Pareto front. We generate the Pareto front for 70 %, 90 %, and 100 % packet reception rate. We choose a traffic volume of one packet per minute, which is a typical average in low traffic applications that may consider using X-MAC, such as environmental monitoring [59, 83].

4.2.2 Performance Trade-Offs

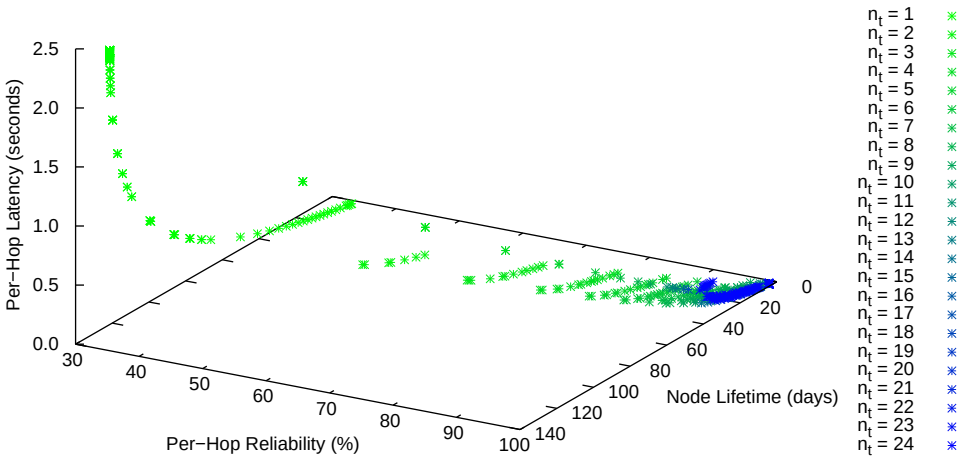
If the packet reception rate is 100 %, no packets are lost and the per-hop reliability is 100 % for all feasible configurations of X-MAC. In particular, it suffices to transmit each packet exactly once. Consequently, there is only a trade-off between node lifetime and per-hop latency; the Pareto front is essentially reduced to a shape in two-dimensional space. Figure 4.7 shows this trade-off.

The general trend is that the we have to accept longer per-hop latency if we want to extend node lifetime, or conversely, shorter node lifetime if we want to decrease per-hop latency. For example, the maximal possible node lifetime is 80 days if our application requires per-hop latency to be shorter than 0.1 seconds; there exists no feasible configuration of X-MAC to achieve a longer node lifetime under this latency constraint. This kind of information is useful to decide whether X-MAC can satisfy the requirements of an application.

Furthermore, we observe that the closer we get to the extreme of one objective, the more we have to give up on the other objective. For example, to extend node lifetime from 40 days to 60 days costs only 25 ms more in per-hop latency, but to extend from 100 days to 120 days costs 170 ms—a sevenfold increase in added per-hop latency to achieve the same 20-day extension in node lifetime.

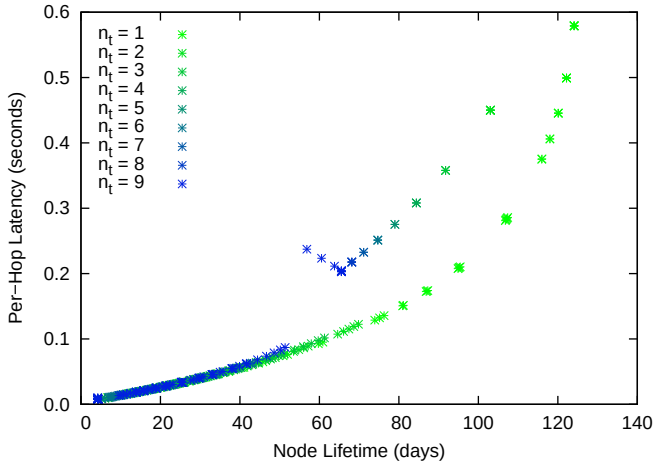


(a) For 90 % packet reception rate.

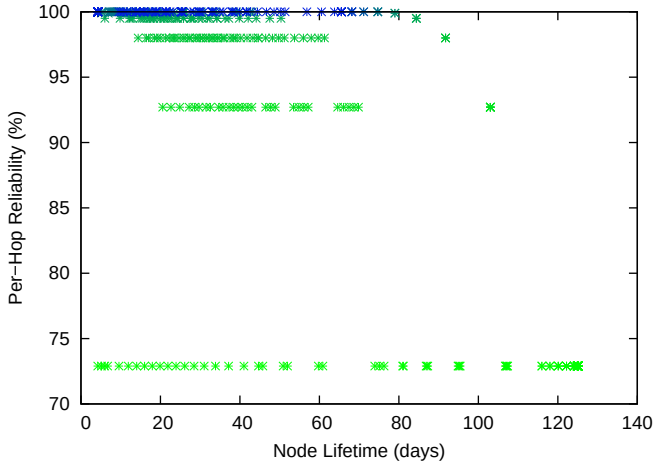


(b) For 70 % packet reception rate.

