PET ENGINEERING COLLEGE

An ISO 9001:2015 Certified Institution

Accredited by NAAC, Approved by AICTE, Recognized by Government of Tamil Nadu and Affiliated to Anna University

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

UNIT - I

INTRODUCTION

CLASS	**: S6 ECE**
SUBJECT CODE	**: CEC365**
SUBJECT NAME	**: WIRELESS SENSOR NETWORKS DESIGN**
REGULATION	**: 2021**

These interactions can be scoped both in time and in space (reporting events only within a given time span, only from certain areas, and so on). These requirements can also change dynamically overtime; sinks have to have a means to inform the sensors of their requirements at runtime. Moreover, these interactions can take place only for one specific request of a sink (so-called "one-shot queries"), or they could be long-lasting relationships between many sensors and many sinks.

The examples also have shown a wide diversity in **deployment options**. They range from well-planned, fixed deployment of sensor nodes (e.g. in machinery maintenance applications) to random deployment by dropping a large number of nodes from an aircraft over a forest fire. In addition, sensor nodes can be mobile themselves and compensate for shortcomings in the deployment process by moving, in a postdeployment phase, to positions such that their sensing tasks can be better fulfilled [17]. They could also be mobile because they are attached to other objects (in the logistics applications, for example) and the network has to adapt itself to the location of nodes.

The applications also influence the available **maintenance options**: Is it feasible and practical to perform maintenance on such sensors – perhaps even required in the course of maintenance on associated machinery? Is maintenance irrelevant because these networks are only deployed in a strictly ad hoc, short-term manner with a clear delimitation of maximum mission time (like in disaster recovery operations)? Or do these sensors have to function unattended, for a long time, with no possibility for maintenance?

Closely related to the maintenance options are the **options for energy supply**. In some applications, wired power supply is possible and the question is mute. For self-sustained sensor nodes, depending on the required mission time, energy supply can be trivial (applications with a few days of usage only) or a challenging research problem, especially when no maintenance is possible but nodes have to work for years. Obviously, acceptable price and size per node play a crucial role in designing energy supply.

1.4 Challenges for WSNs

Handling such a wide range of application types will hardly be possible with any single realization of a WSN. Nonetheless, certain common traits appear, especially with respect to the characteristics and the required mechanisms of such systems. Realizing these characteristics with new mechanisms is the major challenge of the vision of wireless sensor networks.

1.4.1 Characteristic requirements

The following characteristics are shared among most of the application examples discussed above:

Type of service The service type rendered by a conventional communication network is evident – it moves bits from one place to another. For a WSN, moving bits is only a means to an end, but not the actual purpose. Rather, a WSN is expected to provide meaningful information and/or actions about a given task: "People want answers, not numbers" (Steven Glaser, UC Berkeley, in [367]). Additionally, concepts like *scoping* of interactions to specific geographic regions or to time intervals will become important. Hence, new paradigms of using such a network are required, along with new interfaces and new ways of thinking about the service of a network.

Quality of Service Closely related to the type of a network's service is the quality of that service. Traditional quality of service requirements – usually coming from multimedia-type applications – like bounded delay or minimum bandwidth are irrelevant when applications are tolerant to latency [26] or the bandwidth of the transmitted data is very small in the first place. In some cases, only occasional delivery of a packet can be more than enough; in other cases, very high reliability requirements exist. In yet other cases, delay *is* important when actuators are to be controlled in a real-time fashion by the sensor network. The packet delivery ratio is an insufficient metric; what is relevant is the amount and quality of information that can be extracted at given sinks about the observed objects or area.

Therefore, adapted quality concepts like reliable detection of events or the approximation quality of a, say, temperature map is important.

Fault tolerance Since nodes may run out of energy or might be damaged, or since the wireless communication between two nodes can be permanently interrupted, it is important that the WSN as a whole is able to tolerate such faults. To tolerate node failure, redundant deployment is necessary, using more nodes than would be strictly necessary if all nodes functioned correctly.

Lifetime In many scenarios, nodes will have to rely on a limited supply of energy (using batteries). Replacing these energy sources in the field is usually not practicable, and simultaneously, a WSN must operate at least for a given mission time or as long as possible. Hence, the **lifetime** of a WSN becomes a very important figure of merit. Evidently, an energy-efficient way of operation of the WSN is necessary.

As an alternative or supplement to energy supplies, a limited power source (via power sources like solar cells, for example) might also be available on a sensor node. Typically, these sources are not powerful enough to ensure continuous operation but can provide some recharging of batteries. Under such conditions, the lifetime of the network should ideally be infinite.

The lifetime of a network also has direct trade-offs against quality of service: investing more energy can increase quality but decrease lifetime. Concepts to harmonize these trade-offs are required.

The precise *definition of lifetime* depends on the application at hand. A simple option is to use the time until the first node fails (or runs out of energy) as the network lifetime. Other options include the time until the network is disconnected in two or more partitions, the time until 50 % (or some other fixed ratio) of nodes have failed, or the time when for the first time a point in the observed region is no longer covered by at least a single sensor node (when using redundant deployment, it is possible and beneficial to have each point in space covered by several sensor nodes initially).

Scalability Since a WSN might include a large number of nodes, the employed architectures and protocols must be able scale to these numbers.

Wide range of densities In a WSN, the number of nodes per unit area – the *density* of the network – can vary considerably. Different applications will have very different node densities. Even within a given application, density can vary over time and space because nodes fail or move; the density also does not have to homogeneous in the entire network (because of imperfect deployment, for example) and the network should adapt to such variations.

Programmability Not only will it be necessary for the nodes to process information, but also they will have to react flexibly on changes in their tasks. These nodes should be programmable, and their programming must be changeable during operation when new tasks become important. A fixed way of information processing is insufficient.

Maintainability As both the environment of a WSN and the WSN itself change (depleted batteries, failing nodes, new tasks), the system has to adapt. It has to monitor its own health and status to change operational parameters or to choose different trade-offs (e.g. to provide lower quality when energy resource become scarce). In this sense, the network has to maintain itself; it could also be able to interact with external maintenance mechanisms to ensure its extended operation at a required quality [534].

1.4.2 Required mechanisms

To realize these requirements, innovative mechanisms for a communication network have to be found, as well as new architectures, and protocol concepts. A particular challenge here is the need to find mechanisms that are sufficiently specific to the idiosyncrasies of a given application to support the specific quality of service, lifetime, and maintainability requirements [246]. On the other hand, these mechanisms also have to generalize to a wider range of applications lest a complete from-scratch development and implementation of a WSN becomes necessary for every individual application – this would likely render WSNs as a technological concept economically infeasible.

Some of the mechanisms that will form typical parts of WSNs are:

Multihop wireless communication While wireless communication will be a core technique, a direct communication between a sender and a receiver is faced with limitations. In particular, communication over long distances is only possible using prohibitively high transmission power. The use of intermediate nodes as relays can reduce the total required power. Hence, for many forms of WSNs, so-called *multihop communication* will be a necessary ingredient.

Energy-efficient operation To support long lifetimes, energy-efficient operation is a key technique. Options to look into include energy-efficient data transport between two nodes (measured in J/bit) or, more importantly, the energy-efficient determination of a requested information. Also, nonhomogeneous energy consumption – the forming of "hotspots" – is an issue.

Auto-configuration A WSN will have to configure most of its operational parameters autonomously, independent of external configuration – the sheer number of nodes and simplified deployment will require that capability in most applications. As an example, nodes should be able to determine their geographical positions only using other nodes of the network – so-called "self-location". Also, the network should be able to tolerate failing nodes (because of a depleted battery, for example) or to integrate new nodes (because of incremental deployment after failure, for example).

Collaboration and in-network processing In some applications, a single sensor is not able to decide whether an event has happened but several sensors have to collaborate to detect an event and only the joint data of many sensors provides enough information. Information is processed in the network itself in various forms to achieve this collaboration, as opposed to having every node transmit all data to an external network and process it "at the edge" of the network.

An example is to determine the highest or the average temperature within an area and to report that value to a sink. To solve such tasks efficiently, readings from individual sensors can be *aggregated* as they propagate through the network, reducing the amount of data to be transmitted and hence improving the energy efficiency. How to perform such aggregation is an open question.

Data centric Traditional communication networks are typically centered around the transfer of data between two specific devices, each equipped with (at least) one network address – the operation of such networks is thus **address-centric**. In a WSN, where nodes are typically deployed redundantly to protect against node failures or to compensate for the low quality of a single node's actual sensing equipment, the identity of the particular node supplying data becomes irrelevant. What is important are the answers and values themselves, not which node has provided them. Hence, switching from an address-centric paradigm to a **data-centric** paradigm in designing architecture and communication protocols is promising.

An example for such a data-centric interaction would be to request the average temperature in a given location area, as opposed to requiring temperature readings from individual nodes. Such a data-centric paradigm can also be used to set conditions for alerts or events ("raise an alarm if temperature exceeds a threshold"). In this sense, the data-centric approach is closely related to query concepts known from databases; it also combines well with collaboration, in-network processing, and aggregation.

Locality Rather a design guideline than a proper mechanism, the principle of locality will have to be embraced extensively to ensure, in particular, scalability. Nodes, which are very limited in resources like memory, should attempt to limit the state that they accumulate during protocol processing to only information about their direct neighbors. The hope is that this will allow the network to scale to large numbers of nodes without having to rely on powerful processing at each single node. How to combine the locality principle with efficient protocol designs is still an open research topic, however.

Exploit trade-offs Similar to the locality principle, WSNs will have to rely to a large degree on exploiting various inherent trade-offs between mutually contradictory goals, both during system/protocol design and at runtime. Examples for such trade-offs have been mentioned already: higher energy expenditure allows higher result accuracy, or a longer lifetime of the entire network trades off against lifetime of individual nodes. Another important trade-off is node density: depending on application, deployment, and node failures at runtime, the density of the network can change considerably – the protocols will have to handle very different situations, possibly present at different places of a single network. Again, not all the research questions are solved here.

Harnessing these mechanisms such that they are easy to use, yet sufficiently general, for an application programmer is a major challenge. Departing from an address-centric view of the network requires new programming interfaces that go beyond the simple semantics of the conventional socket interface and allow concepts like required accuracy, energy/accuracy trade-offs, or scoping.

1.5 Why are sensor networks different?

On the basis of these application examples and main challenges, two close relatives of WSNs become apparent: Mobile Ad Hoc Networks (MANETs) on the one hand and fieldbuses on the other hand.

1.5.1 Mobile ad hoc networks and wireless sensor networks

An ad hoc network is a network that is setup, literally, for a specific purpose, to meet a quickly appearing communication need. The simplest example of an ad hoc network is perhaps a set of computers connected together via cables to form a small network, like a few laptops in a meeting room. In this example, the aspect of *self-configuration* is crucial – the network is expected to work without manual management or configuration.

Usually, however, the notion of a MANET is associated with wireless communication and specifically *wireless* multihop communication; also, the name indicates the mobility of participating nodes as a typical ingredient. Examples for such networks are disaster relief operations – firefighters communicate with each other – or networks in difficult locations like large construction sites, where the deployment of wireless infrastructure (access points etc.), let alone cables, is not a feasible option. In such networks, the individual nodes together form a network that relays packets between nodes to extend the reach of a single node, allowing the network to span larger geographical areas than would be possible with direct sender – receiver communication. The two basic challenges in a MANET are the reorganization of the network as nodes move about and handling the problems of the limited reach of wireless communication. Literature on MANETs that summarize these problems and their solutions abound, as these networks are still a very active field of research; popular books include [635, 793, 827].

These general problems are shared between MANETs and WSNs. Nonetheless, there are some principal differences between the two concepts, warranting a distinction between them and regarding separate research efforts for each one.

Applications and equipment MANETs are associated with somewhat different applications as well as different user equipment than WSNs: in a MANET, the terminal can be fairly powerful (a laptop or a PDA) with a comparably large battery. This equipment is needed because in the typical MANET applications, there is usually a human in the loop: the MANET is used for voice communication between two distant peers, or it is used for access to a remote infrastructure like a Web server. Therefore, the equipment has to be powerful enough to support these applications.

Application specific Owing to the large number of conceivable combinations of sensing, computing, and communication technology, many different application scenarios for WSNs become possible. It is unlikely that there will be a "one-size-fits-all" solution for all these potentially very different possibilities. As one example, WSNs are conceivable with very different network densities, from very sparse to very dense deployments, which will require different or at least adaptive protocols. This diversity, although present, is not quite as large in MANETs.

Environment interaction Since WSNs have to interact with the environment, their traffic characteristics can be expected to be very different from other, human-driven forms of networks. A typical consequence is that WSNs are likely to exhibit very low data rates over a large timescale, but can have very bursty traffic when something happens (a phenomenon known from real-time systems as event showers or alarm storms). Long periods (months) of inactivity can alternate with short periods (seconds or minutes) of very high activity in the network, pushing its capacity to the limits. MANETs, on the other hand, are used to support more conventional applications (Web, voice, and so on) with their comparably well understood traffic characteristics.

Scale Potentially, WSNs have to scale to much larger numbers (thousands or perhaps hundreds of thousands) of entities than current ad hoc networks, requiring different, more scalable solutions. As a concrete case in point, endowing sensor nodes with a unique identifier is costly (either at production or at runtime) and might be an overhead that could be avoided – hence, protocols that work without such identifiers might become important in WSNs, whereas it is fair to assume such identifiers to exist in MANET nodes.

Energy In both WSNs and MANETs, energy is a scare resource. But WSNs have tighter requirements on network lifetime, and recharging or replacing WSN node batteries is much less an option than in MANETs. Owing to this, the impact of energy considerations on the entire system architecture is much deeper in WSNs than in MANETs.

Self configurability Similar to ad hoc networks, WSNs will most likely be required to self-configure into connected networks, but the difference in traffic, energy trade-offs, and so forth, could require new solutions. Nevertheless, it is in this respect that MANETs and WSNs are probably most similar.

Dependability and QoS The requirements regarding dependability and QoS are quite different. In a MANET, each individual node should be fairly reliable; in a WSN, an individual node is next to irrelevant. The quality of service issues in a MANET are dictated by traditional applications (low jitter for voice applications, for example); for WSNs, entirely new QoS concepts are required, which also take energy explicitly into account.

Data centric Redundant deployment will make data-centric protocols attractive in WSNs. This concept is alien to MANETs. Unless applications like file sharing are used in MANETs, which do bear some resemblance to data centric approaches, data-centric protocols are irrelevant to MANETs – but these applications do not represent the typically envisioned use case.

Simplicity and resource scarceness Since sensor nodes are simple and energy supply is scarce, the operating and networking software must be kept orders of magnitude simpler compared to today's desktop computers. This simplicity may also require breaking with conventional layering rules for networking software, since layering abstractions typically cost time and space. Also, resources like memory, which is relevant for comparably heavy-weight routing protocols as those used in MANETs, is not available in arbitrary quantities, requiring new, scalable, resource-efficient solutions.

Mobility The mobility problem in MANETs is caused by nodes moving around, changing multihop routes in the network that have to be handled. In a WSN, this problem can also exist if the sensor nodes are mobile in the given application. There are two additional aspects of mobility to be considered in WSNs.

First, the sensor network can be used to detect and observe a physical phenomenon (in the intrusion detection applications, for example). This phenomenon is the cause of events that happen in the network (like raising of alarms) and can also cause some local processing, for example, determining whether there really is an intruder. What happens if this phenomenon moves about? Ideally, data that has been gathered at one place should be available at the next one. Also, in tracking applications, it is the explicit task of the network to ensure that some form of activity happens in nodes that surround the phenomenon under observation.

Second, the sinks of information in the network (nodes where information should be delivered to) can be mobile as well. In principle, this is no different than node mobility in the general MANET sense, but can cause some difficulties for protocols that operate efficiently in fully static scenarios. Here, carefully observing trade-offs is necessary.

Furthermore, in both MANET and WSNs, mobility can be correlated – a group of nodes moving in a related, similar fashion. This correlation can be caused in a MANET by, for example, belonging to a group of people traveling together. In a WSN, the movement of nodes can be correlated because nodes are jointly carried by a storm, a river, or some other fluid.

In summary, there are commonalities, but the fact that WSNs have to support very different applications, that they have to interact with the physical environment, and that they have to carefully adjudicate various trade-offs justifies considering WSNs as a system concept distinct from MANETs.

1.5.2 Fieldbuses and wireless sensor networks

Fieldbuses are networks that are specifically designed for operation under hard real-time constraints and usually with inbuilt fault tolerance, to be used predominantly in control applications, that is, as part of a control loop. Examples include the Profibus and IEEE 802.4 Token Bus networks [372] for factory floor automation or the CAN bus for onboard networks in cars; some example summaries on the topic include [532, 644, 881]. Because of the stringent hard real-time requirements,

2

Single-node architecture

Objectives of this Chapter

This fairly long chapter explains the basic part of a wireless sensor network: the nodes as such. It discusses the principal tasks of a node – computation, storage, communication, and sensing/ actuation – and which components are required to perform these tasks. Then, the energy consumption of these components is described: how energy can be stored, gathered from the environment, and saved by intelligently controlling the mode of operation of node components. This control has to be exerted by an operating system like execution environment, which is described in the last major section of this chapter. Finally, some examples of sensor nodes are given.

At the end of this chapter, the reader should have an understanding of the capabilities and limitations of the nodes in a sensor network. It lays the foundation for the following chapter, which discusses the principal options on how individual sensor nodes can be connected into a wireless sensor network.

Chapter Outline

Building a wireless sensor network first of all requires the constituting nodes to be developed and available. These nodes have to meet the requirements that come from the specific requirements of a given application: they might have to be small, cheap, or energy efficient, they have to be equipped with the right sensors, the necessary computation and memory resources, and they need adequate communication facilities. These hardware components and their composition into a functioning node are described in Section 2.1; the power consumption of these components and the ensuing trade-offs are discussed in Section 2.2. As this chapter only focuses onto an individual node, the consequences of choosing a particular communication technology for the architecture of a wireless sensor network as a whole are described in Chapter 3.

In addition to the hardware of sensor nodes, the operating system and programming model is an important consideration. Section 2.3 describes the tasks of such an operating system along with some examples as well as suitable programming interfaces.

2.1 Hardware components

2.1.1 Sensor node hardware overview

When choosing the hardware components for a wireless sensor node, evidently the application's requirements play a decisive factor with regard mostly to size, costs, and energy consumption of the nodes – communication and computation facilities as such are often considered to be of acceptable quality, but the trade-offs between features and costs is crucial. In some extreme cases, an entire sensor node should be smaller than 1 cc, weigh (considerably) less than 100 g, be substantially cheaper than US$1, and dissipate less than 100 µW [667]. In even more extreme visions, the nodes are sometimes claimed to have to be reduced to the size of grains of dust. In more realistic applications, the mere size of a node is not so important; rather, convenience, simple power supply, and cost are more important [126].

These diversities notwithstanding, a certain common trend is observable in the literature when looking at typical hardware platforms for wireless sensor nodes. While there is certainly not a single standard available, nor would such a standard necessarily be able to support all application types, this section will survey these typical sensor node architectures. In addition, there are a number of research projects that focus on shrinking any of the components in size, energy consumption, or costs, based on the fact that custom off-the-shelf components do currently not live up to some of the more stringent application requirements. But as this book focuses on the networking aspects of WSNs, these efforts are not discussed here.