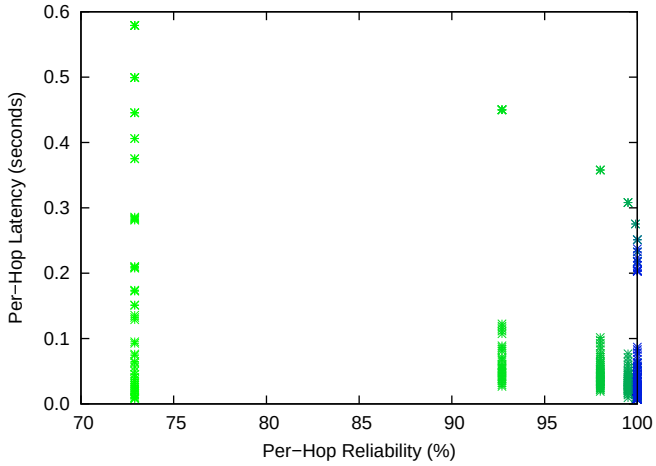
Figure 4.8: Pareto front of X-MAC for a traffic volume of one packet per minute. Points are colored depending on the number of transmissions per packet, n_t. Note the different scaling with respect to per-hop reliability and per-hop latency.



(a) Trade-off between node lifetime and per-hop latency.

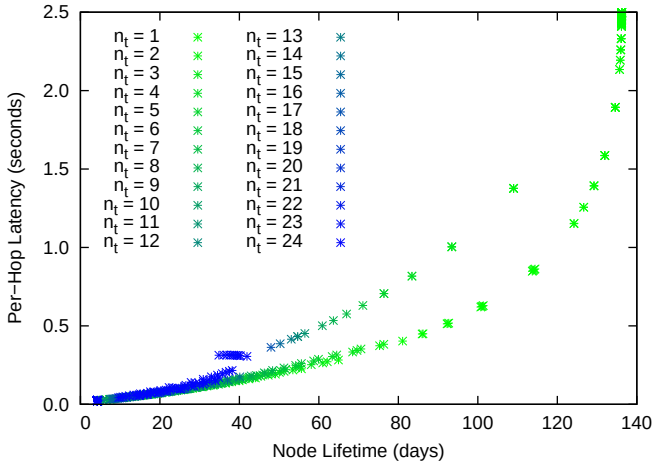


(b) Trade-off between node lifetime and per-hop reliability.

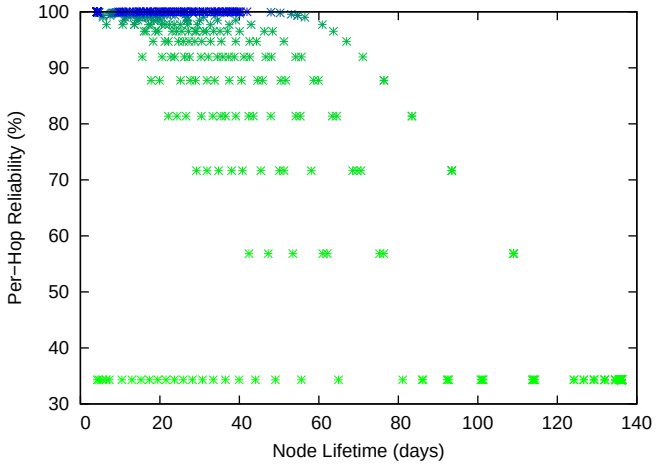


(c) Trade-off between per-hop reliability and per-hop latency.

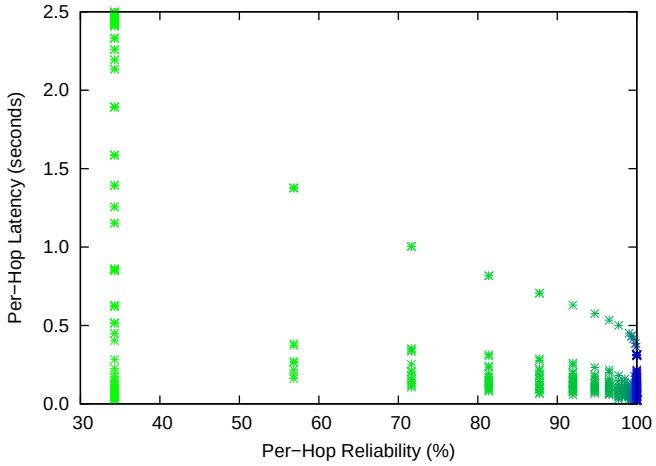
Figure 4.9: Two-dimensional projections of the Pareto front of X-MAC for 90 % packet reception rate and a traffic volume of one packet per minute. Points are colored depending on the number of transmissions per packet, n_t. Note the different scaling compared to Figure 4.10.



(a) Trade-off between node lifetime and per-hop latency.



(b) Trade-off between node lifetime and per-hop reliability.



(c) Trade-off between per-hop reliability and per-hop latency.

Figure 4.10: Two-dimensional projections of the Pareto front of X-MAC for 70 % packet reception rate and a traffic volume of one packet per minute. Points are colored depending on the number of transmissions per packet, n_t. Note the different scaling compared to Figure 4.9.

Figure 4.8 shows the Pareto front of X-MAC for 90 % and 70 % packet reception rate. To be able to better discern the performance trade-offs, we also plot the two-dimensional projections of the Pareto front in Figures 4.9 and 4.10. Each point of the Pareto front corresponds to an optimal configuration of X-MAC, which comprises optimal values for the protocol parameters considered as decision variables: receiver listen time, receiver sleep time, and number of transmissions per packet. The points are colored differently depending on the number of transmissions per packet, n_t. For 90 % packet reception rate, n_t ranges between 1 and 9; for 70 % between 1 and 24.

We refer to the graphs for 90 % packet reception rate (Figures 4.8(a) and 4.9) throughout the discussion, but our observations apply likewise to those for 70 %.