A basic sensor node comprises five main components (Figure 2.1):

Controller A controller to process all the relevant data, capable of executing arbitrary code.

Memory Some memory to store programs and intermediate data; usually, different types of memory are used for programs and data.

Sensors and actuators The actual interface to the physical world: devices that can observe or control physical parameters of the environment.

Communication Turning nodes into a network requires a device for sending and receiving information over a wireless channel.

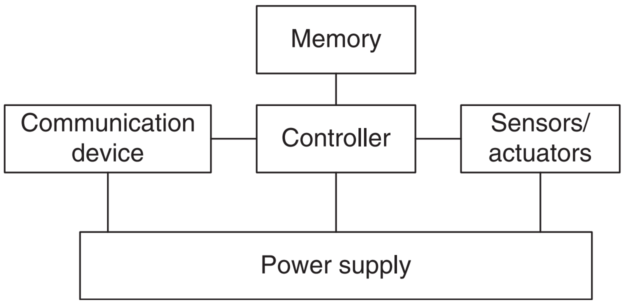

Figure 2.1 Overview of main sensor node hardware components

Power supply As usually no tethered power supply is available, some form of batteries are necessary to provide energy. Sometimes, some form of recharging by obtaining energy from the environment is available as well (e.g. solar cells).

Each of these components has to operate balancing the trade-off between as small an energy consumption as possible on the one hand and the need to fulfill their tasks on the other hand. For example, both the communication device and the controller should be turned off as long as possible. To wake up again, the controller could, for example, use a preprogrammed timer to be reactivated after some time. Alternatively, the sensors could be programmed to raise an interrupt if a given event occurs – say, a temperature value exceeds a given threshold or the communication device detects an incoming transmission.

Supporting such alert functions requires appropriate interconnection between individual components. Moreover, both control and data information have to be exchanged along these interconnections. This interconnection can be very simple – for example, a sensor could simply report an analog value to the controller – or it could be endowed with some intelligence of its own, preprocessing sensor data and only waking up the main controller if an actual event has been detected – for example, detecting a threshold crossing for a simple temperature sensor. Such preprocessing can be highly customized to the specific sensor yet remain simple enough to run continuously, resulting in improved energy efficiency [26].

2.1.2 Controller

Microcontrollers versus microprocessors, FPGAs, and ASICs

The controller is the core of a wireless sensor node. It collects data from the sensors, processes this data, decides when and where to send it, receives data from other sensor nodes, and decides on the actuator's behavior. It has to execute various programs, ranging from time-critical signal processing and communication protocols to application programs; it is the Central Processing Unit (CPU) of the node.

Such a variety of processing tasks can be performed on various controller architectures, representing trade-offs between flexibility, performance, energy efficiency, and costs.

One solution is to use general-purpose processors, like those known from desktop computers. These processors are highly overpowered, and their energy consumption is excessive. But simpler processors do exist, specifically geared toward usage in embedded systems. These processors are commonly referred as **microcontrollers**. Some of the key characteristics why these microcontrollers are particularly suited to embedded systems are their flexibility in connecting with other devices (like sensors), their instruction set amenable to time-critical signal processing, and their typically low power consumption; they are also convenient in that they often have memory built in. In addition, they are freely programmable and hence very flexible. Microcontrollers are also suitable for WSNs since they commonly have the possibility to reduce their power consumption by going into **sleep states** where only parts of the controller are active; details vary considerably between different controllers. Details regarding power consumption and energy efficiency are discussed in Section 2.2. One of the main differences to general-purpose systems is that microcontroller-based systems usually do not feature a memory management unit, somewhat limiting the functionality of memory – for example, protected or virtual memory is difficult, if not impossible, to achieve.

A specialized case of programmable processors are Digital Signal Processors (DSPs). They are specifically geared, with respect to their architecture and their instruction set, for processing large amounts of vectorial data, as is typically the case in signal processing applications. In a wireless sensor node, such a DSP could be used to process data coming from a simple analog, wireless communication device to extract a digital data stream. In broadband wireless communication, DSPs are an appropriate and successfully used platform. But in wireless sensor networks, the requirements on wireless communication are usually much more modest (e.g. simpler, easier to process modulations are used that can be efficiently handled in hardware by the communication device itself) and the signal processing tasks related to the actual sensing of data is also not overly complicated. Hence, these advantages of a DSP are typically not required in a WSN node and they are usually not used.

Another option for the controller is to depart from the high flexibility offered by a (fairly general-purpose) microcontroller and to use Field-Programmable Gate Arrays (FPGAs) or Application-Specific Integrated Circuits (ASICs) instead. An FPGA can be reprogrammed (or rather reconfigured) "in the field" to adapt to a changing set of requirements; however, this can take time and energy – it is not practical to reprogram an FPGA at the same frequency as a microcontroller could change between different programs. An ASIC is a specialized processor, custom designed for a given application such as, for example, high-speed routers and switches. The typical trade-off here is loss of flexibility in return for a considerably better energy efficiency and performance. On the other hand, where a microcontroller requires software development, ASICs provide the same functionality in hardware, resulting in potentially more costly hardware development.

For a dedicated WSN application, where the duties of a the sensor nodes do not change over lifetime and where the number of nodes is big enough to warrant the investment in ASIC development, they can be a superior solution. At the current stage of WSN technology, however, the bigger flexibility and simpler usage of microcontrollers makes them the generally preferred solution. However, this is not necessarily the final solution as "convenient programmability over several orders of energy consumption and data processing requirements is a worthy research goal" [648]. In addition, splitting processing tasks between some low-level, fixed functionality put into a very energy-efficient ASIC and high-level, flexible, relatively rarely invoked processing on a microcontroller is an attractive design and research option [26, 648].

For the remainder of this book, a microcontroller-based architecture is assumed.

Some examples for microcontrollers

Microcontrollers that are used in several wireless sensor node prototypes include the Atmel processor or Texas Instrument's MSP 430. In older prototypes, the Intel StrongArm processors have also been used, but this is no longer considered as a practical option; it is included here for the sake of completeness. Nonetheless, as the principal properties of these processors and controllers are quite similar, conclusions from these earlier research results still hold to a large degree.

Intel StrongARM

The Intel StrongARM [379] is, in WSN terms, a fairly high-end processor as it is mostly geared toward handheld devices like PDAs. The SA-1100 model has a 32-bit Reduced Instruction Set Computer (RISC) core, running at up to 206 MHz.

Texas Instruments MSP 430

Texas Instrument provides an entire family of microcontrollers under the family designation MSP 430 [814]. Unlike the StrongARM, it is explicitly intended for embedded applications. Accordingly, it runs a 16-bit RISC core at considerably lower clock frequencies (up to 4 MHz) but comes with a wide range of interconnection possibilities and an instruction set amenable to easy handling of peripherals of different kinds. It features a varying amount of on-chip RAM (sizes are 2–10 kB), several 12-bit analog/digital converters, and a real-time clock. It is certainly powerful enough to handle the typical computational tasks of a typical wireless sensor node (possibly with the exception of driving the radio front end, depending on how it is connected – bit or byte interface – to the controller).

Atmel ATmega

The Atmel ATmega 128L [28] is an 8-bit microcontroller, also intended for usage in embedded applications and equipped with relevant external interfaces for common peripherals.

2.1.3 Memory

The memory component is fairly straightforward. Evidently, there is a need for Random Access Memory (RAM) to store intermediate sensor readings, packets from other nodes, and so on. While RAM is fast, its main disadvantage is that it loses its content if power supply is interrupted. Program code can be stored in Read-Only Memory (ROM) or, more typically, in Electrically Erasable Programmable Read-Only Memory (EEPROM) or flash memory (the later being similar to EEPROM but allowing data to be erased or written in blocks instead of only a byte at a time). Flash memory can also serve as intermediate storage of data in case RAM is insufficient or when the power supply of RAM should be shut down for some time. The long read and write access delays of flash memory should be taken into account, as well as the high required energy.

Correctly dimensioning memory sizes, especially RAM, can be crucial with respect to manufacturing costs and power consumption. However, even general rules of thumbs are difficult to give as the memory requirements are very much application dependent.

2.1.4 Communication device

Choice of transmission medium

The communication device is used to exchange data between individual nodes. In some cases, wired communication can actually be the method of choice and is frequently applied in many sensor networklike settings (using field buses like Profibus, LON, CAN, or others). The communication devices for these networks are custom off-the-shelf components.

The case of wireless communication is considerably more interesting. The first choice to make is that of the transmission medium – the usual choices include radio frequencies, optical communication, and ultrasound; other media like magnetic inductance are only used in very specific cases. Of these choices, Radio Frequency (RF)-based communication is by far the most relevant one as it best fits the requirements of most WSN applications: It provides relatively long range and high data rates, acceptable error rates at reasonable energy expenditure, and does not require line of sight between sender and receiver. Thus, RF-based communication and transceiver will receive the lion share of attention here; other media are only treated briefly at the end of this section.

For a practical wireless, RF-based system, the carrier frequency has to be carefully chosen. Chapter 4 contains a detailed discussion; for the moment, suffice it to say that wireless sensor networks typically use communication frequencies between about 433 MHz and 2.4 GHz.

The reader is expected to be familiar with the basics of wireless communication; a survey is included in Chapter 4.

Transceivers

For actual communication, both a transmitter and a receiver are required in a sensor node. The essential task is to convert a bit stream coming from a microcontroller (or a sequence of bytes or frames) and convert them to and from radio waves. For practical purposes, it is usually convenient to use a device that combines these two tasks in a single entity. Such combined devices are called **transceivers**. Usually, half-duplex operation is realized since transmitting and receiving at the same time on a wireless medium is impractical in most cases (the receiver would only hear the own transmitter anyway).

A range of low-cost transceivers is commercially available that incorporate all the circuitry required for transmitting and receiving – modulation, demodulation, amplifiers, filters, mixers, and so on. For a judicious choice, the transceiver's tasks and its main characteristics have to be understood.

Transceiver tasks and characteristics

To select appropriate transceivers, a number of characteristics should be taken into account. The most important ones are:

Service to upper layer A receiver has to offer certain services to the upper layers, most notably to the Medium Access Control (MAC) layer. Sometimes, this service is **packet oriented**; sometimes, a transceiver only provides a **byte interface** or even only a **bit interface** to the microcontroller.

In any case, the transceiver must provide an interface that somehow allows the MAC layer to initiate frame transmissions and to hand over the packet from, say, the main memory of the sensor node into the transceiver (or a byte or a bit stream, with additional processing required on the microcontroller). In the other direction, incoming packets must be streamed into buffers accessible by the MAC protocol.

Power consumption and energy efficiency The simplest interpretation of energy efficiency is the energy required to transmit and receive a single bit. Also, to be suitable for use in WSNs, transceivers should be switchable between different states, for example, active and sleeping. The idle power consumption in each of these states and during switching between them is very important – details are discussed in Section 2.2.

Carrier frequency and multiple channels Transceivers are available for different carrier frequencies; evidently, it must match application requirements and regulatory restrictions. It is often useful if the transceiver provides several carrier frequencies ("channels") to choose from, helping to alleviate some congestion problems in dense networks. Such channels or "subbands" are relevant, for example, for certain MAC protocols (FDMA or multichannel CSMA/ALOHA techniques, see Chapter 5).

State change times and energy A transceiver can operate in different modes: sending or receiving, use different channels, or be in different power-safe states. In any case, the time and the energy required to change between two such states are important figures of merit. The turnaround time between sending and receiving, for example, is important for various medium access protocols (see Chapter 5).

Data rates Carrier frequency and used bandwidth together with modulation and coding determine the gross data rate. Typical values are a few tens of kilobits per second – considerably less than in broadband wireless communication, but usually sufficient for WSNs. Different data rates can be achieved, for example, by using different modulations or changing the symbol rate.

Modulations The transceivers typically support one or several of on/off-keying, ASK, FSK, or similar modulations. If several modulations are available, it is convenient for experiments if they are selectable at runtime even though, for real deployment, dynamic switching between modulations is not one of the most discussed options.

Coding Some transceivers allow various coding schemes to be selected.

Transmission power control Some transceivers can directly provide control over the transmission power to be used; some require some external circuitry for that purpose. Usually, only a discrete number of power levels are available from which the actual transmission power can be chosen. Maximum output power is usually determined by regulations.

Noise figure The **noise figure** NF of an element is defined as the ratio of the Signal-to-Noise Ratio (SNR) ratio SNR_I at the input of the element to the SNR ratio SNR_O at the element's output:

$$\mathrm{NF} = \frac{\mathrm{SNR}_I}{\mathrm{SNR}_O}$$

It describes the degradation of SNR due to the element's operation and is typically given in dB:

$$\mathrm{NF\,dB} = \mathrm{SNR}_I\ \mathrm{dB} - \mathrm{SNR}_O\ \mathrm{dB}$$

Gain The **gain** is the ratio of the output signal power to the input signal power and is typically given in dB. Amplifiers with high gain are desirable to achieve good energy efficiency.

Power efficiency The **efficiency** of the radio front end is given as the ratio of the radiated power to the overall power consumed by the front end; for a power amplifier, the efficiency describes the ratio of the output signal's power to the power consumed by the overall power amplifier.

Receiver sensitivity The **receiver sensitivity** (given in dBm) specifies the minimum signal power at the receiver needed to achieve a prescribed E_b/N_0 or a prescribed bit/packet error rate. Better sensitivity levels extend the possible range of a system.

Range While intuitively the range of a transmitter is clear, a formal definition requires some care. The range is considered in absence of interference; it evidently depends on the maximum transmission power, on the antenna characteristics, on the attenuation caused by the environment, which in turn depends on the used carrier frequency, on the modulation/coding scheme that is used, and on the bit error rate that one is willing to accept at the receiver. It also depends on the quality of the receiver, essentially captured by its sensitivity. Typical values are difficult to give here, but prototypes or products with ranges between a few meters and several hundreds of meters are available.

Blocking performance The blocking performance of a receiver is its achieved bit error rate in the presence of an interferer. More precisely, at what power level can an interferer (at a fixed distance) send at a given offset from the carrier frequency such that target BER can still be met? An interferer at higher frequency offsets can be tolerated at large power levels. Evidently, blocking performance can be improved by interposing a filter between antenna and transceiver.

An important special case is an adjacent channel interferer that transmits on neighboring frequencies. The adjacent channel suppression describes a transceiver's capability to filter out signals from adjacent frequency bands (and thus to reduce adjacent channel interference) has a direct impact on the observed Signal to Interference and Noise Ratio (SINR).

Out of band emission The inverse to adjacent channel suppression is the out of band emission of a transmitter. To limit disturbance of other systems, or of the WSN itself in a multichannel setup, the transmitter should produce as little as possible of transmission power outside of its prescribed bandwidth, centered around the carrier frequency.

Carrier sense and RSSI In many medium access control protocols, sensing whether the wireless channel, the carrier, is busy (another node is transmitting) is a critical information. The receiver has to be able to provide that information. The precise semantics of this carrier-sense signal depends on the implementation. For example, the IEEE 802.15.4 standard [468] distinguishes the following modes:

- The received energy is above threshold; however, the underlying signal does not need to comply with the modulation and spectral characteristics.
- A carrier has been detected, that is, some signal which complies with the modulation.
- Carrier detected and energy is present.

Also, the signal strength at which an incoming data packet has been received can provide useful information (e.g. a rough estimate about the distance from the transmitter assuming the transmission power is known); a receiver has to provide this information in the Received Signal Strength Indicator (RSSI).

Frequency stability The **frequency stability** denotes the degree of variation from nominal center frequencies when environmental conditions of oscillators like temperature or pressure change. In extreme cases, poor frequency stability can break down communication links, for example, when one node is placed in sunlight whereas its neighbor is currently in the shade.

Voltage range Transceivers should operate reliably over a range of supply voltages. Otherwise, inefficient voltage stabilization circuitry is required.

Transceivers appropriate for WSNs are available from many manufacturers. Usually, there is an entire family of devices to choose from, for example, customized to different regulatory restrictions on carrier frequency in Europe and North America. Currently popular product series include the RFM TR 1001, the Chipcon CC 1000 and CC 2420 (as one of the first IEEE 802.15.4 compliant models), and the Infineon TDA525x family, to name but a few. They are described in a bit more detail at the end of this section.

An important peculiarity and a key difference compared to other communication devices is the fact that these simple transceivers often lack a unique identifier: each Ethernet device, for example, has a MAC-level address that uniquely identifies this individual device. For simple transceivers, the additional cost of providing such an identifier is relatively high with respect to the device's total costs, and thus, unique identifiers cannot be relied upon to be present in all devices. The availability of such device identifiers is very useful in many communication protocols and their absence will have considerable consequences for protocol design.

Improving these commercial designs to provide better performance at lower energy consumption and reduced cost is an ongoing effort by a large research community, facing challenges such as low transistor transconductance or limitations of integrated passive RF components. As these hardware-related questions are not the main focus of this book, the reader is referred to other material [26, 134, 647].

Transceiver structure

A fairly common structure of transceivers is into the Radio Frequency (RF) front end and the baseband part:

- the **radio frequency front end** performs analog signal processing in the actual radio frequency band, whereas
- the **baseband processor** performs all signal processing in the digital domain and communicates with a sensor node's processor or other digital circuitry.

Between these two parts, a frequency conversion takes place, either directly or via one or several Intermediate Frequencys (IFs). The boundary between the analog and the digital domain is constituted by Digital/Analog Converters (DACs) and Analog/Digital Converters (ADCs).

A detailed discussion of the low-power design of RF front end and baseband circuitry is well beyond the scope of this book; one place to start with is reference [3].

The **RF front end** performs analog signal processing in the actual radio frequency band, for example in the 2.4 GHz Industrial, Scientific, and Medical (ISM) band; it is the first stage of the interface between the electromagnetic waves and the digital signal processing of the further transceiver stages [46, 470]. Some important elements of an RF front ends architecture are sketched in Figure 2.2:

- The Power Amplifier (PA) accepts upconverted signals from the IF or baseband part and amplifies them for transmission over the antenna.
- The Low Noise Amplifier (LNA) amplifies incoming signals up to levels suitable for further processing without significantly reducing the SNR [470]. The range of powers of the incoming signals varies from very weak signals from nodes close to the reception boundary to strong signals from nearby nodes; this range can be up to 100 dB. Without management actions, the LNA is active all the time and can consume a significant fraction of the transceiver's energy.
- Elements like local oscillators or voltage-controlled oscillators and mixers are used for frequency conversion from the RF spectrum to intermediate frequencies or to the baseband. The incoming signal at RF frequencies f_{RF} is multiplied in a mixer with a fixed-frequency signal from the local oscillator (frequency f_{LO}). The resulting intermediate-frequency signal has frequency $f_{LO} - f_{RF}$. Depending on the RF front end architecture, other elements like filters are also present.

The efficiency of RF front ends in wireless sensor networks is discussed in Section 4.3.