Most conspicuous is the staggered shape of the Pareto front with respect to per-hop reliability. Looking at Figures 4.9(b) and 4.9(c), we identify five lines. All points on a line have the same color and thus relate to the same n_t. In fact, there are nine lines, one for each n_t. The first line is at per-hop reliability 72.9 %, the second at 92.6 %, the third at 98.0 %, and the fourth at 99.4 %. For $n_t \geq 5$, the lines are (almost) at per-hop reliability 100 %. We can explain this behavior by looking at the receiver listen times determined by the optimization: the receiver listen time is almost always equal to its lower bound (see Section 3.2.3). As a result, the expression for calculating the per-hop reliability simplifies to

$$P = 1 - [1 - P_{str} \times P_{ack} \times P_{data}]^{n_t} = 1 - 0.729^{n_t}, \tag{4.3}$$

given that $P_{str} = P_{ack} = P_{data} = 0.9$ for a 90 % packet reception rate. Using (4.3) we get the per-hop reliability of each line. We can conclude the following from this observation: the number of transmissions per packet dominates per-hop reliability in the optimal configurations. The improvement in per-hop reliability is much larger by sending a packet more often than by letting receivers listen longer.

The trade-off between node lifetime and per-hop latency (see Figure 4.9(a)) is similar to the one for 100 % packet reception rate. However, we now have to accept longer per-hop latency to achieve the same node lifetime, because it takes longer to delivery a packet due to packet loss. Furthermore, we see in Figure 4.9(b) that X-MAC trades a higher per-hop reliability for a shorter node lifetime. For example, if we require 90 % per-hop reliability, we have to send each packet at least twice and the maximal node lifetime is 100 days. Finally, we learn from Figure 4.9(c) that higher per-hop reliability leads to shorter per-hop latency.

4.3 Pre-Deployment Configuration

In this section we demonstrate that our optimization approach computes optimal MAC protocol parameters for different network properties and application requirements. To this end, we first identify for three typical classes of sensor network applications—environmental monitoring, structural health monitoring, and target tracking—the arising requirements on a MAC protocol. Given different sets of typical requirements and network properties of each class, we use our implementation to compute optimal parameters of X-MAC. Afterwards, we validate the optimization results through experiments on real sensor nodes. Knowing that X-MAC is only applicable to low traffic applications, we use three sets of typical requirements and properties of environmental monitoring. The computed parameters can be used, for example, for configuring the sensor nodes prior to network deployment.

Table 4.3: Importance of MAC protocol requirements for different application classes.

Application Class	Node Lifetime	Per-Hop Latency	Per-Hop Reliability
Environmental monitoring	High	Low	Medium
Structural monitoring	Low–Medium	Low	High
Target tracking	Low	High	High

4.3.1 Requirements of Different Application Classes

Table 4.3 displays the importance of the MAC protocol requirements for the application classes as motivated in the following.

Environmental Monitoring The ultimate goal of environmental monitoring is long-term data collection [59, 60, 84]. The sensor network must run for months or years, possibly from non-rechargeable power sources. Real-world deployments at a redwood tree [84] and on Great Duck Island [59, 79] report observation periods of 44 days and 4 months. Low duty cycle operation and use of batteries with large energy densities [60] are ways of achieving such longevity requirements. To allow for meaningful offline data analysis, long latency is preferable to data loss [59]. Sampling periods typically vary between a few minutes and several hours, depending on the time-scale of change in the observed phenomena.

Structural Monitoring Structural health monitoring (SHM) assesses the structural integrity of bridges and buildings [14]. Most SHM systems [15, 34, 46, 93] use indirect detection, especially through vibration. This requires high frequency sampling (up to 1 kHz [46]), reliable command dissemination, and reliable data collection. In the Torre Aquila Deployment [11], for example, researchers aimed at an overall packet loss rate of less than 0.01 %. The volume of raw data is two to four orders of magnitude larger than that of an environmental monitoring application. Data compression and temporary buffering on source nodes and transmission pipelining along forwarding paths [16] are used to cope with such data volumes. For example, in Wisden [65, 93] aggregated data are transferred at a rate of two packets per seconds. Lifetime is less important in SHM because data acquisition takes only a few hours to a day [93] and high-capacity batteries, such as lantern batteries [46], are used.

Target Tracking Target tracking attempts to estimate the current positions of moving objects [3, 54, 56]. Contrary to periodic monitoring applications, target tracking is even-driven. The sensor network is idle most of the time, but when an object is detected the nodes wake up and engage in tracking the object. Tracking accuracy is the main performance requirement. Accordingly, nodes must be very synchronized and quickly transport location information to the base station. The system in [3], for example, tolerates up to 5 meters deviation for persons and vehicles, and the desired 50 % end-to-end reliability translates into 96 % per-hop reliability

Table 4.4: Network properties and application requirements of different application classes (upper part). The corresponding optimal parameters of X-MAC and theoretical performance are listed in the lower part.

Parameter	Environmental Monitoring	Structural Monitoring	Target Tracking
Network properties			
Traffic volume	1 pkt/5 min	6 pkts/s	2 pkts/s
Packet reception rate (%)	90	90	90
Application requirements			
Minimal node lifetime (days)	120	1	10
Minimal per-hop reliability (%)	80.00	99.99	96.00
Maximal per-hop latency (s)	5.0	1.0	0.5
Optimal protocol parameters			
Receiver listen time (ms)	10.3	5.4	10.3
Receiver sleep time (ms)	7186.9	30.9	157.7
Transmissions per packet	1	6	2
Theoretical performance			
Node lifetime (days)	127	4	10
Per-hop reliability (%)	80.01	99.99	96.00
Per-hop latency (s)	4.999	0.032	0.120

for the stated network layout.[3] Lifetime is no concern in target tracking since the system is only activated by (rare) events and node density is typically high.

4.3.2 Optimization for Different Application Classes

According to the preceding discussion, we identify concrete sets of typical network properties and application requirements for each application class, as shown in the upper part of Table 4.4. The numbers are either directly taken from the literature or derived from the application context. We use the implementation of the optimization component in ECL^{i}PS^e to determine optimal protocol parameters of X-MAC for each set of numbers. With respect to node lifetime, we assume that the batteries supply 2000 mAh at 3 volts.

The lower part of Table 4.4 displays the optimization results. These include the optimal parameters of X-MAC and its theoretical performance based on these parameters. The execution time of the optimization is on the order of a few seconds.

The theoretical performance satisfies the application requirements. As expected, the optimal receiver sleep time is long for environmental monitoring, whereas it is short for structural monitoring and target tracking. To fulfill the high per-hop reliability requirement of structural monitoring, each packet must be sent 6 times. A node dies after about four days if it has the radio turned on all the time. This explains the theoretical node lifetime of structural monitoring. The theoretical per-hop latencies of structural monitoring and target tracking are much shorter than the required maximal per-hop latencies. This is because the mild node lifetime

[3] According to Figure 5 in [3], each node is less than 15 hops away from the base station. This gives a per-hop reliability of approximately $(0.5)^{1/15} = 0.96$.

Table 4.5: Typical network properties and application requirements of environmental monitoring (upper part). The corresponding optimal parameters of X-MAC as well as theoretical and measured performance are listed in the lower part.

Parameter	Setting (1)	Setting (2)	Setting (3)
Network properties			
Traffic volume	1 pkt/30 s	1 pkt/min	1 pkt/5 min
Packet loss rate (%)	90	90	90
Application requirements			
Minimal node lifetime (days)	90	60	120
Minimal per-hop reliability (%)	80	90	80
Maximal per-hop latency (s)	2	5	5
Optimal protocol parameters			
Receiver listen time (ms)	10.3	5.4	10.3
Receiver sleep time (ms)	1170.7	2639.5	7186.9
Transmissions per packet	1	2	1
Theoretical performance			
Node lifetime (days)	90	60	127
Per-hop reliability (%)	80	93	80
Per-hop latency (s)	0.823	2.162	4.999
Measured performance			
Node lifetime (days)	92	59	130
Per-hop reliability (%)	76	91	75
Per-hop latency (s)	–	–	–

requirements permit substantial optimization of the per-hop latencies; the receiver sleep time can be made shorter than needed, which leads to low per-hop latency.

4.3.3 Validation of Optimization Results

We use three Tmote Sky sensor nodes to validate the optimization results. Experimental setup and measurement method are as described in Section 4.1.3. We consider typical network properties and application requirements of environmental monitoring, listed in the upper part of Table 4.5, because X-MAC is designed for this type of low traffic application. We configure X-MAC with the parameters computed by the optimization component and vary the duration of the experiment between 3 hours and 24 hours to cover at least 300 packet transmissions. We assume again batteries that supply 2000 mAh at 3 volts for node lifetime calculations.

The optimal protocol parameters of X-MAC as well as theoretical and measured performance are shown in the lower part of Table 4.5. With respect to node lifetime, we see that theoretical and measured values correspond very well and the application requirements are satisfied for Settings (1) and (3). Only for Setting (2) the measured node lifetime is 1 day shorter than required. Looking at per-hop reliability, the application requirement is satisfied in Setting (2), and the deviations between measured and theoretical values correspond to the deviations observed in Section 4.1.3. The inaccuracies for Settings (1) and (3) clearly result from the small-scale timing of strobe packets. In both cases, the receiver listen time is set to its upper bound (see Section 3.2.3), meaning that nodes should listen long enough to

hear two strobe packets. However, the probability that this happens is overestimated by the protocol model, which explains the gap between theoretical and measured per-hop reliability. To fix this inaccuracy, we have to take precise time measurements with an oscilloscope and use those in the protocol model. An additional argument to support this reasoning provides Setting (2). Here, the receiver listen time is set to its lower bound and thus nodes can hear only one strobe while listening. Packets are sent twice to achieve the required 90 % per-hop reliability, which is indeed the case as we measure 91 % per-hop reliability.

In summary, we can state that our optimization approach is capable of tuning the performance of X-MAC close the theoretical optimum, while satisfying the application requirements in most cases.

4.4 Runtime Adaptation

This section demonstrates the effectiveness of our proposed system to automatically adapt a MAC protocol to dynamic changes in traffic volume and packet loss rate. As described in Chapter 3, adaptation proceeds in a control loop: network properties are constantly collected to keep the base station up-to-date; optimization executes if the current performance is unsatisfactory; new protocol parameters are disseminated to the nodes which reconfigure their MAC protocol. To evaluate the adaptation procedure, we imitate changes in traffic volume and packet loss rate and run three control loop iterations using a network of three Tmote Sky sensor nodes.

4.4.1 Setup and Method

We use the same setup and measurement method as in Section 4.1.3. Traffic is injected according to a Poisson process; packet loss is controlled by letting nodes ignore incoming packets according to a Bernoulli process. We assume application requirements of 60 days minimal node lifetime, 90 % minimal per-hop reliability, 0.5 seconds maximal per-hop latency and batteries that supply 2000 mAh at 3 volts. The experiment proceeds as follows:

0. Initially, the traffic volume is 1 packet in 20 seconds and the packet reception rate is 95 %. Given these network properties and the application requirements, the optimization component determines optimal parameters for X-MAC. We program the sensor nodes accordingly and start the experiment. The nodes report packet reception events and power profiles via USB to the base station.