Transceiver operational states

Many transceivers can distinguish four operational states [670]:

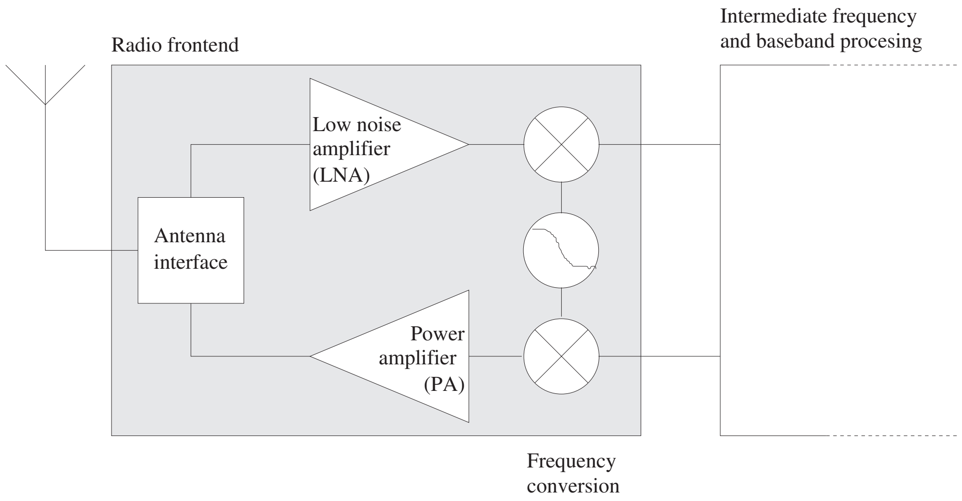

Figure 2.2 RF front end [46]

Transmit In the **transmit state**, the transmit part of the transceiver is active and the antenna radiates energy.

Receive In the **receive state** the receive part is active.

Idle A transceiver that is ready to receive but is not currently receiving anything is said to be in an **idle state**. In this idle state, many parts of the receive circuitry are active, and others can be switched off. For example, in the synchronization circuitry, some elements concerned with acquisition are active, while those concerned with tracking can be switched off and activated only when the acquisition has found something. Myers et al. [580] also discuss techniques for switching off parts of the acquisition circuitry for IEEE 802.11 transceivers. A major source of power dissipation is **leakage**.

Sleep In the **sleep state**, significant parts of the transceiver are switched off. There are transceivers offering several different sleep states, see reference [580] for a discussion of sleep states for IEEE 802.11 transceivers. These sleep states differ in the amount of circuitry switched off and in the associated **recovery times** and **startup energy** [855]. For example, in a complete power down of the transceiver, the startup costs include a complete initialization as well as configuration of the radio, whereas in "lighter" sleep modes, the clock driving certain transceiver parts is throttled down while configuration and operational state is remembered.

The sensor node's protocol stack and operating software must decide into which state the transceiver is switched, according to the current and anticipated communications needs. One problem complicating this decision is that the operation of state changes also dissipate power [670]. For example, a transceiver waking up from the sleep mode to the transmit mode requires some startup time and startup energy, for example, to ramp up phase-locked loops or voltage-controlled oscillators. During this startup time, no transmission or reception of data is possible [762]. The problem of scheduling the node states (equivalently: switching on and off node/transceiver components) so as to minimize average power consumption (also called **power management**) is rather complex, an in-depth treatment can be found in reference [85], and a further reference is [741].

Advanced radio concepts

Apart from these basic transceiver concepts, a number of advanced concepts for radio communication are the objectives of current research. Three of them are briefly summarized here.

Wakeup radio

Looking at the transceiver concepts described above, one of the most power-intensive operations is waiting for a transmission to come in, ready to receive it. During this time, the receiver circuit must be powered up so that the wireless channel can be observed, spending energy without any immediate benefit.

While it seems unavoidable to provide a receiver with power during the actual reception of a packet, it would be desirable not to have to invest power while the node is only waiting for a packet to come in. A receiver structure is necessary that does not need power but can detect when a packet starts to arrive. To keep this specialized receiver simple, it suffices for it to raise an event to notify other components of an incoming packet; upon such an event, the main receiver can be turned on and perform the actual reception of the packet.

Such receiver concepts are called **wakeup receivers** [312, 667, 752, 931, 931]: Their only purpose is to wake up the main receiver without needing (a significant amount of) power to do so – Zhong et al. [931] state a target power consumption of less than 1 µW. In the simplest case, this wakeup would happen for every packet; a more sophisticated version would be able to decide, using proper address information at the start of the packet, whether the incoming packet is actually destined for this node and only then wake up the main receiver.

Such wakeup receivers are tremendously attractive as they would do away with one of the main problems of WSNs: the need to be permanently able to receive in a network with low average traffic. It would considerably simplify a lot of the design problems of WSNs, in particular of the medium access control – Section 5.2.4 will discuss these aspects and some ensuing problems in more detail. Unfortunately, so far the realization of a reliable, well-performing wakeup receiver has not been achieved yet.

Spread-spectrum transceivers

Simple transceiver concepts, based on modulations like Amplitude Shift Keying (ASK) or Frequency Shift Keying (FSK), can suffer from limited performance, especially in scenarios with a lot of interference. To overcome this limitation, the use of spread-spectrum transceivers has been proposed by some researchers [155, 281]. These transceivers, however, suffer mostly from complex hardware and consequently higher prices, which has prevented them from becoming a mainstream concept for WSNs so far. Section 4.2.5 presents details.

Ultrawideband communication

UltraWideBand (UWB) communication is a fairly radical change from conventional wireless communication as outlined above. Instead of modulating a digital signal onto a carrier frequency, a very large bandwidth is used to directly transmit the digital sequence as very short impulses (to form nearly rectangular impulses requires considerable bandwidth, because of which this concept is not used traditionally) [44, 646, 866, 885].[1] Accordingly, these impulses occupy a large spectrum starting from a few Hertz up to the range of several GHz. The challenge is to synchronize sender and receiver sufficiently (to an accuracy of trillionth of seconds) so that the impulses can be correctly detected. A side effect of precisely timed impulses is that UWB is fairly resistant to multipath fading [181, 472], which can be a serious obstacle for carrier-based radio communication.

Using such a large bandwidth, an ultrawideband communication will overlap with the spectrum of a conventional radio system. But, because of the large spreading of the signal, a very small transmission power suffices. This power can be small enough so that it vanishes in the noise floor from the perspective of a traditional radio system.

As one concrete example, consider a time-hopping Pulse Position Modulation (PPM) proposed as combined modulation and multiple access scheme by Win and Scholtz [885]. For each symbol, a number of pulses are transmitted with almost periodic spacing. The deviations from the periodicity encode both the modulation as well as the transmitting user.

For a communication system, the effect is that a very high data rate can be realized over short distances; what is more, UWB communication can relatively easily penetrate obstacles such as doors, which are impermeable to narrowband radio waves. For a WSN, the high data rate is not strictly necessary but can be leveraged to reduce the on-time of the transceivers. The nature of UWB also allows to precisely measure distances (with claimed precision of centimeters).

These desirable features of UWB communication have to be balanced against the difficulties of building such transceivers at low-cost and low-power consumption. More precisely, an UWB transmitter is actually relatively simple since it does not need oscillators or related circuitry found in transmitters for a carrier-frequency-based transmitter. The receivers, on the other hand, require complex timing synchronization. As of this writing, UWB transceivers have not yet been used in prototypes for wireless sensor nodes.

[1] A more precise definition of an ultrawideband system is that it uses at least 500 MHz or a fractional spectrum of at least 20 % of the carrier frequency. This definition would theoretically encompass also spread-spectrum systems with high bandwidth; however, most people have the usage of short pulses in mind when speaking about UWB.

One of the best sources of information about UWB in WSN might be the documents of the IEEE 802.15.4a study group, which looks at UWB as an alternative physical layer for the IEEE 802.15.4 standard for short-range, low bitrate wireless communication. Some references to start from are [82, 187, 566, 603, 884]. A comparison between UWB and Direct Sequence Spread Spectrum (DSSS) technologies for sensor networks has been made in [939], under the assumption of an equal bandwidth for both types of systems.

Nonradio frequency wireless communication

While most of the wireless sensor network work has focused on the use of radio waves as communication media, other options exists. In particular, optical communication and ultrasound communication have been considered as alternatives.

Optical

KAHN et al. [392] and others have considered the use of optical links between sensor nodes. Its main advantage is the very small energy per bit required for both generating and detecting optical light – simple Light-Emitting Diodes (LEDs) are good examples for high-efficiency senders. The required circuitry for an optical transceiver is also simpler and the device as a whole can be smaller than the radio frequency counterpart. Also, communication can take place concurrently with only negligible interference. The evident disadvantage, however, is that communicating peers need to have a line of sight connection and that optical communication is more strongly influenced by weather conditions.

As a case in point, consider the so-called "corner-cube reflector": three mirrors placed at right angles to each other in a way that each beam of light directed at it is reflected back to its source (as long as it comes from a cone centered around the main diagonal of the cube) – an example for such a structure is shown in Figure 2.3. This reflection property holds only as long as the mirrors are exactly at right angles. When one the mirrors is slightly moved, a signal can be modulated onto an incoming ray of light, effectively transmitting information back to the sender. In fact, data rates up to 1 kb/s have been demonstrated using such a device. Its main advantage is that the mechanical movement of one such mirror only takes very little energy, compared to actually generating a beam of light or even a radio wave. Hence, a passive readout of sensor nodes can be done very energy

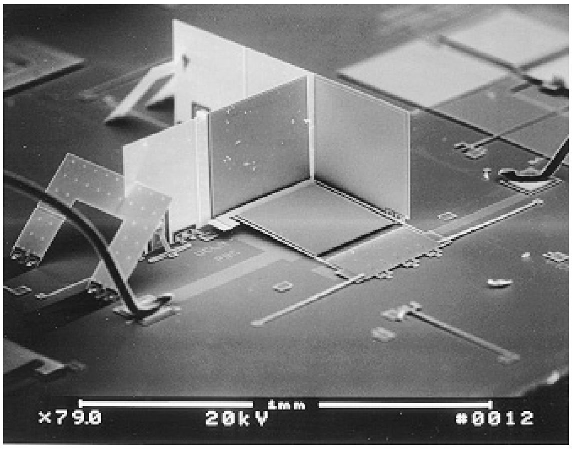

Figure 2.3 Example of a corner-cube reflector for optical communication [168]. Reproduced by permission of IEEE

efficiently over long distances as long as the reader has enough power to produce the laser beam (up to 150 m have been demonstrated using a 5 mW laser).

Ultrasound

Both radio frequency and optical communication are suitable for open-air environments. In some application scenarios, however, sensor nodes are used in environments where radio or optical communication is not applicable because these waves do not penetrate the surrounding medium. One such medium is water, and an application scenario is the surveillance of marine ground floor erosion to help in the construction of offshore wind farms. Sensors are deployed on the marine ground floor and have to communicate amongst themselves. In such an underwater environment, ultrasound is an attractive communication medium as it travels relatively long distances at comparably low power.

A further aspect of ultrasound is its use in location systems as a secondary means of communication with a different propagation speed. Details will be discussed in Chapter 9.

Some examples of radio transceivers

To complete this discussion of possible communication devices, a few examples of standard radio transceivers that are commonly used in various WSN prototype nodes should be briefly described. All these transceivers are in fact commodity, off-the-shelf items available via usual distributors. They are all single-chip solutions, integrating transmitter and receiver functionality, requiring only a small number of external parts and have a fairly low-power consumption. In principle, similar equipment is available from a number of manufacturers – as can be expected, there is not one “best product” available, but each of them has particular advantages and disadvantages.

RFM TR1000 family

The TR1000 family of radio transceivers from RF Monolithics[2] is available for the 916 MHz and 868 MHz frequency range. It works in a 400 kHz wide band centered at, for example, 916.50 MHz. It is intended for short-range radio communication with up to 115.2 kbps. The modulation is either on-off-keying (at a maximum rate of 30 kbps) or ASK; it also provides a dynamically tunable output power. The maximum radiated power is given in the data sheet [690] as 1.5 dBm, $\approx$ 1.4 mW, whereas in the Mica motes a number of 0.75 mW is given [351]. The transceiver offers received signal strength information. It is attractive because of its low-power consumption in both send and receive modes and especially in sleep mode. Details about parameters and configurations can be found in the data sheet [690].

Hardware accelerators (Mica motes)

The Mica motes use the RFM TR1000 transceiver and contain also a set of **hardware accelerators**. On the one hand, the transceiver offers a very low-level interface, giving the microcontroller tight control over frame formats, MAC protocols, and so forth. On the other hand, framing and MAC can be very computation intensive, for example, for computing checksums, for making bytes out of serially received bits or for detecting Start Frame Delimiters (SFDs) in a stream of symbols. The hardware accelerators offer some of these primitive computations in hardware, right at the disposal of the microcontroller.

Chipcon CC1000 and CC2420 family

Chipcon[3] offers a wide range of transceivers that are appealing for use in WSN hardware. To name but two examples: The CC1000 operates in a wider frequency range, between 300 and 1000 MHz,

[2] `http://www.rfm.com`
[3] `http://www.chipcon.com`

programmable in steps of 250 Hz. It uses FSK as modulation, provides RSSI, and has programmable output power. An interesting feature is the possibility to compensate for crystal temperature drift. It should also be possible to use it in frequency hopping protocols. Details can be found in the data sheet[157].

The CC2420 [158] is a more complicated device. It implements the physical layer as prescribed by the IEEE 802.15.4 standard with the required support for this standard's MAC protocol. In fact, the company claims that this is the first commercially available single-chip transceiver for IEEE 802.15.4. As a consequence of implementing this standard, the transceiver operates in the 2.4 GHz band and features the required DSSS modem, resulting in a data rate of 250 kbps. It achieves this at still relatively low-power consumption, although not quite on par with the simpler transceivers described so far.

Infineon TDA 525x family

The Infineon TDA 525x family provides flexible, single-chip, energy-efficient transceivers. The TDA 5250 [375], as an example, is a 868–870 MHztransceiver providing both ASK and FSK modulation, it has a highly efficient power amplifier, RSSI information, a tunable crystal oscillator, an onboard data filter, and an intelligent power-down feature. One of the interesting features is a self-polling mechanism, which can very quickly determine data rate. Compared to some other transceiver, it also has an excellent blocking performance that makes it quite resistant to interference.

IEEE 802.15.4/Ember EM2420 RF transceiver

The IEEE 802.15.4 low-rate Wireless Personal Area Network (WPAN) [468] works in three different frequency bands and employs a DSSS scheme. Some basic data can be found in Table 2.1. For one particular RF front-end design, the Ember[4] EM2420 RF Transceiver [240], some numbers on power dissipation are available. For a radiated power of −0.5 dBm (corresponding to ≈0.9 mW) and with a supply voltage of 3.3 V, the transmit mode draws a current of 22.7 mA, corresponding to ≈74.9 mW, whereas in the receive mode, 25.2 mA current are drawn, corresponding to ≈83.2 mW. In the sleep mode, only 12 µA are drawn.

In all bands, DSSS is used. In the 868 MHz band, only a single channel with a data rate of 20 kbps is available, in the 915 MHz band ten channels of 40kbps each and in the 2.4 GHz band 16 channels of 250 kbps are available. In the lower two bands, the chips are Binary Phase Shift Keying (BPSK)-modulated, and the data symbols are encoded differentially. A pseudonoise sequence of 15 chips is used for every bit. The modulation scheme in the 2.4 GHz band is a little

Table 2.1 The different PHY's of the IEEE 802.15.4 standard [468]. Reproduced by permission of IEEE

Band	868 MHz	915 MHz	2.4 GHz
Frequency [MHz]	868–868.6	902–928	2400–2483.5
Chip rate [kchips/s]	300	600	2000
# of channels	1	10	16
Modulation	BPSK	BPSK	O-QPSK
Data rate [kb/s]	20	40	250
Symbol rate [ksymbols/s]	20	40	62.5
Symbol type	binary	binary	16-ary orthogonal

[4] `http://www.ember.com`

bit more complicated. As can be observed from the table, a channel symbol consists of four user bits. These 16 different symbol values are distinguished by using 16 different nearly orthogonal pseudorandom chip sequences. The resulting chip sequence is then modulated using a modulation scheme called *offset*-Quaternary Phase Shift Keying (QPSK). Some of the design rationale for this modulation scheme is also given in reference [115, Chap. 3].

National Semiconductor LMX3162

The radio hardware of the μAMPS-1 node [563, 762, 872] consists of a digital baseband processor implemented on an FPGA, whereas for the RF front end, a (now obsolete) National Semiconductor LMX3162 transceiver [588] is used. The LMX3162 operates in the 2.4 GHz band and offers six different radiated power levels from 0 dBm up to 20 dBm. To transmit data, the baseband processor can control an externally controllable Voltage-Controlled Oscillator (VCO). The main components of the RF front end (phase-lock loop, transmit and receive circuitry) can be shut off. The baseband processor controls the VCO and also provides timing information to a TDMA-based MAC protocol (see Chapter 5). For data transmission, FSK with a data rate of 1 Mbps is used.

Conexant RDSSS9M

The WINS sensor node of Rockwell[5] carries a Conexant RDSSS9M transceiver, consisting of the RF part working in the ISM band between 902 and 928 MHzand a microcontroller (a 65C02) responsible for processing DSSS signals with a spreading factor of 12 bits per chip. The data rate is 100 kbps. The RF front end offers radiated power levels of 1 mW, 10 mW and 100 mW. A number of 40 sub-bands are available, which can be freely selected. The microcontroller implements portions of a MAC protocol also.

2.1.5 Sensors and actuators

Without the actual sensors and actuators, a wireless sensor network would be beside the point entirely. But as the discussion of possible application areas has already indicated, the possible range of sensors is vast. It is only possible to give a rough idea on which sensors and actuators can be used in a WSN.

Sensors

Sensors can be roughly categorized into three categories (following reference [670]):

Passive, omnidirectional sensors These sensors can measure a physical quantity at the point of the sensor node without actually manipulating the environment by active probing – in this sense, they are passive. Moreover, some of these sensors actually are self-powered in the sense that they obtain the energy they need from the environment – energy is only needed to amplify their analog signal. There is no notion of "direction" involved in these measurements. Typical examples for such sensors include thermometer, light sensors, vibration, microphones, humidity, mechanical stress or tension in materials, chemical sensors sensitive for given substances, smoke detectors, air pressure, and so on.

Passive, narrow-beam sensors These sensors are passive as well, but have a well-defined notion of direction of measurement. A typical example is a camera, which can "take measurements" in a given direction, but has to be rotated if need be.

Active sensors This last group of sensors actively probes the environment, for example, a sonar or radar sensor or some types of seismic sensors, which generate shock waves by small

[5] See `http://wins.rsc.rockwell.com/`.

explosions. These are quite specific – triggering an explosion is certainly not a lightly undertaken action – and require quite special attention.

In practice, sensors from all of these types are available in many different forms with many individual peculiarities. Obvious trade-offs include accuracy, dependability, energy consumption, cost, size, and so on – all this would make a detailed discussion of individual sensors quite ineffective.

Overall, most of the theoretical work on WSNs considers passive, omnidirectional sensors. Narrow-beam-type sensors like cameras are used in some practical testbeds, but there is no real systematic investigation on how to control and schedule the movement of such sensors. Active sensors are not treated in the literature to any noticeable extent.

An assumption occasionally made in the literature [128, 129] is that each sensor node has a certain **area of coverage** for which it can reliably and accurately report the particular quantity that it is observing. More elaborately, a sensor detection model is used, relating the distance between a sensor and the to-be-detected event or object to a detection probability; an example for such a detection model is contained in references [599, 944].

Strictly speaking, this assumption of a coverage area is difficult to justify in its simplest form. Nonetheless, it can be practically useful: It is often possible to postulate, on the basis of application-specific knowledge, some properties of the physical quantity under consideration, in particular, how quickly it can change with respect to distance. For example, temperature or air pressure are unlikely to vary very strongly within a few meters. Hence, allowing for some inevitable inaccuracies in the measurement, the maximum rate of changeover distance can be used to derive such a "coverage radius" within which the values of a single sensor node are considered "good enough". The precise mathematical tools for such a derivation are spatial versions of the sampling theorems.

Actuators

Actuators are just about as diverse as sensors, yet for the purposes of designing a WSN, they are a bit simpler to take account of: In principle, all that a sensor node can do is to open or close a switch or a relay or to set a value in some way. Whether this controls a motor, a light bulb, or some other physical object is not really of concern to the way communication protocols are designed. Hence, in this book, we shall treat actuators fairly summarily without distinguishing between different types.

In a real network, however, care has to be taken to properly account for the idiosyncrasies of different actuators. Also, it is good design practice in most embedded system applications to pair any actuator with a controlling sensor – following the principle to "never trust an actuator" [429].

2.1.6 Power supply of sensor nodes

For untethered wireless sensor nodes, the power supply is a crucial system component. There are essentially two aspects: First, storing energy and providing power in the required form; second, attempting to replenish consumed energy by "scavenging" it from some node-external power source over time.

Storing power is conventionally done using batteries. As a rough orientation, a normal AA battery stores about 2.2–2.5 Ah at 1.5 V. Battery design is a science and industry in itself, and energy scavenging has attracted a lot of attention in research. This section can only provide some small glimpses of this vast field; some papers that deal with these questions (and serve as the basis for this section) are references [134, 392, 667, 670] and, in particular, reference [703].

Storing energy: Batteries

Traditional batteries

The power source of a sensor node is a battery, either nonrechargeable ("primary batteries") or, if an energy scavenging device is present on the node, also rechargeable ("secondary batteries").

Table 2.2 Energy densities for various primary and secondary battery types [703]

Primary batteries			
Chemistry	Zinc-air	Lithium	Alkaline
Energy (J/cm^3)	3780	2880	1200
Secondary batteries			
Chemistry	Lithium	NiMHd	NiCd
Energy (J/cm^3)	1080	860	650

In some form or other, batteries are electro-chemical stores for energy – the chemicals being the main determining factor of battery technology.

Upon these batteries, very tough requirements are imposed:

Capacity They should have high capacity at a small weight, small volume, and low price. The main metric is energy per volume, J/cm^3. Table 2.2 shows some typical values of energy densities, using traditional, macroscale battery technologies. In addition, research on "microscale" batteries, for example, deposited directly onto a chip, is currently ongoing.

Capacity under load They should withstand various usage patterns as a sensor node can consume quite different levels of power over time and actually draw high current in certain operation modes.

Current numbers on power consumption of WSN nodes vary and are treated in detail in Section 2.2, so it is difficult to provide precise guidelines. But for most technologies, the larger the battery, the more power can be delivered instantaneously. In addition, the rated battery capacity specified by a manufacturer is only valid as long as maximum discharge currents are not exceeded, lest capacity drops or even premature battery failure occurs [670].[6]

Self-discharge Their self-discharge should be low; they might also have to last for a long time (using certain technologies, batteries are operational only for a few months, irrespective of whether power is drawn from them or not).

Zinc-air batteries, for example, have only a very short lifetime (on the order of weeks), which offsets their attractively high energy density.

Efficient recharging Recharging should be efficient even at low and intermittently available recharge power; consequently, the battery should also not exhibit any "memory effect".

Some of the energy-scavenging techniques described below are only able to produce current in the μA region (but possibly sustained) at only a few volts at best. Current battery technology would basically not recharge at such values.

Relaxation Their relaxation effect – the seeming self-recharging of an empty or almost empty battery when no current is drawn from it, based on chemical diffusion processes within the cell – should be clearly understood. Battery lifetime and usable capacity is considerably extended if this effect is leveraged. As but one example, it is possible to use multiple batteries in parallel and "schedule" the discharge from one battery to another, depending on relaxation properties and power requirements of the operations to be supported [153].

[6] This effect is due to the need for active material in a battery to be transported to the electrodes. If too much power is drawn, this transport is not fast enough and the battery fails even though energy is still stored in it.

Unconventional energy stores

Apart from traditional batteries, there are also other forms of energy reservoirs that can be contemplated. In a wider sense, fuel cells also qualify as an electro-chemical storage of energy, directly producing electrical energy by oxidizing hydrogen or hydrocarbon fuels. Fuel cells actually have excellent energy densities (e.g. methanol as a fuel stores 17.6 kJ/cm^3), but currently available systems still require a nonnegligible minimum size for pumps, valves, and so on. A slightly more traditional approach to using energy stored in hydrocarbons is to use miniature versions of heat engines, for example, a turbine [243]. Shrinking such heat engines to the desired sizes still requires a considerable research effort in MicroElectroMechanical Systems (MEMSs); predictions regarding power vary between 0.1–10 W at sizes of about 1 cc [703]. And lastly, even radioactive substances have been proposed as an energy store [463]. Another option are so-called "gold caps", high-quality and high-capacity capacitors, which can store relatively large amounts of energy, can be easily and quickly recharged, and do not wear out over time.

DC–DC Conversion

Unfortunately, batteries (or other forms of energy storage) alone are not sufficient as a direct power source for a sensor node. One typical problem is the reduction of a battery's voltage as its capacity drops. Consequently, less power is delivered to the sensor node's circuits, with immediate consequences for oscillator frequencies and transmission power – a node on a weak battery will have a smaller transmission range than one with a full battery, possibly throwing off any calibrations done for the range at full battery ranges.

A DC – DC converter can be used to overcome this problem by regulating the voltage delivered to the node's circuitry. To ensure a constant voltage even though the battery's supply voltage drops, the DC – DC converter has to draw increasingly higher current from the battery when the battery is already becoming weak, speeding up battery death (see Figure 3 in reference [670]). Also, the DC – DC converter does consume energy for its own operation, reducing overall efficiency. But the advantages of predictable operation during the entire life cycle can outweigh these disadvantages.

Energy scavenging

Some of the unconventional energy stores described above – fuel cells, micro heat engines, radioactivity – convert energy from some stored, secondary form into electricity in a less direct and easy to use way than a normal battery would do. The entire energy supply is stored on the node itself – once the fuel supply is exhausted, the node fails.

To ensure truly long-lasting nodes and wireless sensor networks, such a limited energy store is unacceptable. Rather, energy from a node's environment must be tapped into and made available to the node – **energy scavenging** should take place. Several approaches exist [667, 701, 703]:

Photovoltaics The well-known solar cells can be used to power sensor nodes. The available power depends on whether nodes are used outdoors or indoors, and on time of day and whether for outdoor usage. Different technologies are best suited for either outdoor or indoor usage. The resulting power is somewhere between 10 $\mu W/cm^2$ indoors and 15 mW/cm^2 outdoors. Single cells achieve a fairly stable output voltage of about 0.6 V (and have therefore to be used in series) as long as the drawn current does not exceed a critical threshold, which depends, among other factors, on the light intensity. Hence, solar cells are usually used to recharge secondary batteries. Best trade-offs between complexity of recharging circuitry, solar cell efficiency, and battery lifetime are still open questions.

Temperature gradients Differences in temperature can be directly converted to electrical energy. Theoretically, even small difference of, for example, 5 K can produce considerable power, but practical devices fall very short of theoretical upper limits (given by the Carnot efficiency).

Seebeck effect-based thermoelectric generators are commonly considered; one example is a generator, which will be commercially available soon, that achieves about 80 $\mu W/cm^2$ at about 1 V from a 5 Kelvin temperature difference.[7]

Vibrations One almost pervasive form of mechanical energy is vibrations: walls or windows in buildings are resonating with cars or trucks passing in the streets, machinery often has low-frequency vibrations, ventilations also cause it, and so on. The available energy depends on both amplitude and frequency of the vibration and ranges from about 0.1 $\mu W/cm^3$ up to $10,000$ $\mu W/cm^3$ for some extreme cases (typical upper limits are lower).

Converting vibrations to electrical energy can be undertaken by various means, based on electromagnetic, electrostatic, or piezoelectric principles. Figure 2.4 shows, as an example, a generator based on a variable capacitor [549]. Practical devices of 1 cm^3 can produce about 200 $\mu W/cm^3$ from 2.25 m/s^2, 120 Hz vibration sources, actually sufficient to power simple wireless transmitters [702].

Pressure variations Somewhat akin to vibrations, a variation of pressure can also be used as a power source. Such piezoelectric generators are in fact used already. One well-known example is the inclusion of a piezoelectric generator in the heel of a shoe, to generate power as a human walks about [759]. This device can produce, on average, 330 $\mu W/cm^2$. It is, however, not clear how such technologies can be applied to WSNs.

Flow of air/liquid Another often-used power source is the flow of air or liquid in wind mills or turbines. The challenge here is again the miniaturization, but some of the work on millimeter-scale MEMS gas turbines might be reusable [243]. However, this has so far not produced any notable results.

To summarize, Table 2.3 gives an overview of typical values of power and energy densities for different energy sources. The values in this table vary somewhat from those presented above as partially different technologies or environments were assumed; all these numbers can only serve as a general orientation but should always be taken with a grain of salt.

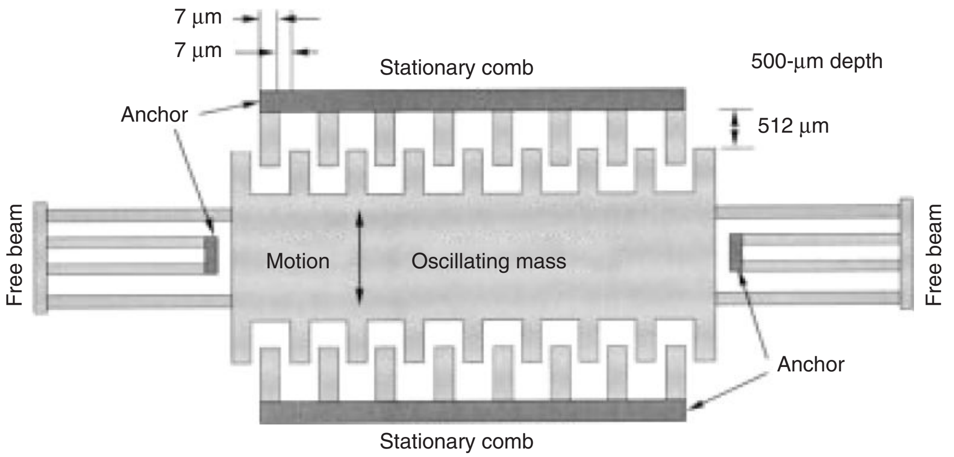

Figure 2.4 A MEMS device for converting vibrations to electrical energy, based on a variable capacitor [549]. Reproduced by permission of IEEE

[7] Compare `http://www.adsx.com`.

Table 2.3 Comparison of energy sources [667]

Energy source	Energy density
Batteries (zinc-air)	1050–1560 mWh/cm^3
Batteries (rechargeable lithium)	300 mWh/cm^3 (at 3–4 V)
Energy source	**Power density**
Solar (outdoors)	15 mW/cm^2 (direct sun)
	0.15 mW/cm^2 (cloudy day)
Solar (indoors)	0.006 mW/cm^2 (standard office desk)
	0.57 mW/cm^2 (<60 W desk lamp)
Vibrations	0.01–0.1 mW/cm^3
Acoustic noise	$3 \cdot 10^{-6}$ mW/cm^2 at 75 dB
	$9,6 \cdot 10^{-4}$ mW/cm^2 at 100 dB
Passive human-powered systems	1.8 mW (shoe inserts)
Nuclear reaction	80 mW/cm^3, 10^6 mWh/cm^3

As these examples show, energy scavenging usually has to be combined with secondary batteries as the actual power sources are not able to provide power consistently, uninterruptedly, at a required level; rather, they tend to fluctuate over time. This requires additional circuitry for recharging of batteries, possibly converting to higher power levels, and a battery technology that can be recharged at low currents. An alternative approach is to align the task execution pattern of the sensor network (which sensor is active when) with the characteristics of energy scavenging – Kansal and Srivastava [399] introduce this idea and describe some protocols and algorithms; they show that the network lifetime is extended by up to 200 % if these scavenging characteristics are taken into account in the task allocation.

2.2 Energy consumption of sensor nodes

2.2.1 Operation states with different power consumption

As the previous section has shown, energy supply for a sensor node is at a premium: batteries have small capacity, and recharging by energy scavenging is complicated and volatile. Hence, the energy consumption of a sensor node must be tightly controlled. The main consumers of energy are the controller, the radio front ends, to some degree the memory, and, depending on the type, the sensors.

To give an example, consider the energy consumed by a microcontroller per instruction. A typical ball park number is about 1 nJ per instruction [391]. To put this into perspective with the battery capacity numbers from Section 2.1.6, assume a battery volume of one cubic millimeter, which is about the maximum possible for the most ambitious visions of “smart dust”. Such a battery could store about 1 J. To use such a battery to power a node even only a single day, the node must not consume continuously more than $1/(24 \cdot 60 \cdot 60)$ Ws/s ≈ 11.5 µW. No current controller, let alone an entire node, is able to work at such low-power levels.

One important contribution to reduce power consumption of these components comes from chip-level and lower technologies: Designing low-power chips is the best starting point for an energy-efficient sensor node. But this is only one half of the picture, as any advantages gained by such designs can easily be squandered when the components are improperly operated.

The crucial observation for proper operation is that most of the time a wireless sensor node has nothing to do. Hence, it is best to turn it off. Naturally, it should be able to wake up again, on the basis of external stimuli or on the basis of time. Therefore, completely turning off a node is not possible, but rather, its operational state can be adapted to the tasks at hand. Introducing and using multiple states of operation with reduced energy consumption in return for reduced functionality is the core technique for energy-efficient wireless sensor node. In fact, this approach is well known even from standard personal computer hardware, where, for example, the Advanced Configuration and Power Interface (ACPI) [8] introduces one state representing the fully operational machine and four sleep states of graded functionality/power consumption/wakeup time (time necessary to return to fully operational state). The term Dynamic Power Management (DPM) summarizes this field of work (see e.g. reference [63] for a slightly older, but quite a broad-range overview).

These modes can be introduced for all components of a sensor node, in particular, for controller, radio front end, memory, and sensors. Different models usually support different numbers of such sleep states with different characteristics; some examples are provided in the following sections. For a controller, typical states are "active", "idle", and "sleep"; a radio modem could turn transmitter, receiver, or both on or off; sensors and memory could also be turned on or off. The usual terminology is to speak of a "deeper" sleep state if less power is consumed.

While such a graded sleep state model is straightforward enough, it is complicated by the fact that transitions between states take both time and energy. The usual assumption is that the deeper the sleep state, the more time and energy it takes to wake up again to fully operational state (or to another, less deep sleep state). Hence, it may be worthwhile to remain in an idle state instead of going to deeper sleep states even from an energy consumption point of view.

Figure 2.5 illustrates this notion based on a commonly used model (used in, e.g. references [558, 769]). At time t_1, the decision whether or not a component (say, the microcontroller) is to be put into sleep mode should be taken to reduce power consumption from P_{active} to P_{sleep}. If it remains active and the next event occurs at time t_{event}, then a total energy of $E_{\text{active}} = P_{\text{active}}(t_{\text{event}} - t_1)$ has be spent uselessly idling. Putting the component into sleep mode, on the other hand, requires a time τ_{down} until sleep mode has been reached; as a simplification, assume that the average power consumption during this phase is $(P_{\text{active}} + P_{\text{sleep}})/2$. Then, P_{sleep} is consumed until t_{event}. In total, $\tau_{\text{down}}(P_{\text{active}} + P_{\text{sleep}})/2 + (t_{\text{event}} - t_1 - \tau_{\text{down}})P_{\text{sleep}}$ energy is required in sleep mode as opposed to $(t_{\text{event}} - t_1)P_{\text{active}}$ when remaining active. The energy saving is thus

$$\begin{aligned} E_{\text{saved}} =& (t_{\text{event}} - t_1)P_{\text{active}} - (\tau_{\text{down}}(P_{\text{active}} + P_{\text{sleep}})/2 + \\ &(t_{\text{event}} - t_1 - \tau_{\text{down}})P_{\text{sleep}}). \end{aligned} \tag{2.1}$$

Once the event to be processed occurs, however, an additional overhead of

$$E_{\text{overhead}} = \tau_{\text{up}}(P_{\text{active}} + P_{\text{sleep}})/2, \tag{2.2}$$

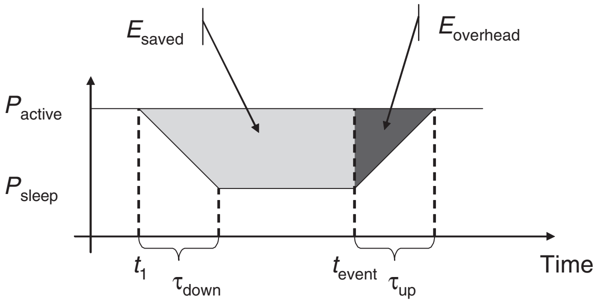

Figure 2.5 Energy savings and overheads for sleep modes

is incurred to come back to operational state before the event can be processed, again making a simplifying assumption about average power consumption during makeup. This energy is indeed an overhead since no useful activity can be undertaken during this time. Clearly, switching to a sleep mode is only beneficial if $E_{\text{overhead}} < E_{\text{saved}}$ or, equivalently, if the time to the next event is sufficiently large:

$$(t_{\text{event}} - t_1) > \frac{1}{2}\left(\tau_{\text{down}} + \frac{P_{\text{active}} + P_{\text{sleep}}}{P_{\text{active}} - P_{\text{sleep}}}\tau_{\text{up}}\right). \tag{2.3}$$

Careful scheduling of such transitions has been considered from several perspectives – reference [769], for example, gives a fairly abstract treatment – and in fact, a lot of medium access control research in wireless sensor networks can be regarded as the problem of when to turn off the receiver of a node.

2.2.2 Microcontroller energy consumption

Basic power consumption in discrete operation states

Embedded controllers commonly implement the concept of multiple operational states as outlined above; it is also fairly easy to control. Some examples probably best explain the idea.

Intel StrongARM
The Intel StrongARM [379] provides three sleep modes:

- In *normal mode*, all parts of the processor are fully powered. Power consumption is up to 400 mW.
- In *idle mode*, clocks to the CPU are stopped; clocks that pertain to peripherals are active. Any interrupt will cause return to normal mode. Power consumption is up to 100 mW.
- In *sleep mode*, only the real-time clock remains active. Wakeup occurs after a timer interrupt and takes up to 160 ms. Power consumption is up to 50 µW.

Texas Instruments MSP 430
The MSP430 family [814] features a wider range of operation modes: One fully operational mode, which consumes about 1.2 mW (all power values given at 1 MHz and 3 V). There are four sleep modes in total. The deepest sleep mode, LPM4, only consumes 0.3 µW, but the controller is only woken up by external interrupts in this mode. In the next higher mode, LPM3, a clock is also still running, which can be used for scheduled wake ups, and still consumes only about 6 µW.