1. After 30 minutes we check whether the performance is satisfactory. To imitate varying network properties, we increase the traffic volume to 1 packet in 15 seconds and reduce the packet reception rate to 90 %. This correlation between data traffic and packet loss is typical in large-scale sensor networks [99].

2. After another 30 minutes we check whether the performance is still satisfactory. We expect that this is not the case. The optimization computes optimal parameters given the observed change in network properties. We adapt X-MAC by reprogramming the sensor nodes with the new parameters.

3. We check after 30 minutes whether the performance is again as desired.

By executing these steps, the experiment simulates three consecutive control loop iterations as they may proceed in a full-fledged implementation.

Table 4.6: Protocol parameters and performance for initial and changed network properties.

Parameter	Initial	Changed
Network properties		
Traffic volume	1 pkt/20 s	1 pkt/15 s
Packet reception rate (%)	95	90
Optimal protocol parameters		
Receiver listen time (ms)	6.5	5.4
Receiver sleep time (ms)	786.4	546.5
Transmissions per packet	2	2
Theoretical performance		
Node lifetime (days)	60.0	60.0
Per-hop reliability (%)	98.2	92.6
Per-hop latency (s)	0.499	0.453

4.4.2 Results and Discussion

We present our record of steps 1. to 3. of the experiment. As reference, Table 4.6 lists the optimal parameters of X-MAC and the theoretical performance for initial and changed network properties.

1. In the first iteration of the control loop the measured performance is as expected: node lifetime is 58.6 days and per-hop reliability is 97.8 %.[4] In the strict sense, the node lifetime requirement is not satisfied and X-MAC should be reconfigured. Such an aggressive approach induces high control overhead and can lead to system instability. Therefore, it is advisable to allow a margin of error for each requirement.

2. In the second iteration the measured node lifetime drops to 43.5 days; measured per-hop reliability is 92.7 %. The current configuration of X-MAC does no longer achieve satisfactory performance as node lifetime is about one third lower than required. Thus the optimization component determines new protocol parameters based on the network properties observed by the sensor nodes and X-MAC is reconfigured.[5]

3. In the third iteration the measured performance is again satisfactory: node lifetime is 57.5 days and per-hop reliability is 89.4 %.

This record shows that the protocol parameters of X-MAC are effectively adapted to the changes in traffic volume and packet loss rate in the second iteration of the control loop. Moreover, it took only 23 seconds for the optimization component to compute the new parameters, suggesting that our proposed system can react to changes in environment and application requirements in a timely manner.

[4] We do not report per-hop latency for the reasons explained in Section 4.1.3.

[5] For example, the middle node (see Figure 4.3) records 114 received packets (out of 123) within the 30-minute time window. This yields directly the traffic volume in the network. The protocol model also needs to know the probability of successfully receiving a packet, which is *not* equivalent to per-hop reliability. Using the (old) protocol parameters of X-MAC, it is possible to determine this probability, which is 89.7 % in this experiment. X-MAC is however adapted to a 90 % probability, an error of 0.3 %.

Chapter 5

Conclusions and Future Work

The main contribution of this thesis is a system for automatic parameter optimization and runtime adaptation of sensor network MAC protocols. A case study implementing the optimization approach demonstrates its capability to optimize multiple conflicting performance metrics of a MAC protocol and to explore the trade-offs among them. Moreover, experiments on a small-scale network of real sensor nodes substantiate that the proposed system adapts MAC protocol parameters timely and effectively to changes in network conditions. The system can be used in isolation or as part of a sensor network management system to ensure optimal performance and compliance with the application requirements.

Further experiments on a larger sensor testbed are necessary to demonstrate the scalability of our approach and to study how MAC protocol adaptation affects the overall behavior of a wireless sensor network. A full-fledged implementation entails also adding the collection and dissemination services needed to close the control loop between the sensor nodes and the base station.

Another possible direction of future research is the application of our optimization approach to other sensor network MAC protocols. In particular, it seems worthwhile to study protocols based on low power probing, such as RI-MAC [78] and the one proposed in [61]. This would allow us to compare low power listening with low power probing—the two major approaches of random channel access—studying their differences and similarities in parameter adaptation and performance trade-offs. Furthermore, the constraint programming approach taken in this thesis could unfold its full potential when applied to multi-channel MAC protocols, such as Y-MAC [47]. These protocols allocate available radio channels to communication links, a task which involves combinatorial optimization and could thus be solved efficiently with constraint programming.

Bibliography

[1] K. R. Apt, *Principles of Constraint Programming.* Cambridge University Press, 2003.

[2] K. R. Apt and M. G. Wallace, *Constraint Logic Programming using Eclipse.* Cambridge University Press, 2007.

[3] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "A line in the sand: A wireless sensor network for target detection, classification, and tracking," *Computer Networks*, vol. 46, pp. 605–634, July 2004.

[4] H. Baldus, K. Klabunde, and G. Müsch, "Reliable set-up of medical body-sensor networks," in *Proceedings of the 1st European Workshop on Wireless Sensor Networks (EWSN)*, ser. LNCS, vol. 2920, Berlin, Germany, 2004, pp. 353–363.

[5] R. Barták, "Constraint programming: In pursuit of the holy grail," in *Proceedings of the Week of Doctoral Students (WDS)*, Prague, Czech Republic, 1999, pp. 555–564.

[6] J. Beutel, "Metrics for sensor network platforms," in *Proceedings of the ACM Workshop on Real-World Wireless Sensor Networks (REALWSN)*, Uppsala, Sweden, 2006, pp. 26–30.

[7] H. L. Bodlaender, R. B. Tan, T. C. van Dijk, and J. van Leeuwen, "Integer maximum flow in wireless sensor networks with energy constraint," in *Proceedings of the 11th Scandinavian Workshop on Algorithm Theory (SWAT)*, ser. LNCS, vol. 5124, Gothenburg, Sweden, 2008, pp. 102–113.

[8] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, 2004.

[9] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Boulder, Colorado, USA, 2006, pp. 307–320.

[10] N. Burri, P. von Rickenbach, and R. Wattenhofer, "Dozer: Ultra-low power data gathering in sensor networks," in *Proceedings of the 6th ACM International Conference on Information Processing in Sensor Networks (IPSN)*, Cambridge, Massachusetts, USA, 2007, pp. 450–459.

[11] M. Ceriotti, L. Mottola, G. P. Picco, A. L., S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon, "Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment," in *Proceedings of the 8th ACM International Conference on Information Processing in Sensor Networks (IPSN)*, San Francisco, California, USA, 2009.

[12] J.-H. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, pp. 609–619, August 2004.

[13] J.-Y. Chen, G. Pandurangan, and D. Xu, "Robust computation of aggregates in wireless sensor networks: Distributed randomized algorithms and analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 9, pp. 987–1000, September 2006.

[14] K. Chintalapudi, T. Fu, J. Paek, N. Kothari, S. Rangwala, J. Caffrey, R. Govindan, E. Johnson, and S. Masri, "Monitoring civil structures with a wireless sensor network," *IEEE Internet Computing*, vol. 10, no. 2, pp. 26–34, March-April 2006.

[15] K. Chintalapudi, J. Paek, O. Gnawali, T. S. Fu, K. Dantu, J. Caffrey, R. Govindan, E. Johnson, and S. Masri, "Structural damage detection and localization using netshm," in *Proceedings of the 5th ACM International Conference on Information Processing in Sensor Networks (IPSN)*, Nashville, Tennessee, USA, 2006, pp. 475–482.

[16] K. K. Chintalapudi and L. Venkatraman, "On the design of mac protocols for low-latency hard real-time discrete control applications over 802.15.4 hardware," in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN)*, St. Louis, Missouri, USA, 2008, pp. 356–367.

[17] C. A. C. Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Knowledge and Information Systems*, vol. 1, no. 3, pp. 269–308, August 1999.

[18] I. Demirkol, C. Ersoy, and F. Alagöz, "Mac protocols for wireless sensor networks: A survey," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 115–121, April 2006.

[19] S. Du, A. K. Saha, and D. B. Johnson, "Rmac: A routing-enhanced duty-cycle mac protocol for wireless sensor networks," in *Proceedings of the 26th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Anchorage, Alaska, USA, 2007, pp. 1478–1486.

[20] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki – a lightweight and flexible operating system for tiny networked sensors," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN)*, Tampa, Florida, USA, 2004, pp. 455–462.

[21] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali, "Protothreads: Simplifying event-driven programming of memory-constrained embedded systems," in *Proceedings of the 4th ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Boulder, Colorado, USA, 2006, pp. 29–42.

[22] A. Dunkels, F. Österlind, and Z. He, "An adaptive communication architecture for wireless sensor networks," in *Proceedings of the 5th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Sydney, Australia, 2007, pp. 335–349.

[23] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proceedings of the 4th Workshop on Embedded Networked Sensors (EmNets)*, Cork, Ireland, 2007, pp. 28–32.

[24] A. El-Hoiydi, "Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, vol. 5, New York, NY, USA, 2002, pp. 3418–3423.

[25] A. El-Hoiydi and J.-D. Decotignie, "Wisemac: An ultra low power mac protocol for multi-hop wireless sensor networks," in *Proceedings of the 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, ser. Lecture Notes in Computer Science, LNCS 3121, Turku, Finland, 2004, pp. 18–31.

[26] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt, "Mspsim–an extensible simulator for msp430-equipped sensor boards," in *Proceedings of the 4th European Conference on Wireless Sensor Networks (EWSN)*, Delft, The Netherlands, 2007.

[27] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Seattle, Washington, USA, 1999, pp. 263–270.

[28] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo, "The collection tree protocol," 2006, TEP 123.

[29] C. Frei and B. Faltings, "Abstraction and constraint satisfaction techniques for planning bandwidth allocation," in *Proceedings of the 19th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Tel Aviv, Israel, 2000, pp. 235–244.

[30] T. Frühwirth and P. Brisset, "Optimal placement of base stations in wireless indoor telecommunication," in *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming (CP)*, ser. LNCS, vol. 1998, Pisa, Italy, 1998, pp. 476–480.

[31] A. Förster, "Machine learning techniques applied to wireless ad-hoc networks: Guide and survey," in *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP)*, Melbourne, Australia, 2007, pp. 365–370.

[32] A. Förster and A. L. Murphy, "Froms: Feedback routing for optimizing multiple sinks in wsn with reinforcement learning," in *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP)*, Melbourne, Australia, 2007, pp. 371–376.

[33] A. Förster, A. L. Murphy, J. Schiller, and K. Terfloth, "An efficient implementation of reinforcement learning based routing on real wsn hardware," in *Proceedings of the 4th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Avignon, France, 2008, pp. 247–252.

[34] O. Gnawali, K.-Y. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, and E. Kohler, "The tenet architecture for tiered sensor networks," in *Proceedings of the 4th ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Boulder, Colorado, USA, 2006, pp. 153–166.

[35] C. Guettier, G. L. Lann, and J.-F. Hermant, "Ad hoc sensor networks, constraint programming and distributed agreement," in *Proceedings of the IEEE International Conference on Information Technology: Research and Education (ITRE)*, Newark, New Jersey, USA, 2003, pp. 291–295.