Atmel ATmega
The Atmel ATmega 128L [28] has six different modes of power consumption, which are in principle similar to the MSP 430 but differ in some details. Its power consumption varies between 6 mW and 15 mW in idle and active modes and is about 75 µW in power-down modes.

Dynamic voltage scaling

A more sophisticated possibility than discrete operational states is to use a continuous notion of functionality/power adaptation by adapting the speed with which a controller operates. The idea is to choose the best possible speed with which to compute a task that has to be completed by a given deadline. One obvious solution is to switch the controller in full operation mode, compute the task at highest speed, and go back to a sleep mode as quickly as possible.

The alternative approach is to compute the task only at the speed that is required to finish it before the deadline. The rationale is the fact that a controller running at lower speed, that is, lower clock rates, consumes less power than at full speed. This is due to the fact that the supply voltage can be reduced at lower clock rates while still guaranteeing correct operation. This technique is called Dynamic Voltage Scaling (DVS) [133].

This technique is actually beneficial for CMOS chips: As the actual power consumption P depends quadratically on the supply voltage V_{DD} [649], reducing the voltage is a very efficient way to reduce power consumption. Power consumption also depends on the frequency f, hence $P \propto f \cdot V_{\text{DD}}^2$.

Consequently, dynamic voltage scaling also reduces energy consumption. The Transmeta Crusoe processor, for example, can be scaled from 700 MHz at 1.65 V down to 200 MHz at 1.1 V [649]. This reduces the power consumption by a factor of $\frac{700 \cdot 1.65^2}{200 \cdot 1.1^2} = 7.875$, but the speed is only reduced by a factor of $700/200 = 3.5$. Hence, the energy required per instruction is reduced by $3.5/7.875 \approx 44\,\%$. Other processors and microcontrollers behave similarly, Figure 2.6 shows an example for the StrongARM SA-1100 [558]. The ultimate reason for this improvement is the convex shape of the function power against speed, caused by varying the supply voltage.

When applying dynamic voltage scaling, care has to be taken to operate the controller within its specifications. There are minimum and maximum clock rates for each device, and for each clock rate, there is a minimum and maximum threshold that must be obeyed. Hence, when there is nothing to process, going into sleep modes is still the only option. Also, using arbitrary voltages requires a quite efficient DC-DC converter to be used [134].

How to control DVS from an application or from the operating system is discussed in Section 2.3.4 on page 48.

2.2.3 Memory

From an energy perspective, the most relevant kinds of memory are on-chip memory of a microcontroller and FLASH memory – off-chip RAM is rarely if ever used. In fact, the power needed to drive on-chip memory is usually included in the power consumption numbers given for the controllers.

Hence, the most relevant part is FLASH memory – in fact, the construction and usage of FLASH memory can heavily influence node lifetime. The relevant metrics are the read and write times and

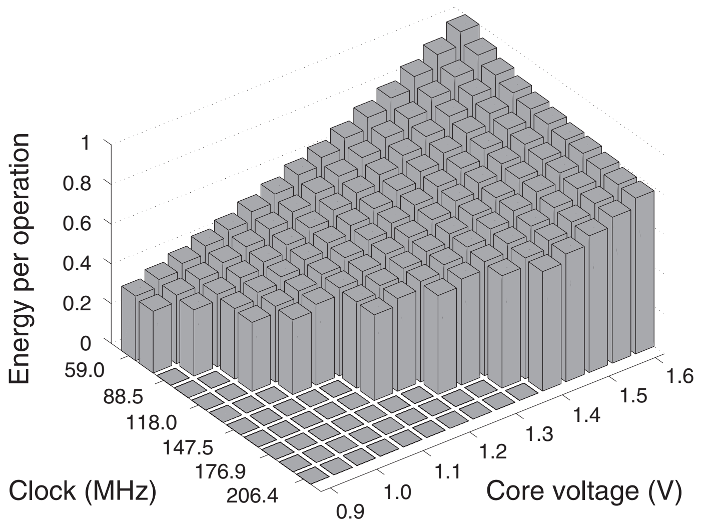

Figure 2.6 Energy per operation with dynamic power scaling on an Intel StrongARM SA-1100 [558]. Reproduced by permission of IEEE

energy consumption. All this information is readily available from manufacturers' data sheets and do vary depending on several factors. Read times and read energy consumption tend to be quite similar between different types of FLASH memory [329]. Writing is somewhat more complicated, as it depends on the granularity with which data can be accessed (individual bytes or only complete pages of various sizes). One means for comparability is to look at the numbers for overwriting the whole chip. Considerable differences in erase and write energy consumption exist, up to ratios of 900:1 between different types of memory [329].

To give a concrete example, consider the energy consumption necessary for reading and writing to the Flash memory used on the Mica nodes [534]. Reading data takes 1.111 nAh, writing requires 83.333 nAh.

Hence, writing to FLASH memory can be a time- and energy-consuming task that is best avoided if somehow possible. For detailed numbers, it is necessary to consult the documentation of the particular wireless sensor node and its FLASH memory under consideration.

2.2.4 Radio transceivers

A radio transceiver has essentially two tasks: transmitting and receiving data between a pair of nodes. Similar to microcontrollers, radio transceivers can operate in different modes, the simplest ones are being turned on or turned off. To accommodate the necessary low total energy consumption, the transceivers should be turned off most of the time and only be activated when necessary – they work at a low **duty cycle**. But this incurs additional complexity, time and power overhead that has to be taken into account.

To understand the energy consumption behavior of radio transceivers and their impact on the protocol design, models for the energy consumption per bit for both sending and receiving are required. Several such models of different accuracy and level of detail exist and are mostly textbook knowledge [14, 661, 682, 938] (for a research paper example see reference [762]); the presentation here mostly follows reference [559], in particular, with respect to concrete numbers.

Modeling energy consumption during transmission

In principle, the energy consumed by a transmitter is due to two sources [670]: one part is due to RF signal generation, which mostly depends on chosen modulation and target distance and hence on the transmission power P_{tx}, that is, the power radiated by the antenna. A second part is due to electronic components necessary for frequency synthesis, frequency conversion, filters, and so on. These costs are basically constant.

One of the most crucial decisions when transmitting a packet is thus the choice of P_{tx}. Chapter 4 will discuss some of the factors involved in such a decision; controlling the transmission power will also play a role in several other chapters of Part II. For the present discussion, let us assume that the desired transmission power P_{tx} is known – Chapter 4 will make it clear that P_{tx} is a function of system aspects like energy per bit over noise E_b/N_0, the bandwidth efficiency η_{BW}, the distance d and the path loss coefficient γ.

The transmitted power is generated by the amplifier of a transmitter. Its own power consumption P_{amp} depends on its architecture, but for most of them, their consumed power depends on the power they are to generate. In the most simplistic models, these two values are proportional to each other, but this is an oversimplification. A more realistic model assumes that a certain constant power level is always required irrespective of radiated power, plus a proportional offset:

$$P_{\text{amp}} = \alpha_{\text{amp}} + \beta_{\text{amp}} P_{\text{tx}}. \qquad (2.4)$$

where α_{amp} and β_{amp} are constants depending on process technology and amplifier architecture [559].

As an example, Min and Chandrakasan [563] report, for the μAMPS-1 nodes, $\alpha_{\text{amp}} = 174\,\text{mW}$ and $\beta_{\text{amp}} = 5.0$. Accordingly, the **efficiency of the power amplifier** η_{PA} for $P_{\text{tx}} = 0$ dBm $= 1$ mW radiated power is given by

$$\eta_{\text{PA}} = \frac{P_{\text{tx}}}{P_{\text{amp}}} = \frac{1\,\text{mW}}{174\,\text{mW} + 5.0 \cdot 1\,\text{mW}} \approx 0.55\,\%.$$

This model implies that the amplifier's efficiency $P_{\text{tx}}/P_{\text{amp}}$ is best at maximum output power. Maximum power is, however, not necessarily the common case and therefore such a design is not necessarily the most beneficial one – in cellular systems, for example, amplifiers often do not operate at their maximum output power. While it is not clear how this observation would translate to WSNs, it appears promising especially in dense networks to use amplifiers with different efficiency characteristics [447]. Nonetheless, here we shall restrict the attention to the model of Equation (2.4).

In addition to the amplifier, other circuitry has to be powered up during transmission as well, for example, baseband processors. This power is referred to as P_{txElec}.

The energy to transmit a packet n-bits long (including all headers) then depends on how long it takes to send the packet, determined by the nominal bit rate R and the coding rate R_{code}, and on the total consumed power during transmission. If, in addition, the transceiver has to be turned on before transmission, startup costs also are incurred (mostly to allow voltage-controlled oscillators and phase-locked loops to settle). Equation (2.5) summarizes these effects.

$$E_{\text{tx}}(n, R_{\text{code}}, P_{\text{amp}}) = T_{\text{start}} P_{\text{start}} + \frac{n}{R R_{\text{code}}} (P_{\text{txElec}} + P_{\text{amp}}). \tag{2.5}$$

It should be pointed out that this equation does not depend on the modulation chosen for transmission (Section 4.3 will discuss in detail an example containing multiple modulations). Measurements based on IEEE 802.11 hardware [221] have shown that in fact there is a slight dependence on the modulation, but the difference between 1 Mbit/s and 11 Mbit/s is less than 10 % for all considered transmission power values, so this is an acceptable simplification. Moreover, it is assumed that the coding overhead only depends on the coding rate, which is an acceptable assumption. In this model, the antenna efficiency is missing as well, that is, it is assumed to have a perfect antenna. Otherwise, there would be further power losses between the output of the PA and the radiated power.

This model can be easily enhanced by the effects of Forward Error Correction (FEC) coding since, with respect to transmission, FEC just increases the number of bits approximately by a factor of one divided by the code rate (see Chapter 6), since the coding energy is negligible [563].

Disregarding the distance-independent terms in these energy costs and only assuming a simplified energy cost proportional to some power of the distance has been called "one of the top myths" of energy consumption in radio communication [560]. Clearly, choosing such an inappropriately simplified model would have considerable consequences on system design, for example, incorrectly favoring a multihop approach (see Chapter 3).

Modeling energy consumption during reception

Similar to the transmitter, the receiver can be either turned off or turned on. While being turned on, it can either actively receive a packet or can be idle, observing the channel and ready to receive. Evidently, the power consumption while it is turned off is negligible. Even the difference between idling and actually receiving is very small and can, for most purposes, be assumed to be zero.

To elucidate, the energy E_{rcvd} required to receive a packet has a startup component $T_{\text{start}} P_{\text{start}}$ similar to the transmission case when the receiver had been turned off (startup times are considered equal for transmission and receiving here); it also has a component that is proportional to the packet time $\frac{n}{RR_{\text{code}}}$. During this time of actual reception, receiver circuitry has to be powered up, requiring a (more or less constant) power of P_{rxElec} – for example, to drive the LNA in the RF front end. The last component is the decoding overhead, which is incurred for every bit – this decoding overhead can be substantial depending on the concrete FEC in use; Section 6.2.3 goes into details here. Equation (2.6) summarizes these components.

$$E_{\text{rcvd}} = T_{\text{start}} P_{\text{start}} + \frac{n}{RR_{\text{code}}} P_{\text{rxElec}} + n E_{\text{decBit}}. \tag{2.6}$$

The decoding energy is relatively complicated to model, as it depends on a number of hardware and system parameters – for example, is decoding done in dedicated hardware (by, for example, a dedicated Viterbi decoder for convolutional codes) or in software on a microcontroller; it also depends on supply voltage, decoding time per bit (which in turn depends on processing speed influenced by techniques like DVS), constraint length K of the used code, and other parameters. MIN and CHANDRAKASAN [559] give more details.

Again, it is worthwhile pointing out that different modulation schemes only implicitly affect this result via the increase in time to transmit the packet.

Some numbers

Providing concrete numbers for exemplary radio transceivers is even more difficult than it is for microcontrollers: The range of commercially available transceivers is vast, with many different characteristics. Transceivers that appear to have excellent energy characteristics might suffer from other shortcomings like poor frequency stability under temperature variations (leading to partitioning of a network when parts of the node are placed in the shade and others in sunlight), poor blocking performance, high susceptibility to interference on neighboring frequency channels, or undesirable error characteristics; they could also lack features that other transceivers have, like tunability to multiple frequencies. Hence, the numbers presented here should be considered very cautiously, even more so since they had been collected from different sources and were likely determined in noncomparable environments (and not all numbers are available for all examples). Still, they should serve to provide some impression of current performance figures for actual hardware.

Table 2.4 summarizes the parameters discussed here for a number of different nodes. These numbers have been collected from references [670] and [563];[8] the data sheets [588, 690] offer further information. Note that the way of reporting such figures in the literature is anything but uniform and that hence many of the numbers given here had to be calculated or estimated. The reader is encouraged to check with the original publications for full detail. In particular, the data about the WINS and MEDUSA-II node do not allow to distinguish between α_{amp} and P_{txElec} and β_{amp} is estimated by curve fitting. STEMM and KATZ [789] present additional data for some older hardware geared toward handheld devices. References [351, 353, 725, 769] also contain further examples for sensor nodes; FEENEY and NILSSON [254] and EBERT et al. [221] present actual measurement results for IEEE 802.11-based hardware. One useful reference number for rule-of-thumb estimations might be the 1 µJ required to transmit a single bit and 0.5 µJ to receive one for the RFM TR1000 transceiver [353].

Looking at the startup times in Table 2.4, we see that actually considerable time and energy can be spent to turn on a transceiver. CHANDRAKASAN et al. [134] argue therefore that architectures with short startup times are preferable and point out the impact of startup time on the energy per bit when using different modulations; they also propose an appropriate transceiver architecture with

[8] Since these numbers are likely obtained by different measurement methods, they are not directly comparable and the reader must be cautious. However, at least they give useful ballpark estimates.

Table 2.4 Some parameters of transceiver energy consumption

Symbol	Description	Example transceiver		
		μAMPS-1 [559]	WINS [670]	MEDUSA-II [670]
α_{amp}	Equation (2.4)	174 mW	N/A	N/A
β_{amp}	Equation (2.4)	5.0	8.9	7.43
P_{amp}	Amplifier pwr.	179–674 mW	N/A	N/A
P_{rxElec}	Reception pwr.	279 mW	368.3 mW	12.48 mW
P_{rxIdle}	Receive idle	N/A	344.2 mW	12.34 mW
P_{start}	Startup pwr.	58.7 mW	N/A	N/A
P_{txElec}	Transmit pwr.	151 mW	$\approx$ 386 mW	11.61 mW
R	Transmission rate	1 Mbps	100 kbps	OOK 30 kbps ASK 115.2 kbps
T_{start}	Startup time	466 µs	N/A	N/A

fast startup time. Wang et al. [855] also point this out and provide figures on how startup time influences the choice between modulations.

These startup costs motivate some considerations of the entire system architecture. One possible idea is to have only very simply functionalities on line that can handle most of the processing, for example, decide whether a packet is intended for a given node, and only startup other components, for example, the controller, if necessary [648]. Clearly, wakeup radios are the most advanced version of this concept. Naturally, startup costs also have to be taken into account during protocol design.

Another common observation based on these figures is that transmitting and receiving have comparable power consumption, at least for short-range communication [648]. Details differ, of course, but it is an acceptable approximation to assume $P_{\text{txElec}} = P_{\text{rxElec}}$ and even neglecting the amplifier part can be admissible as long as very low transmission powers are used. In fact, for some architectures, receiving consumes more power than transmitting.

Chandrakasan et al. [132] summarize these numbers into an energy per bit versus bitrate figure, pointing out that energy efficiency improves as transmission rates go up if duty cycling is used on the radio.

Dynamic scaling of radio power consumption

Applying controller-based Dynamic Voltage Scaling (DVS) principles to radio transceivers as well is tempting, but nontrivial. Scaling down supply voltage or frequency to obtain lower power consumption in exchange for higher latency is only applicable to some of the electronic parts of a transceiver, but this would mean that the remainder of the circuitry – the amplifier, for instance, which cannot be scaled down as its radiated and hence its consumed power mostly depends on the communication distance – still has to be run at high power over an extended period of time [670].

However, the frequency/voltage versus performance trade-off exploited in DVS is not the only possible trade-off to exploit. Any such "parameter versus performance" trade-off that has a convex characteristic should be amenable to an analogous optimization technique. For radio communication, in particular, possible parameters include the choice of modulation and/or code, giving raise to Dynamic Modulation Scaling (DMS), Dynamic Code Scaling (DCS) and Dynamic Modulation-Code Scaling (DMCS) optimization techniques [449, 559, 650, 735, 738]. The claim that such trade-offs do not apply to communication is another one of the "myths" of energy consumption in communication [560].

The idea of these approaches is to dynamically adapt modulation, coding, or other parameters to maximize system metrics like throughput or, particularly relevant here, energy efficiency. It rests on the hardware's ability to actually perform such modulation adaptations, but this is a commonly found property of modern transceivers. In addition, delay constraints and time-varying radio channel properties have to be taken into account.

The details of these approaches are somewhat involved, and partially, complicated optimization problems have to be approximately solved. The required computational effort should not be underestimated and a combined analysis should be undertaken on how best to split up energy consumption. Nonetheless, these approaches are quite beneficial in energy efficiency terms.

2.2.5 Relationship between computation and communication

Looking at the energy consumption numbers for both microcontrollers and radio transceivers, an evident question to ask is which is the best way to invest the precious energy resources of a sensor node: Is it better to send data or to compute? What is the relation in energy consumption between sending data and computing?

Again, details about this relationship heavily depend on the particular hardware in use, but a few rule-of-thumb figures can be given here. Typically, computing a single instruction on a microcontroller requires about 1 nJ. Also, 1 nJ about suffices to take a single sample in a radio transceiver; Bluetooth transceivers could be expected to require roughly 100 nJ to transmit a single bit (disregarding issues like startup cost and packet lengths) [391]. For other hardware, the ratio of the energy consumption to send one bit compared to computing a single instruction is between 1500 to 2700 for Rockwell WINS nodes, between 220 to 2900 for MEDUSA II nodes, and about 1400 for WINS NG 2.0 nodes [670]. HILL et al. [353] notes, for the RFM TR1000 radio transceiver, 1 μJ to transmit a single bit and 0.5 μJ to receive one; their processor takes about 8 nJ per instruction. This results in a (actually quite good) ratio of about 190 for communication to computation costs. In a slightly different perspective, communicating 1 kB of data over 100 m consumes roughly the same amount of energy as computing three million instructions [648]. HILL and CULLER [351] give some more numbers for specific applications.

Disregarding the details, it is clear that communication is a considerably more expensive undertaking than computation. Still, energy required for computation cannot be simply ignored; depending on the computational task, it is usually still smaller than the energy for communication, but still noticeable. This basic observation motivates a number of approaches and design decisions for the networking architecture of wireless sensor networks. The core idea is to invest into computation within the network whenever possible to safe on communication costs, leading to the notion of *in-network processing* and *aggregation*. These ideas will be discussed in detail in Chapter 3.

2.2.6 Power consumption of sensor and actuators

Providing any guidelines about the power consumption of the actual sensors and actuators is next to impossible because of the wide diversity of these devices. For some of them – for example, passive light or temperature sensors – the power consumption can perhaps be ignored in comparison to other devices on a wireless node (although HILL et al. [353] report a power consumption of 0.6 to 1 mA for a temperature sensor). For others, in particular, active devices like sonar, power consumption can be quite considerable and must even be considered in the dimensioning of power sources on the sensor node, not to overstress batteries, for example. To derive any meaningful numbers, requires a look at the intended application scenarios and the intended sensors to be used. Some hints on power consumption of sensor/controller interfaces, namely, AD converters, can be found in reference [26].

In addition, the sampling rate evidently is quite important. Not only does more frequent sampling require more energy for the sensors as such but also the data has to processed and, possibly, communicated somewhere.

3

Network architecture

Objectives of this Chapter

This chapter introduces the basic principles of turning individual sensor nodes into a wireless sensor network. On the basis of the high-level application scenarios of Chapter 1, more concrete scenarios and the resulting optimization goals of how a network should function are discussed. On the basis of these scenarios and goals, a few principles for the design of networking protocols in wireless sensor networks are derived – these principles and the resulting protocol mechanisms constitute the core differences of WSNs compared to other network types. To make the resulting capabilities of a WSN usable, a proper service interface is required, as is an integration of WSNs into larger network contexts.

At the end of this chapter, the reader should be able to appreciate the basic networking "philosophy" followed by wireless sensor network research. Upon this basis, the next part of the book will then discuss in detail individual networking functionalities.

Chapter Outline

The architecture of wireless sensor networks draws upon many sources. Historically, a lot of related work has been done in the context of self-organizing, mobile, ad hoc networks (references [635, 793, 827] provide some overview material). While these networks are intended for different purposes, they share the need for a decentralized, distributed form of organization. From a different perspective, sensor networks are related to real-time computing [429, 514] and even to some concepts from peer-to-peer computing [55, 480, 574, 608, 842], active networks [111], and mobile agents/swarm intelligence [86, 98, 176, 220, 892].

Consequently, the number of ideas and publications on networking architectures for wireless sensor networks is vast, and it is often difficult to clearly attribute who first came up with a certain idea, especially since many of them are fairly obvious extrapolations of ideas from the areas just mentioned; also, similar concepts have often been proposed more or less concurrently by different authors. Nonetheless, proper attribution shall be given where possible. A (not necessarily complete) collection of important architectural papers on wireless sensor networks is [26, 88, 126, 134, 233, 245, 246, 274, 342, 344, 351, 353, 392, 433, 500, 534, 648, 653, 667, 758, 778, 788, 798, 921, 923]; pointers and discussion of architectural issues are also included in practically all overview papers, for example, [17, 367, 670, 699].

3.1 Sensor network scenarios

3.1.1 Types of sources and sinks

Section 1.3 has introduced several typical interaction patterns found in WSNs – event detection, periodic measurements, function approximation and edge detection, or tracking – it has also already briefly touched upon the definition of "sources" and "sinks". A source is any entity in the network that can provide information, that is, typically a sensor node; it could also be an actuator node that provides feedback about an operation.

A sink, on the other hand, is the entity where information is required. There are essentially three options for a sink: it could belong to the sensor network as such and be just another sensor/actuator node or it could be an entity outside this network. For this second case, the sink could be an actual device, for example, a handheld or PDA used to interact with the sensor network; it could also be merely a gateway to another larger network such as the Internet, where the actual request for the information comes from some node "far away" and only indirectly connected to such a sensor network. These main types of sinks are illustrated by Figure 3.1, showing sources and sinks in direct communication.

For much of the remaining discussion, this distinction between various types of sinks is actually fairly irrelevant. It is important, as discussed in Section 3.1.4, whether sources or sinks move, but what they do with the information is not a primary concern of the networking architecture. There are some consequences of a sink being a gateway node; they will be discussed in Section 3.5.

3.1.2 Single-hop versus multihop networks

From the basics of radio communication and the inherent power limitation of radio communication follows a limitation on the feasible distance between a sender and a receiver. Because of this

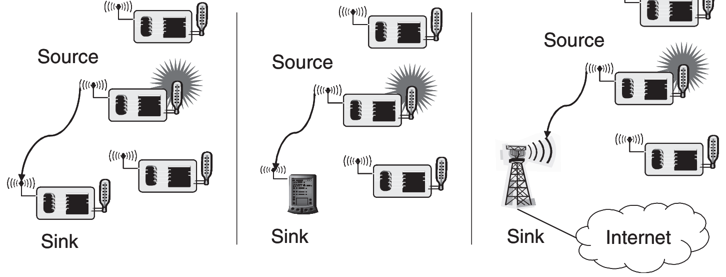

Figure 3.1 Three types of sinks in a very simple, single-hop sensor network

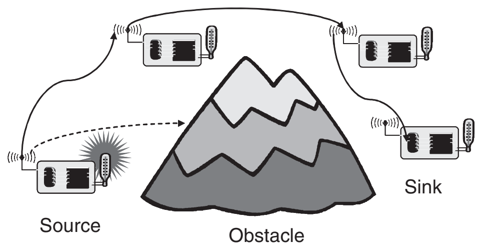

Figure 3.2 Multihop networks: As direct communication is impossible because of distance and/or obstacles, multihop communication can circumvent the problem

limited distance, the simple, direct communication between source and sink is not always possible, specifically in WSNs, which are intended to cover a lot of ground (e.g. in environmental or agriculture applications) or that operate in difficult radio environments with strong attenuation (e.g. in buildings).

To overcome such limited distances, an obvious way out is to use relay stations, with the data packets taking multi hops from the source to the sink. This concept of multihop networks (illustrated in Figure 3.2) is particularly attractive for WSNs as the sensor nodes themselves can act as such relay nodes, foregoing the need for additional equipment. Depending on the particular application, the likelihood of having an intermediate sensor node at the right place can actually be quite high – for example, when a given area has to be uniformly equipped with sensor nodes anyway – but nevertheless, there is not always a guarantee that such multihop routes from source to sink exist, nor that such a route is particularly short.

While multihopping is an evident and working solution to overcome problems with large distances or obstacles, it has also been claimed to improve the energy efficiency of communication. The intuition behind this claim is that, as attenuation of radio signals is at least quadratic in most environments (and usually larger), it consumes less energy to use relays instead of direct communication: When targeting for a constant SNR at all receivers (assuming for simplicity negligible error rates at this SNR), the *radiated* energy required for direct communication over a distance d is cd^α (c some constant, $\alpha \geq 2$ the path loss coefficient); using a relay at distance $d/2$ reduces this energy to $2c(d/2)^\alpha$.

But this calculation considers only the radiated energy, not the actually *consumed* energy – in particular, the energy consumed in the intermediate relay node. Even assuming that this relay belongs to the WSN and is willing to cooperate, when computing the total required energy it is necessary to take into account the complete power consumption of Section 2.2.4. It is an easy exercise to show that energy is actually wasted if intermediate relays are used for short distances d. Only for large d does the radiated energy dominate the fixed energy costs consumed in transmitter and receiver electronics – the concrete distance where direct and multihop communication are in balance depends on a lot of device-specific and environment-specific parameters. Nonetheless, this relationship is often not considered. In fact, MIN and CHANDRAKASAN [560] classify the misconception that multihopping saves energy as the number one myth about energy consumption in wireless communication. Great care should be taken when applying multihopping with the end of improved energy efficiency.

It should be pointed out that only multihop networks operating in a **store and forward** fashion are considered here. In such a network, a node has to correctly receive a packet before it can forward it somewhere. Alternative, innovative approaches attempt to exploit even erroneous reception of packets, for example, when multiple nodes send the same packet and each individual transmission could not be received, but collectively, a node can reconstruct the full packet. Such **cooperative relaying** techniques are not considered here.

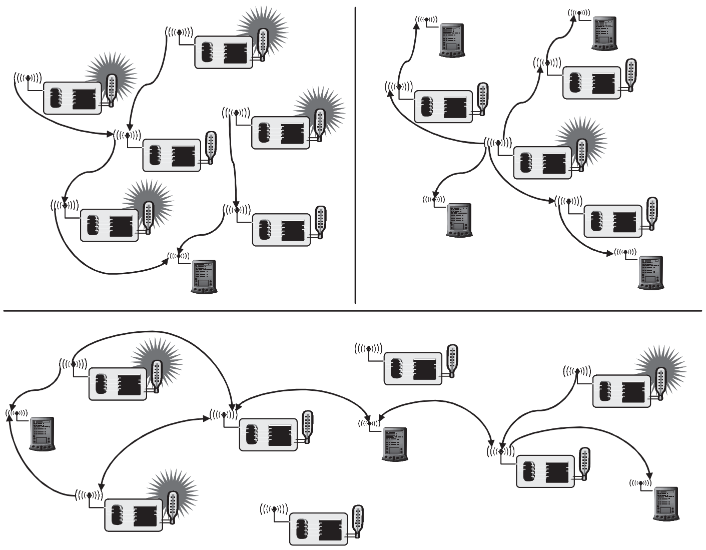

Figure 3.3 Multiple sources and/or multiple sinks. Note how in the scenario in the lower half, both sinks and active sources are used to forward data to the sinks at the left and right end of the network

3.1.3 Multiple sinks and sources

So far, only networks with a single source and a single sink have been illustrated. In many cases, there are multiple sources and/or multiple sinks present. In the most challenging case, multiple sources should send information to multiple sinks, where either all or some of the information has to reach all or some of the sinks. Figure 3.3 illustrates these combinations.

3.1.4 Three types of mobility

In the scenarios discussed above, all participants were stationary. But one of the main virtues of wireless communication is its ability to support mobile participants. In wireless sensor networks, mobility can appear in three main forms:

Node mobility The wireless sensor nodes themselves can be mobile. The meaning of such mobility is highly application dependent. In examples like environmental control, node mobility should not happen; in livestock surveillance (sensor nodes attached to cattle, for example), it is the common rule.

In the face of node mobility, the network has to reorganize itself frequently enough to be able to function correctly. It is clear that there are trade-offs between the frequency and speed of node movement on the one hand and the energy required to maintain a desired level of functionality in the network on the other hand.

Sink mobility The information sinks can be mobile (Figure 3.4). While this can be a special case of node mobility, the important aspect is the mobility of an information sink that is not part of the sensor network, for example, a human user requested information via a PDA while walking in an intelligent building.

In a simple case, such a requester can interact with the WSN at one point and complete its interactions before moving on. In many cases, consecutive interactions can be treated as

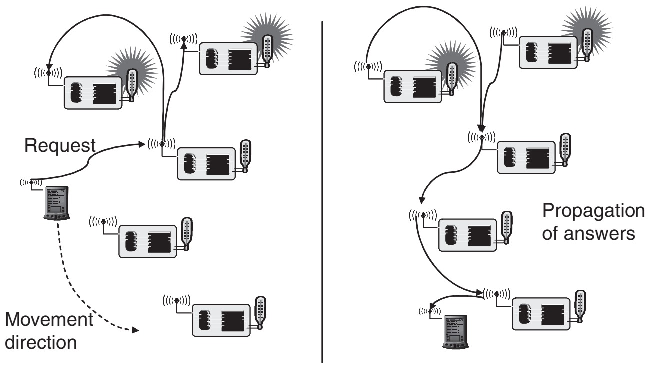

Figure 3.4 A mobile sink moves through a sensor network as information is being retrieved on its behalf

separate, unrelated requests. Whether the requester is allowed interactions with any node or only with specific nodes is a design choice for the appropriate protocol layers.

A mobile requester is particularly interesting, however, if the requested data is not locally available but must be retrieved from some remote part of the network. Hence, while the requester would likely communicate only with nodes in its vicinity, it might have moved to some other place. The network, possibly with the assistance of the mobile requester, must make provisions that the requested data actually follows and reaches the requester despite its movements [758].

Event mobility In applications like event detection and in particular in tracking applications, the cause of the events or the objects to be tracked can be mobile.

In such scenarios, it is (usually) important that the observed event is covered by a sufficient number of sensors at all time. Hence, sensors will wake up around the object, engaged in higher activity to observe the present object, and then go back to sleep. As the event source moves through the network, it is accompanied by an area of activity within the network – this has been called the *frisbee* model, introduced in reference [126] (which also describes algorithms for handling the “wakeup wavefront”). This notion is described by Figure 3.5, where the task is to detect a moving elephant and to observe it as it moves around. Nodes that do not actively detect anything are intended to switch to lower sleep states unless they are required to convey information from the zone of activity to some remote sink (not shown in Figure 3.5).

Communication protocols for WSNs will have to render appropriate support for these forms of mobility. In particular, event mobility is quite uncommon, compared to previous forms of mobile or wireless networks.

3.2 Optimization goals and figures of merit

For all these scenarios and application types, different forms of networking solutions can be found. The challenging question is how to optimize a network, how to compare these solutions, how to decide which approach better supports a given application, and how to turn relatively imprecise optimization goals into measurable figures of merit? While a general answer appears impossible considering the large variety of possible applications, a few aspects are fairly evident.

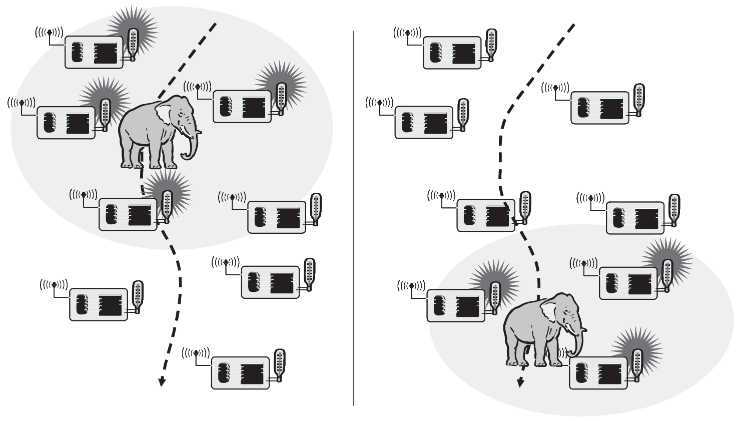

Figure 3.5 Area of sensor nodes detecting an event – an elephant [378] – that moves through the network along with the event source (dashed line indicate the elephant's trajectory; shaded ellipse the activity area following or even preceding the elephant)

3.2.1 Quality of service

WSNs differ from other conventional communication networks mainly in the type of service they offer. These networks essentially only move bits from one place to another. Possibly, additional requirements about the offered Quality of Service (QoS) are made, especially in the context of multimedia applications. Such QoS can be regarded as a low-level, networking-device-observable attribute – bandwidth, delay, jitter, packet loss rate – or as a high-level, user-observable, so-called subjective attribute like the perceived quality of a voice communication or a video transmission. While the first kind of attributes is applicable to a certain degree to WSNs as well (bandwidth, for example, is quite unimportant), the second one clearly is not, but is really the more important one to consider! Hence, high-level QoS attributes corresponding to the subjective QoS attributes in conventional networks are required.

But just like in traditional networks, high-level QoS attributes in WSN highly depend on the application. Some generic possibilities are:

Event detection/reporting probability What is the probability that an event that actually occurred is not detected or, more precisely, not reported to an information sink that is interested in such an event? For example, not reporting a fire alarm to a surveillance station would be a severe shortcoming.

Clearly, this probability can depend on/be traded off against the overhead spent in setting up structures in the network that support the reporting of such an event (e.g. routing tables) or against the run-time overhead (e.g. sampling frequencies).

Event classification error If events are not only to be detected but also to be classified, the error in classification must be small.

Event detection delay What is the delay between detecting an event and reporting it to any/all interested sinks?

Missing reports In applications that require periodic reporting, the probability of undelivered reports should be small.

Approximation accuracy For function approximation applications (e.g. approximating the temperature as a function of location for a given area), what is the average/maximum absolute or relative error with respect to the actual function?[1] Similarly, for edge detection applications, what is the accuracy of edge descriptions; are some missed at all?

Tracking accuracy Tracking applications must not miss an object to be tracked, the reported position should be as close to the real position as possible, and the error should be small. Other aspects of tracking accuracy are, for example, the sensitivity to sensing gaps [923].

3.2.2 Energy efficiency

Much of the discussion has already shown that energy is a precious resource in wireless sensor networks and that energy efficiency should therefore make an evident optimization goal. It is clear that with an arbitrary amount of energy, most of the QoS metrics defined above can be increased almost at will (approximation and tracking accuracy are notable exceptions as they also depend on the density of the network). Hence, putting the delivered QoS and the energy required to do so into perspective should give a first, reasonable understanding of the term energy efficiency.

The term "energy efficiency" is, in fact, rather an umbrella term for many different aspects of a system, which should be carefully distinguished to form actual, measurable figures of merit. The most commonly considered aspects are:

Energy per correctly received bit How much energy, counting all sources of energy consumption at all possible intermediate hops, is spent on average to transport one bit of information (payload) from the source to the destination? This is often a useful metric for periodic monitoring applications.

Energy per reported (unique) event Similarly, what is the average energy spent to report one event? Since the same event is sometimes reported from various sources, it is usual to normalize this metric to only the unique events (redundant information about an already known event does not provide additional information).

Delay/energy trade-offs Some applications have a notion of "urgent" events, which can justify an increased energy investment for a speedy reporting of such events. Here, the trade-off between delay and energy overhead is interesting.

Network lifetime The time for which the network is operational or, put another way, the time during which it is able to fulfill its tasks (starting from a given amount of stored energy). It is not quite clear, however, when this time ends. Possible definitions are:

Time to first node death When does the first node in the network run out of energy or fail and stop operating?

Network half-life When have 50 % of the nodes run out of energy and stopped operating? Any other fixed percentile is applicable as well.

Time to partition When does the first partition of the network in two (or more) disconnected parts occur? This can be as early as the death of the first node (if that was in a pivotal position) or occur very late if the network topology is robust.

[1] Clearly, this requires assumptions about the function to be approximated; discontinuous functions or functions with unlimited first derivative are impossible to approximate with a finite number of sensors.

Time to loss of coverage Usually, with redundant network deployment and sensors that can observe a region instead of just the very spot where the node is located, each point in the deployment region is observed by multiple sensor nodes. A possible figure of merit is thus the time when for the first time any spot in the deployment region is no longer covered by any node's observations.

If *k* redundant observations are necessary (for tracking applications, for example), the corresponding definition of loss of coverage would be the first time any spot in the deployment region is no longer covered by at least *k* different sensor nodes.

Time to failure of first event notification A network partition can be seen as irrelevant if the unreachable part of the network does not want to report any events in the first place. Hence, a possibly more application-specific interpretation of partition is the inability to deliver an event. This can be due to an event not being noticed because the responsible sensor is dead or because a partition between source and sink has occurred.

It should be noted that simulating network lifetimes can be a difficult statistical problem.

Obviously, the longer these times are, the better does a network perform. More generally, it is also possible to look at the (complementary) distribution of node lifetimes (with what probability does a node survive a given amount of time?) or at the relative survival times of a network (at what time are how many percent of the nodes still operational?). This latter function allows an intuition about many WSN-specific protocols in that they tend to sacrifice long lifetimes in return for an improvement in short lifetimes – they "sharpen the drop" (Figure 3.6).

All these metrics can of course only be evaluated under a clear set of assumptions about the energy consumption characteristics of a given node, about the actual "load" that the network has to deal with (e.g. when and where do events happen), and also about the behavior of the radio channel.

3.2.3 Scalability

The ability to maintain performance characteristics irrespective of the size of the network is referred to as scalability. With WSN potentially consisting of thousands of nodes, scalability is an evidently indispensable requirement. Scalability is ill served by any construct that requires globally consistent state, such as addresses or routing table entries that have to be maintained. Hence, the need to restrict such information is enforced by and goes hand in hand with the resource limitations of sensor nodes, especially with respect to memory.