[36] R. W. Ha, P.-H. Ho, X. S. Shen, and J. Zhang, "Sleep scheduling for wireless sensor networks via network flow model," *Computer Communications*, vol. 29, pp. 2469–2481, Marco 2006.

[37] Y. Halevi and A. Ray, "Integrated communication and control systems: Part i–analysis," *Journal of Dynamic Systems, Measurement, and Control*, vol. 110, pp. 367–373, December 1988.

[38] G. Halkes and K. Langendoen, "Crankshaft: An energy-efficient mac-protocol for dense wireless sensor networks," in *Proceedings of the 4th European Conference on Wireless Sensor Networks (EWSN)*, ser. LNCS, vol. 4373, Delft, The Netherlands, 2007, pp. 228–244.

[39] J. He, J. Rexford, and M. Chiang, "Don't optimize existing protocols, design optimizable protocols," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 53–58, July 2007.

[40] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS)*, Wailea Maui, Hawaii, USA, 2000, pp. 1–10.

[41] J. L. Hill and D. E. Culler, "Mica: A wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, no. 6, pp. 12–24, November/December 2002.

[42] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, Maryland, USA, 2004, pp. 81–94.

[43] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom)*, Boston, Massachusetts, USA, 2000, pp. 56–67.

[44] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," in *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, San Jose, CA, USA, 2002, pp. 96–107.

[45] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: Mobile networking for "smart dust"," in *Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Seattle, Washington, USA, 1999, pp. 271–278.

[46] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *Proceedings of the 6th ACM International Conference on Information Processing in Sensor Networks (IPSN)*, Cambridge, Massachusetts, USA, 2007, pp. 254–263.

[47] Y. Kim, H. Shin, and H. Cha, "Y-mac: An energy-efficient multi-channel mac protocol for dense wireless sensor networks," in *Proceedings of the 7th ACM International Conference on Information Processing in Sensor Networks (IPSN)*, St. Louis, Missouri, 2008, pp. 53–63.

[48] P. R. Kotecha, M. Bhushan, and R. D. Gudi, "Constraint programming based robust sensor network design," *Industrial and Engineering Chemistry Research*, vol. 46, no. 18, pp. 5985–5999, August 2007.

[49] V. Kumar, "Algorithms for constraint-satisfaction problems: A survey," *AI Magazine*, vol. 13, no. 1, pp. 32–44, 1992.

[50] K. Langendoen, "Medium access control in wireless sensor networks," in *Medium Access Control in Wireless Networks*, H. Wu and Y. Pan, Eds. Nova Science Publishers, 2008.

[51] H. K. Le, D. Henriksson, and T. Abdelzaher, "A practical multi-channel media access control protocol for wireless sensor networks," in *Proceedings of the 7th ACM International Conference on Information Processing in Sensor Networks*, St. Louis, Missouri, USA, 2008, pp. 70–81.

[52] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "Tinyos: An operating system for sensor networks," in *Ambient Intelligence*. Springer-Verlag, 2004.

[53] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, California, USA, 2004.

[54] D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed, "Detection, classification, and tracking of targets," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 17–29, March 2002.

[55] J. Li and G. Y. Lazarou, "A bit-map-assisted energy-efficient mac scheme for wireless sensor networks," in *Proceedings of the 3rd ACM/IEEE International*

Symposium on Information Processing in Sensor Networks (IPSN), Berkeley, California, USA, 2004, pp. 55–60.

[56] L. Luo, T. F. Abdelzahner, T. He, and J. A. Stankovic, "Envirosuite: An environmentally immersive programming framework for sensor networks," *ACM Transactions on Embedded Computing Systems*, vol. 5, no. 3, pp. 543–576, August 2006.

[57] R. Madan and S. Lall, "Distributed algorithms for maximum lifetime routing in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 8, pp. 2185–2193, August 2006.

[58] S. Mahlknecht and M. Böck, "Csma-mps: A minimum preamble sampling mac protocol for low power wireless sensor networks," in *Proceedings of the 5th IEEE International Workshop on Factory Communication Systems (WFCS)*, Vienne, Austria, 2004, pp. 73–80.

[59] A. Mainwaring, J. Polastre, R. Szewczyk, C. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM International workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, Georgia, USA, 2002, pp. 88–97.

[60] K. Martinez, J. K. Hart, and R. Ong, "Environmental sensor networks," *IEEE Computer*, vol. 37, no. 8, pp. 50–56, August 2004.

[61] R. Musaloiu-E., C.-J. M. Liang, and A. Terzis, "Koala: Ultra-low power data retrieval in wireless sensor networks," in *Proceedings of the 7th ACM International Conference on Information Processing in Sensor Networks (IPSN)*, St. Louis, Missouri, USA, 2008, pp. 421–432.

[62] V. Namboodiri and A. Keshavarzian, "Alert: An adaptive low-latency event-driven mac protocol for wireless sensor networks," in *Proceedings of the 7th ACM International Conference on Information Processing in Sensor Networks (IPSN)*, St. Louis, Missouri, USA, 2008, pp. 159–170.

[63] J. Nilsson, "Real-time control systems with delays," Ph.D. dissertation, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1998.

[64] A. Osyczka, "Multicriterion optimization for engineering design," in *Design Optimization*, J. S. Gero, Ed. Academic Press, 1985, pp. 193–227.

[65] J. Paek, K. Chintalapudi, R. Govindan, J. Caffrey, and S. Masri, "A wireless sensor network for structural health monitoring: Performance and experience," in *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNetS)*, Sydney, Australia, 2005, pp. 1–10.