The need for extreme scalability has direct consequences for the protocol design. Often, a penalty in performance or complexity has to be paid for small networks as discussed in the following Section 3.3.1. Architectures and protocols should implement *appropriate* scalability support rather than trying to be as scalable as possible. Applications with a few dozen nodes might admit more-efficient solutions than applications with thousands of nodes; these smaller applications might be

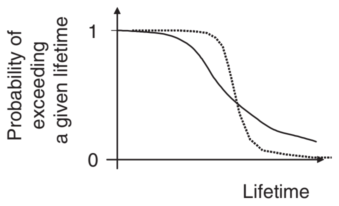

Figure 3.6 Two probability curves of a node exceeding a given lifetime – the dotted curve trades off better minimal lifetime against reduced maximum lifetime

more common in the first place. Nonetheless, a considerable amount of research has been invested into highly scalable architectures and protocols.

3.2.4 Robustness

Related to QoS and somewhat also to scalability requirements, wireless sensor networks should also exhibit an appropriate robustness. They should not fail just because a limited number of nodes run out of energy, or because their environment changes and severs existing radio links between two nodes – if possible, these failures have to be compensated for, for example, by finding other routes. A precise evaluation of robustness is difficult in practice and depends mostly on failure models for both nodes and communication links.

3.3 Design principles for WSNs

Appropriate QoS support, energy efficiency, and scalability are important design and optimization goals for wireless sensor networks. But these goals themselves do not provide many hints on how to structure a network such that they are achieved. A few basic principles have emerged, which can be useful when designing networking protocols; the description here follows partially references [246, 699]. Nonetheless, the general advice to always consider the needs of a concrete application holds here as well – for each of these basic principles, there are examples where following them would result in inferior solutions.

3.3.1 Distributed organization

Both the scalability and the robustness optimization goal, and to some degree also the other goals, make it imperative to organize the network in a distributed fashion. That means that there should be no centralized entity in charge – such an entity could, for example, control medium access or make routing decisions, similar to the tasks performed by a base station in cellular mobile networks. The disadvantages of such a centralized approach are obvious as it introduces exposed points of failure and is difficult to implement in a radio network, where participants only have a limited communication range. Rather, the WSNs nodes should cooperatively organize the network, using distributed algorithms and protocols. **Self-organization** is a commonly used term for this principle.

When organizing a network in a distributed fashion, it is necessary to be aware of potential shortcomings of this approach. In many circumstances, a centralized approach can produce solutions that perform better or require less resources (in particular, energy). To combine the advantages, one possibility is to use centralized principles in a localized fashion by dynamically electing, out of the set of equal nodes, specific nodes that assume the responsibilities of a centralized agent, for example, to organize medium access. Such elections result in a hierarchy, which has to be dynamic: The election process should be repeated continuously lest the resources of the elected nodes be overtaxed, the elected node runs out of energy, and the robustness disadvantages of such – even only localized – hierarchies manifest themselves. The particular election rules and triggering conditions for reelection vary considerably, depending on the purpose for which these hierarchies are used. Chapter 10 will, to a large degree, deal with the question of how to determine such hierarchies in a distributed fashion.

3.3.2 In-network processing

When organizing a network in a distributed fashion, the nodes in the network are not only passing on packets or executing application programs, they are also actively involved in taking decisions about how to operate the network. This is a specific form of information processing that happens in the network, but is limited to information about the network itself. It is possible to extend this concept by also taking the concrete data that is to be transported by the network into account in this information processing, making **in-network processing** a first-rank design principle.

Several techniques for in-network processing exist, and by definition, this approach is open to an arbitrary extension – any form of data processing that improves an application is applicable. A few example techniques are outlined here; they will reappear in various of the following chapters, especially in Chapter 12.

Aggregation

Perhaps the simplest in-network processing technique is aggregation. Suppose a sink is interested in obtaining periodic measurements from all sensors, but it is only relevant to check whether the average value has changed, or whether the difference between minimum and maximum value is too big. In such a case, it is evidently not necessary to transport are readings from all sensors to the sink, but rather, it suffices to send the average or the minimum and maximum value. Recalling from Section 2.3 that transmitting data is considerably more expensive than even complex computation shows the great energy-efficiency benefits of this approach. The name **aggregation** stems from the fact that in nodes intermediate between sources and sinks, information is aggregated into a condensed form out of information provided by nodes further away from the sink (and potentially, the aggregator's own readings).

Clearly, the aggregation function to be applied in the intermediate nodes must satisfy some conditions for the result to be meaningful; most importantly, this function should be **composable**. A further classification [528] of aggregate functions distinguishes *duplicate-sensitive* versus *insensitive*, *summary* versus *exemplary*, *monotone* versus *nonmonotone*, and *algebraic* versus *holistic* (a more detailed discussion can be found in Section 12.3). Functions like average, counting, or minimum can profit a lot from aggregation; holistic functions like the median are not amenable to aggregation at all.

Figure 3.7 illustrates the idea of aggregation. In the left half, a number of sensors transmit readings to a sink, using multihop communication. In total, 13 messages are required (the numbers in the figure indicate the number of messages traveling across a given link). When the highlighted nodes perform aggregation – for example, by computing average values (shown in the right half of the figure) – only 6 messages are necessary.

Challenges in this context include how to determine where to aggregate results from which nodes, how long to wait for such results, and determining the impact of lost packets.

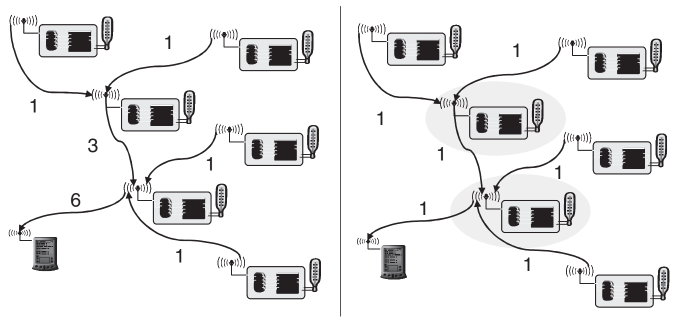

Figure 3.7 Aggregation example

Distributed source coding and distributed compression

Aggregation condenses and sacrifices information about the measured values in order not to have to transmit all bits of data from all sources to the sink. Is it possible to reduce the number of transmitted bits (compared to simply transmitting all bits) but still obtain the *full* information about all sensor readings at the sink?

While this question sounds surprising at first, it is indeed possible to give a positive answer. It is related to the coding and compression problems known from conventional networks, where a lot of effort is invested to encode, for example, a video sequence, to reduce the required bandwidth [901]. The problem here is slightly different, in that we are interested to encode the information provided by several sensors, not just by a single camera; moreover, traditional coding schemes tend to put effort into the encoding, which might be too computationally complex for simple sensor nodes.

How can the fact that information is provided by multiple sensors be exploited to help in coding? If the sensors were connected and could exchange their data, this would be conceivable (using relatively standard compression algorithms), but of course pointless. Hence, some implicit, joint information between two sensors is required. Recall here that these sensors are embedded in a physical environment – it is quite likely that the readings of adjacent sensors are going to be quite similar; they are *correlated*. Such **correlation** can indeed be exploited such that not simply the sum of the data must be transmitted but that overhead can be saved here. The theoretical basis is the theorem by Slepian and Wolf [774], which carries their name. Good overview papers are references [653, 901].

Slepian-Wolf theorem–based work is an example of exploiting spatial correlation that is commonly present in sensor readings, as long as the network is sufficiently dense, compared to the derivate of the observed function and the degree of correlation between readings at two places. Similarly, **temporal correlation** can be exploited in sensor network protocols.

Distributed and collaborative signal processing

The in-networking processing approaches discussed so far have not really used the ability for *processing* in the sensor nodes, or have only used this for trivial operations like averaging or finding the maximum. When complex computations on a certain amount of data is to be done, it can still be more energy efficient to compute these functions on the sensor nodes despite their limited processing power, if in return the amount of data that has to be communicated can be reduced.

An example for this concept is the distributed computation of a Fast Fourier Transform (FFT) [152]. Depending on where the input data is located, there are different algorithms available to compute an FFT in a distributed fashion, with different trade-offs between local computation complexity and the need for communication. In principle, this is similar to algorithm design for parallel computers. However, here not only the latency of communication but also the energy consumption of communication and computation are relevant parameters to decide between various algorithms.

Such distributed computations are mostly applicable to signal processing type algorithms; typical examples are beamforming and target tracking applications. Zhao and Guibas [924] provide a good overview of this topic.

Mobile code/Agent-based networking

With the possibility of executing programs in the network, other programming paradigms or computational models are feasible. One such model is the idea of **mobile code** or **agent-based networking**. The idea is to have a small, compact representation of program code that is small enough to be sent from node to node. This code is then executed locally, for example, collecting measurements, and then decides where to be sent next. This idea has been used in various environments; a classic example is that of a software agent that is sent out to collect the best possible travel itinerary by hopping from one travel agent's computer to another and eventually returning to the user who has posted this inquiry. There is a vast amount of literature available on mobile code/software agents in general, see, for example, references [98, 176, 892]. A newer take on this approach is to consider biologically inspired systems, in particular, the **swarm intelligence** of groups of simple entities, working together to reach a common goal [86, 220].

In wireless sensor networks, mobile agents and related concepts have been considered in various contexts, mostly with respect to routing of queries and for data fusion; see, for example, references [96, 99, 207, 663, 664, 665, 829]. Also, virtual machines for WSNs have been proposed that have a native language that admits a compact representation of the most typical operations that mobile code in a WSN would execute, allowing this code to be small [481].

3.3.3 Adaptive fidelity and accuracy

Section 2.3.4 has already discussed, in the context of a single node, the notion of making the fidelity of computation results contingent upon the amount of energy available for that particular computation. This notion can and should be extended from a single node to an entire network [246].

As an example, consider a function approximation application. Clearly, when more sensors participate in the approximation, the function is sampled at more points and the approximation is better. But in return for this, more energy has to be invested. Similar examples hold for event detection and tracking applications and in general for WSNs.

Hence, it is up to an application to somehow define the degree of accuracy of the results (assuming that it can live with imprecise, approximated results) and it is the task of the communication protocols to try to achieve at least this accuracy as energy efficiently as possible. Moreover, the application should be able to adapt its requirements to the current status of the network – how many nodes have already failed, how much energy could be scavenged from the environment, what are the operational conditions (have critical events happened recently), and so forth. Therefore, the application needs feedback from the network about its status to make such decisions.

But as already discussed in the context of WSN-specific QoS metrics, the large variety of WSN applications makes it quite challenging to come up with a uniform interface for expressing such requirements, let alone with communication protocols that implement these decisions. This is still one of the core research problems of WSN.

3.3.4 Data centricity

Address data, not nodes

In traditional communication networks, the focus of a communication relationship is usually the pair of communicating peers – the sender and the receiver of data. In a wireless sensor network, on the other hand, the interest of an application is not so much in the *identity* of a particular sensor node, it is much rather in the actual information reported about the physical environment. This is especially the case when a WSN is redundantly deployed such that any given event could be reported by multiple nodes – it is of no concern to the application precisely which of these nodes is providing data. This fact that not the identity of nodes but the data are at the center of attention is called **data-centric networking**. For an application, this essentially means that an interface is exposed by the network where data, not nodes, is addressed in requests. The set of nodes that is involved in such a *data-centric address* is implicitly defined by the property that a node can contribute data to such an address.

As an example, consider the elephant-tracking example from Figure 3.5. In a data-centric application, all the application would have to do is state its desire to be informed about events of a certain type – "presence of elephant" – and the nodes in the network that possess "elephant detectors" are implicitly informed about this request. In an identity-centric network, the requesting node would have to find out somehow all nodes that provide this capability and address them explicitly. As another example, it is useful to consider the location of nodes as a property that defines whether a node belongs to a certain group or not. The typical example here is the desire to communicate with all nodes in a given area, say, to retrieve the (average) temperature measured by all nodes in the living room of a given building.

Data-centric networking allows very different networking architectures compared to traditional, identity-centric networks. For one, it is the ultimate justification for some in-network processing techniques like data fusion and aggregation. Data-centric addressing also enables simple expressions of communication relationships – it is no longer necessary to distinguish between one-to-one, one-to-many, many-to-one, or many-to-many relationships as the set of participating nodes is only implicitly defined. In addition to this decoupling of identities, data-centric addressing also supports a decoupling in time as a request to provide data does not have to specify when the answer should happen – a property that is useful for event-detection applications, for example.

Apart from providing a more natural way for an application to express its requirements, data-centric networking and addressing is also claimed to improve performance and especially energy efficiency of a WSN. One reason is the hope that data-centric solutions scale better by being implementable using purely local information about direct neighbors. Another reason could be the easier integration of a notion of adaptive accuracy into a data-centric framework as the data as well as its desired accuracy can be explicitly expressed – it is not at all clear how stating accuracy requirements in an identity-centric network could even be formulated, let alone implemented. But this is still an objective of current research.

Implementation options for data-centric networking

There are several possible ways to make this abstract notion of data-centric networks more concrete. Each way implies a certain set of interfaces that would be usable by an application. The three most important ones are briefly sketched here and partially discussed in more detail in later chapters.

Overlay networks and distributed hash tables

There are some evident similarities between well-known peer-to-peer applications [55, 480, 574, 608, 842] like file sharing and WSN: In both cases, the user/requester is interested only in looking up and obtaining data, not in its source; the request for data and its availability can be decoupled in time; both types of networks should scale to large numbers.

In peer-to-peer networking, the solution for an efficient lookup of retrieval of data from an unknown source is usually to form an overlay network, implementing a Distributed Hash Table (DHT) [686, 704, 792, 922]. The desired data can be identified via a given key (a hash) and the DHT will provide one (or possibly several) sources for the data associated with this key. The crucial point is that this data source lookup can be performed efficiently, requiring $O(\log n)$ steps where n is the number of nodes, even with only distributed, localized information about where information is stored in the peer-to-peer network.

Despite these similarities, there are some crucial differences. First of all, it is not clear how the rather static key of a DHT would correspond to the more dynamic, parameterized requests in a WSN. Second, and more importantly, DHTs, coming from an IP-networking background, tend to ignore the distance/the hop count between two nodes and consider nodes as adjacent only on the basis of semantic information about their stored keys. This hop-count-agnostic behavior is unacceptable for WSNs where each hop incurs considerable communication overhead. There is some on-going work on taking the topology of the underlying network also into account [460, 683, 846] or the position of nodes [685, 760] when constructing the overlay network, but the applicability of this work to WSN is still open. Chapter 12 will deal with these approaches in more detail.

Publish/Subscribe

The required separation in both time and identity of a sink node asking for information and the act of providing this information is not well matched with the synchronous characteristics of a request/reply protocol. What is rather necessary is a means to express the need for certain data and the delivery of the data, where the data as such is specified and not the involved entities.

This behavior is realized by the **publish/subscribe** approach [251]: Any node interested in a given kind of data can *subscribe* to it, and any node can *publish* data, along with information about its kind as well. Upon a publication, all subscribers to this kind of data are notified of the new data. The elephant example is then easily expressed by sink nodes subscribing to the event "elephant detected"; any node that is detecting an elephant can then, at any later time, publish this event. If a subscriber is no longer interested, it can simply *unsubscribe* from any kind of event and will no longer be notified of such events. Evidently, subscription and publication can happen at different points in time and the identities of subscribers and publishers do not have to be known to each other.

Implementing this abstract concept of publishing and subscribing to information can be done in various ways. One possibility is to use a central entity where subscriptions and publications are matched to each other, but this is evidently inappropriate for WSNs. A distributed solution is preferable but considerably more complicated.

Also relevant is the expressiveness of the data descriptions (their "names") used to match publications and subscriptions. A first idea is to use explicit subjects or keywords as names, which have to be defined up front – published data only matches to subscriptions with the same keyword (like in the "elephant detected" example above). This subject-based approach can be extended into hierarchical schemes where subjects are arranged in a tree; a subscription to a given subject then also implies interest in any descendent subjects. A more general naming scheme allows to formulate the matching condition between subscriptions and publications as general predicates over the content of the publication and is hence referred to as **content-based publish/subscribe** approach (see e.g. reference [123] and the references therein for an introduction and overview).

In practice, general predicates on the content are somewhat clumsy to handle and restricted expressions (also called *filters*) of the form `(attribute, value, operator)` are preferable, where `attribute` corresponds to the subjects from above (e.g. `temperature`) and can assume values, `value` is a concrete value like `"25°C"` or a placeholder (`ALL` or `ANY`), and `operator` is a relational operator like "=", "<", "≤". Moreover, this formalism also lends itself very conveniently to the expression of accuracy requirements or periodic measurement support.

The question remains where to send publication and subscription messages if a decentralized approach is chosen – simply flooding all messages evidently defeats the purpose. MÜHL et al. [576] give an overview of various approaches, for example, flooding the subscriptions or exploiting information contained in the content-based filters to limit propagation of messages [121, 122, 575].

Publish/subscribe networking is a very popular approach for WSN. In fact, some of the most popular protocols are incarnations of this principle and are discussed in detail in Part II, in particular in Chapter 12.

Databases

A somewhat different view on WSN is to consider them as (dynamic) databases [269, 303, 374, 887]. This view matches very well with the idea of using a data-centric organization of the networking protocols. Being interested in certain aspects of the physical environment that is surveyed by a WSN is equivalent to formulating queries for a database.

To cast the sensor networks into the framework of relational databases, it is useful to regard the sensors as a virtual table to which relational operators can be applied. Then, extracting the average temperature reading from all sensors in a given room can be simply written as shown in Listing 3.1 [528] – it should come as no surprise to anybody acquainted with the Standard Query Language (SQL).

Listing 3.1: Example of an SQL-based request for sensor readings [528]

```
SELECT AVG(temperature)
FROM sensors
WHERE location = "Room 123"
```

Such SQL-based querying of a WSN can be extended to an easy-to-grasp interface to wireless sensor networks, being capable of expressing most salient interaction patterns with a WSN. It is, however, not quite as clear how to translate this interface into actual networking protocols that implement this interface and can provide the results for such queries. In a traditional relational database, this implementation of a query is done by determining an execution plan; the same is necessary here. Here, however, the execution plan has to be distributed and has to explicitly take communication costs into account.

3.3.5 Exploit location information

Another useful technique is to exploit location information in the communication protocols whenever such information is present. Since the location of an event is a crucial information for many applications, there have to be mechanisms that determine the location of sensor nodes (and possibly also that of observed events) – they are discussed in detail in Chapter 9. Once such information is available, it can simplify the design and operation of communication protocols and can improve their energy efficiency considerably. We shall see various examples in different protocols in Part II.

3.3.6 Exploit activity patterns

Activity patterns in a wireless sensor network tend to be quite different from traditional networks. While it is true that the data rate averaged over a long time can be very small when there is only very rarely an event to report, this can change dramatically when something does happen. Once an event has happened, it can be detected by a larger number of sensors, breaking into a frenzy of activity, causing a well-known event shower effect. Hence, the protocol design should be able to handle such bursts of traffic by being able to switch between modes of quiescence and of high activity.