[66] E. M. Petriu, N. D. Georganas, D. C. Petriu, and D. Makrakis, "Sensor-based information appliances," *IEEE Instrumentation & Measurement Magazine*, vol. 3, no. 4, pp. 31–35, December 2000.

[67] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, Maryland, USA, 2004, pp. 95–107.

[68] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proceedings of the 4th ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*, Los Angeles, California, USA, 2005.

[69] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," in *Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, California, USA, 2003, pp. 181–192.

[70] B. Raman and K. Chebrolu, "Censor networks: A critique of "sensor networks" from a systems perspective," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 75–78, July 2008.

[71] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-mac: A hybrid mac for wireless sensor networks," in *Proceedings of the 3rd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, San Diego, California, USA, 2005, pp. 90–101.

[72] B. J. Ritzel, J. W. Eheart, and S. Ranjithan, "Using genetic algorithms to solve a multiple objective groundwater pollution containment problem," *Water Resources Research*, vol. 30, no. 5, pp. 1589–1603, May 1994.

[73] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2003.

[74] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Optimizing sensor networks in the energy-latency-density design space," *IEEE Transactions on Mobile Computing*, vol. 1, no. 1, pp. 70–80, January–March 2002.

[75] T. Seeley, *The Wisdom of the Hive.* Harvard University Press, 2005.

[76] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and beyond: New aggregation techniques for sensor networks," in *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, Maryland, USA, 2004, pp. 239–249.

[77] Y. Sun, S. Du, O. Gurewitz, and D. B. Johnson, "Dw-mac: A low latency, energy efficient demand-wakeup mac protocol for wireless sensor networks," in *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Hong Kong, Hong Kong, China, 2008, pp. 53–62.

[78] Y. Sun, O. Gurewitz, and D. B. Johnson, "Ri-mac: A receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks," in *Proceedings of the 6th ACM International Conference on Embedded Network Sensor Systems (SenSys)*, Raleigh, North Carolina, USA, 2008, pp. 1–14.

[79] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, Maryland, USA, 2004, pp. 214–226.

[80] H. Tamaki, H. Kita, and S. Kobayashi, "Multi-objective optimization by genetic algorithms," in *Proceedings of the International Conference on Evolutionary Computation*, Nagoya, Japan, 1996, pp. 517–522.

[81] *CC2420 Datasheet*, Texas Instruments, March 2007.

[82] F. A. Tobagi and L. Kleinrock, "Packet switching in radio channels: Part Il–the hidden terminal problem in carrier sense multiple-access and the busy-tone solution," *IEEE Transactions on Communications*, vol. 23, no. 12, pp. 1417–1433, December 1975.

[83] G. Tolle and D. Culler, "Design of an application-cooperative management system for wireless sensor networks," in *Proceedings of the 2nd European Workshop on Wireless Sensor Networks (EWSN)*, Istanbul, Turkey, 2005, pp. 121–132.

[84] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hongtoll, "A macroscope in the redwoods," in *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, San Diego, California, USA, 2005, pp. 51–63.

[85] L. v. Hoesel and P. Havinga, "A lightweight medium access protocol (lmac) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches," in *Proceedings of the 1st International Workshop on Networked Sensing Systems (INSS)*, Tokyo, Japan, 2004.

[86] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, California, USA, 2003, pp. 171–180.

[87] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis," Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA, Tech. Rep., 1998.

[88] M. Wallace, S. Novello, and J. Schimpf, "Eclipse: A platform for constraint logic programming," IC-Parc, William Penney Laboratory, Imperial College, London, UK, Tech. Rep., 1997.

[89] M. Wallace, "Constraint programming – the paradigm to watch," *Constraint Programming Letters*, vol. 1, pp. 7–13, November 2007.

[90] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[91] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, California, USA, 2003, pp. 14–27.

[92] F. Xia, Y.-C. Tian, Y. Li, and Y. Sun, "Wireless sensor actuator network design for mobile control applications," *Sensors*, vol. 7, pp. 2157–2173, October 2007.

[93] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, Maryland, USA, 2004, pp. 13–24.

[94] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the 21st Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, New York City, New York, USA, 2002, pp. 1567–1576.

[95] ——, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, June 2004.

[96] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle mac with scheduled channel polling," in *Proceedings of the 4th ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Boulder, Colorado, USA, 2006, pp. 321–334.

[97] K. Yuen, B. Li, and B. Liang, "Distributed data gathering in multi-sink sensor networks with correlated sources," in *Proceedings of IFIP Networking*, ser. LNCS, vol. 3976, Coimbra, Portugal, 2006, pp. 868–879.

[98] W. Zhang, M. S. Branicky, and S. Phillips, "Stability of networked control systems," *IEEE Control Systems Magazine*, vol. 21, no. 1, pp. 84–99, February 2001.

[99] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, California, USA, 2003, pp. 1–13.

[100] J. Zhao, R. Govindan, and D. Estrin, "Sensor network tomography: Monitoring wireless sensor networks," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 1, pp. 64–64, January 2002.

[101] ——, "Computing aggregates for monitoring wireless sensor networks," in *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, Anchorage, AK, USA, 2003, pp. 139–148.

[102] T. Zheng, S. Radhakrishnan, and V. Sarangan, "Pmac: An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS)*, Denver, Colorado, USA, 2005, pp. 65–72.

[103] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, November 1999.

[104] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Proceedings of the 31st IEEE Conference on Local Computer Networks (LCN)*, Tampa, Florida, USA, 2006, pp. 641–648.