3.3.7 Exploit heterogeneity

Related to the exploitation of activity patterns is the exploitation of heterogeneity in the network. Sensor nodes can be heterogenous by constructions, that is, some nodes have larger batteries, farther-reaching communication devices, or more processing power. They can also be heterogenous by evolution, that is, all nodes started from an equal state, but because some nodes had to perform more tasks during the operation of the network, they have depleted their energy resources or other nodes had better opportunities to scavenge energy from the environment (e.g. nodes in shade are at a disadvantage when solar cells are used).

Whether by construction or by evolution, heterogeneity in the network is both a burden and an opportunity. The opportunity is in an asymmetric assignment of tasks, giving nodes with more resources or more capabilities the more demanding tasks. For example, nodes with more memory or faster processors can be better suited for aggregation, nodes with more energy reserves for hierarchical coordination, or nodes with a farther-reaching radio device should invest their energy mostly for long-distance communication, whereas, shorter-distance communication can be undertaken by the other nodes. The burden is that these asymmetric task assignments cannot usually be static but have to be reevaluated as time passes and the node/network state evolves. Task reassignment in turn is an activity that requires resources and has to be balanced against the potential benefits.

3.3.8 Component-based protocol stacks and cross-layer optimization

Finally, a consideration about the implementation aspects of communication protocols in WSNs is necessary. Section 2.3.3 has already made the case for a component-based as opposed to a layering-based model of protocol implementation in WSN. What remains to be defined is mainly a default collection of components, not all of which have to be always available at all times on all sensor nodes, but which can form a basic "toolbox" of protocols and algorithms to build upon.

In fact, most of the chapters of Part II are about such building blocks. All wireless sensor networks will require some – even if only simple – form of physical, MAC and link layer[2] protocols; there will be wireless sensor networks that require routing and transport layer functionalities. Moreover, "helper modules" like time synchronization, topology control, or localization can be useful. On top of these "basic" components, more abstract functionalities can then be built. As a consequence, the set of components that is active on a sensor node can be complex, and will change from application to application.

Protocol components will also interact with each other in essentially two different ways [330]. One is the simple exchange of data packets as they are passed from one component to another as it is processed by different protocols. The other interaction type is the exchange of cross-layer information.

This possibility for cross-layer information exchange holds great promise for protocol optimization, but is also not without danger. Kawadia and Kumar [412], for example, argue that imprudent use of cross-layer designs can lead to feedback loops, endangering both functionality and performance of the entire system. Clearly, these concerns should not be easily disregarded and care has to be taken to avoid such unexpected feedback loops.

3.4 Service interfaces of WSNs

3.4.1 Structuring application/protocol stack interfaces

Looking at Section 2.3's discussion of a component-based operating system and protocol stack already enables one possibility to treat an application: It is just another component that can directly interact with other components using whatever interface specification exists between them (e.g. the command/event structure of TinyOS). The application could even consist of several components,

[2] While these components do not form a layer in the strict sense of the word, it will still be useful to refer to the corresponding functionality as that of a, say, "physical layer".

integrated at various places into the protocol stack. This approach has several advantages: It is streamlined with the overall protocol structure, makes it easy to introduce application-specific code into the WSN at various levels, and does not require the definition of an abstract, specific service interface. Moreover, such a tight integration allows the application programmer a very fine-grained control over which protocols (which components) are chosen for a specific task; for example, it is possible to select out of different routing protocols the one best suited for a given application by accessing this component's services.

But this generality and flexibility is also the potential downside of this approach. The allowing of the application programmer to mess with protocol stacks and operating system internals should not be undertaken carelessly. In traditional networks such as the Internet, the application programmer can access the services of the network via a commonly accepted interface: sockets [791]. This interface makes clear provisions on how to handle connections, how to send and receive packets, and how to inquire about state information of the network.[3] This clarity is owing to the evident tasks that this interface serves – the exchange of packets with one (sometimes, several) communication peers.

Therefore, there is the design choice between treating the application as just another component or designing a service interface that makes all components, in their entirety, accessible in a standardized fashion. These two options are outlined by Figure 3.8. A service interface would allow to raise the level of abstraction with which an application can interact with the WSN – instead of having to specify which value to read from which particular sensor, it might be desirable to provide an application with the possibility to express sensing tasks in terms that are close to the semantics of the application. In this sense, such a service interface can hide considerable complexity and is actually conceivable as a "middleware" in its own right.

Clearly, with a tighter integration of the application into the protocol stack, a broader optimization spectrum is open to the application programmer. On the downside, more experience will be necessary than when using a standardized service interface. The question is therefore on the one hand the price of standardization with respect to the potential loss of performance and on the other hand, the complexity of the service interface.

In fact, the much bigger complexity and variety of communication patterns in wireless sensor networks compared to Internet networks makes a more expressive and potentially complex service interface necessary. To better understand this trade-off, a clearer understanding of expressibility requirements of such an interface is necessary.

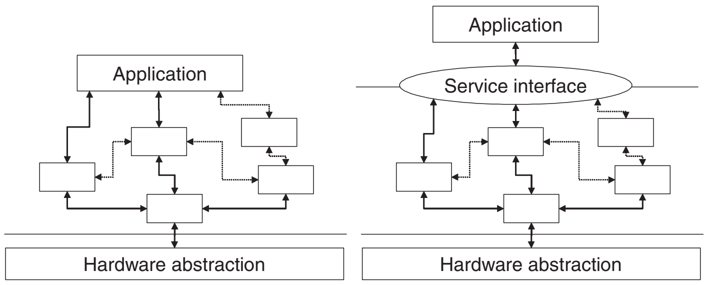

Figure 3.8 Two options for interfacing an application to a protocol stack: As just another component or via a deliberately designed, general service interface

[3] It is certainly correct to argue that the socket interface has its shortcomings and open issues, especially with regard to wireless communication. But these issues are mostly related to the wish to access lower-layer information, for example, received signal strength information, which is not directly exposed by the interface, but only via various, nonstandard workarounds.

3.4.2 Expressibility requirements for WSN service interfaces

The most important functionalities that a service interface should expose include:

- Support for simple request/response interactions: retrieving a measured value from some sensor or setting a parameter in some node. This is a synchronous interaction pattern in the sense that the result (or possibly the acknowledgment) is expected immediately. In addition, the responses can be required to be provided periodically, supporting periodic measurement-type applications.
- Support for asynchronous event notifications: a requesting node can require the network to inform it if a given condition becomes true, for example, if a certain event has happened. This is an asynchronous pattern in the sense that there is no a priori relationship between the time the request is made and the time the information is provided.

  This form of asynchronous requests should be accompanied by the possibility to cancel the request for information. It can be further refined by provisions about what should happen after the condition becomes true; a typical example is to request periodic reporting of measured values after an event.
- For both types of interactions, the addressees should be definable in several ways. The simplest option is an explicit enumeration of the single or multiple communication peers to whom a (synchronous or asynchronous) request is made – this corresponds to the peer address in a socket communication.

  More interesting is the question of how to express data centricity. One option, closely related to the publish/subscribe approach discussed in Section 3.3.4, is the implicit definition of peers by some form of a membership function of an abstract group of nodes. Possible examples for such membership functions include:
  - Location – all nodes that are in a given region of space belong to a group.
  - Observed value – all nodes that have observed values matching a given predicate belong to a group. An example would be to require the measured temperature to be larger than 20°C.

  Along with these groups, the usual set-theoretic operations of intersection, union, or difference between groups should be included in the service interface as well.

  Because of this natural need for a service interface semantics that corresponds to the publish/subscribe concept, this approach is a quite natural, but not the only possible, fit with WSNs.
- In-networking processing functionality has to be accessible. For an operation that accesses an entire group of nodes, especially when reading values from this group (either synchronously or asynchronously), it should be possible to specify what kind of in-network processing should be applied to it. In particular, processing that modifies the nature of the result (i.e., data fusion) must be explicitly allowed by the requesting application.

  In addition, it can be desirable for an application to be able to infuse its own in-network processing functions into the network. For example, a new aggregation function could be defined or a specific mobile agent has to be written by the application programmer anyway.

  In-network processing and application-specific code may also be useful to detect **complex events**: events that cannot be detected locally, by a single sensor, but for which data has to be exchanged between sensors.
- Related to the specification of aggregation functions is the specification of the required accuracy of a result. This can take on the form of specifying bounds on the number of group members that should contribute to a result, or the level of compression that should be applied. Hand in hand with required accuracy goes the acceptable energy expenditure to produce a given piece of information.
- Timeliness requirements about the delivery of data is a similar aspect. For example, it may be possible to provide a result quickly but at higher energy costs (e.g. by forcing nodes to wake up earlier than they would wake up anyway) or slowly but at reduced energy costs (e.g. by piggy-backing information on other data packets that have to exchanged anyway).

In general, any trade-offs regarding the energy consumption of any possible exchange of data packets should be made explicit as far as possible.

- The need to access location, timing, or network status information (e.g. energy reserves available in the nodes or the current rate of energy scavenging) via the service interface.
  It may also be useful to agglomerate location information into higher-level abstractions to be able to talk about objects that correspond to a human view of things, for example, "room 123". Similarly, facts like the administrative entity a sensor network belongs to can be practically important [751].
- To support the seamless connection of various nodes or entire networks as well as the simple access to services in an "unknown" network, there is a need for an explicit description of the set of available capabilities of the node/the network – for example, which physical parameters can be observed or which entities can be controlled. Sgroi et al. [751] argue for a "concept repository" for this purpose.
- Security requirements as well as properties have be somehow expressed.
- While not a direct part of an actual service interface, additional management functionality, for example, for updating components, can be convenient to be present in the interface as well.

To avoid confusion, it is worthwhile to point out that the design of synchronous or asynchronous interface semantics has very little to do with a blocking or nonblocking design of the actual service invocation. It is, for example, easy to implement an asynchronous semantics with blocking invocations as long as the operating system provides threads. These really are separate issues.

3.4.3 Discussion

Evidently, the wealth of options that a general-purpose interface to WSNs would have to offer is vast. Looking at the overall picture, three key issues – data centricity, trade-offs against energy, and accuracy – make these networks quite different from all existing network types and how to offer them in a convenient service interface to an application programmer is anything but clear. It is hence perhaps not so surprising that there has only been relatively little work on a systematic approach to service interfaces for WSN.

One attempt has been undertaken by Sgroi et al. [751], who start from a relatively conservative client/server interface paradigm and use it to arrange a "query manager" and a "command interface", embellished by additional sets of parameters. While their parameter sets are relatively extensive and can incorporate most of the issues above, it is not clear that this API can indeed support all types of programming models in WSNs. It is, in particular, unclear how to extend in-network processing functionalities (e.g. write new aggregation functions) based on their API, how to control energy trade-offs, or how to select from an application one out of several components that are suitable for a given tasks (e.g. select one of several routing protocols).

Some of the candidates for data centricity, in particular, publish/subscribe and databases, are relatively close to meeting all these requirements for a service interface, but all of them still need extensions. The publish/subscribe interface, for example, can be extended by subscriptions that express accuracy and tasking aspects. Akin to publish/subscribe is the notion of events where an application can express interests in single events or in certain complex events. One example is the DSWare system [490]. Also, the database approach appears promising.

On the basis of the idea of mobile agents, the SensorWare system [96] provides a set of simple commands, in particular, `query`, `send`, `wait`, `value`, and `replicate`. These commands allow mobile code to send itself to some other node, to replicate it into the network by sending itself to the "children" of a node, or to wait for the results returned from these children. This provides considerable flexibility, but is still a fairly low level of programming.

One example for a highly application-specific way of defining service requests is EnviroTrack [2], which is specialized to the tracking of mobile objects. It allows to define "contexts" for certain tracking tasks, which have activation functions and "reporting" objects, resulting in an extremely compact expression of the service request, which then has to be transformed into concrete interactions of sensor nodes.

As one example for another research approach, consider the attempt to model the behavior and characteristics of sensor networks as a set of Unified Modeling Language (UML) schemes, resulting in the "Sensor Modeling Language" (SensorML) [749]. As this effort is driven by specific application requirements (geosciences and earth-observing satellites), it concentrates mostly on the description of the capabilities of individual sensors, but makes provisions to express, for example, accuracy and data processing. The big advantage here is the potential to describe the "meaning" of measured parameters explicitly. Another example for such an approach is to use the DARPA Agent Markup Language (DAML) as an explicit description language for the capabilities of heterogeneous sensors [384]. How such UML-based concepts could be applied to entire networks is, however, completely open.

Looking at the high complexity of service interfaces necessary to harness all the possible options and requirements of how an application might want to interact with a protocol stack, it is rather questionable whether the existing, quite heavy-weight, but still limited, proposals for service interfaces are the last word on the topic. A better understanding in structuring this interaction is still necessary. Moreover, the price to pay in performance optimization when using a predefined service interface still has to be weighted against the danger of inexperienced application programmers messing with the protocol stack's internals.

3.5 Gateway concepts

3.5.1 The need for gateways

For practical deployment, a sensor network only concerned with itself is insufficient. The network rather has to be able to interact with other information devices, for example, a user equipped with a PDA moving in the coverage area of the network or with a remote user, trying to interact with the sensor network via the Internet (the standard example is to read the temperature sensors in one's home while traveling and accessing the Internet via a wireless connection). Figure 3.9 shows this networking scenario.

To this end, the WSN first of all has to be able to exchange data with such a mobile device or with some sort of gateway, which provides the physical connection to the Internet. This is relatively straightforward on the physical, MAC, and link layer – either the mobile device/the gateway is

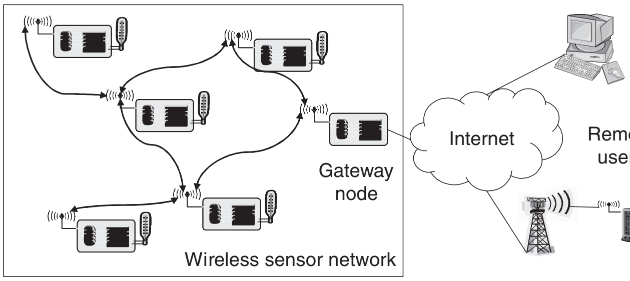

Figure 3.9 A wireless sensor network with gateway node, enabling access to remote clients via the Internet

equipped with a radio transceiver as used in the WSN, or some (probably not all) nodes in the WSN support standard wireless communication technologies such as IEEE 802.11. Either option can be advantageous, depending on the application and the typical use case. Possible trade-offs include the percentage of multitechnology sensor nodes that would be required to serve mobile users in comparison with the overhead and inconvenience to fit WSN transceivers to mobile devices like PDAs.

The design of gateways becomes much more challenging when considering their logical design. One option to ponder is to regard a gateway as a simple router between Internet and sensor network. This would entail the use of Internet protocols within the sensor network. While this option has been considered as well [215] and should not be disregarded lightly, it is the prevalent consensus that WSNs will require specific, heavily optimized protocols. Thus, a simple router will not suffice as a gateway.

The remaining possibility is therefore to design the gateway as an actual application-level gateway: on the basis of the application-level information, the gateway will have to decide its action. A rough distinction of the open problems can be made according to from where the communication is initiated.

3.5.2 WSN to Internet communication

Assume that the initiator of a WSN–Internet communication resides in the WSN (Figure 3.10) – for example, a sensor node wants to deliver an alarm message to some Internet host. The first problem to solve is akin to ad hoc networks, namely, how to find the gateway from within the network. Basically, a routing problem to a node that offers a specific service has to be solved, integrating routing and service discovery [139, 420, 435, 696, 799].

If several such gateways are available, how to choose between them? In particular, if not all Internet hosts are reachable via each gateway or at least if some gateway should be preferred for a given destination host? How to handle several gateways, each capable of IP networking, and the communication among them? One option is to build an IP overlay network on top of the sensor network [946].

How does a sensor node know to which Internet host to address such a message? Or even worse, how to map a semantic notion (“Alert Alice”) to a concrete IP address? Even if the sensor node does not need to be able to process the IP protocol, it has to include sufficient information (IP address and port number, for example) in its own packets; the gateway then has to extract this information and translate it into IP packets. An ensuing question is which source address to use here – the gateway in a sense has to perform tasks similar to that of a Network Address Translation (NAT) device [225].

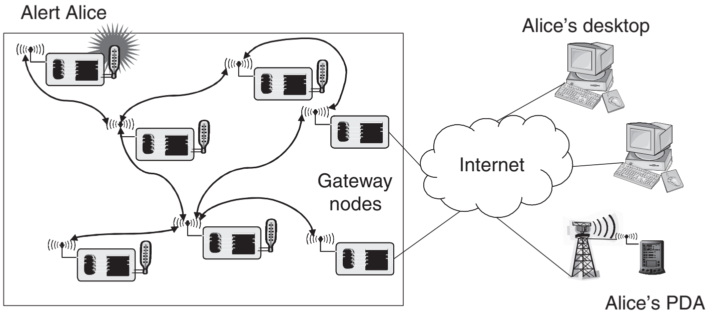

Figure 3.10 An event notification to “Alice” needs decisions about, among others, gateway choice, mapping “Alice” to a concrete IP address, and translating an intra-WSN event notification message to an Internet application message

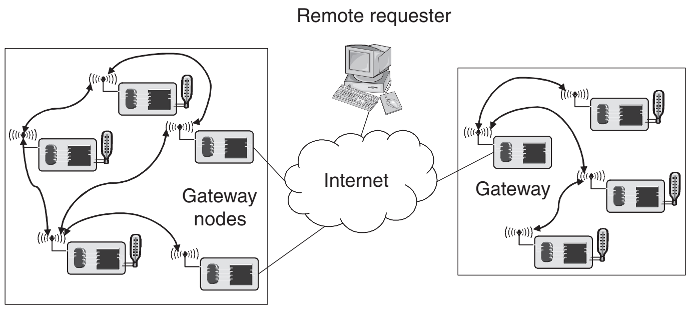

Figure 3.11 Requesting sensor network information from a remote terminal entails choices about which network to address, which gateway node of a given network, and how and where to adapt application-layer protocol in the Internet to WSN-specific protocols

3.5.3 Internet to WSN communication

The case of an Internet-based entity trying to access services of a WSN is even more challenging (Figure 3.11). This is fairly simple if this requesting terminal is able to directly communicate with the WSN, for example, a mobile requester equipped with a WSN transceiver, and also has all the necessary protocol components at its disposal. In this case, the requesting terminal can be a direct part of the WSN and no particular treatment is necessary.

The more general case is, however, a terminal "far away" requesting the service, not immediately able to communicate with any sensor node and thus requiring the assistance of a gateway node. First of all, again the question of service discovery presents itself – how to find out that there actually is a sensor network in the desired location, and how to find out about the existence of a gateway node?

Once the requesting terminal has obtained this information, how to access the actual services? Clearly, addressing an individual sensor (like addressing a communication peer in a traditional Internet application) both goes against the grain of the sensor network philosophy where an individual sensor node is irrelevant compared to the data that it provides and is impossible if a sensor node does not even have an IP address.

The requesting terminal can instead send a properly formatted request to this gateway, which acts as an application-level gateway or a proxy for the individual/set of sensor nodes that can answer this request; the gateway translates this request into the proper intrasensor network protocol interactions. This assumes that there is an application-level protocol that a remote requester and gateway can use and that is more suitable for communication over the Internet than the actual sensor network protocols and that is more convenient for the remote terminal to use. The gateway can then mask, for example, a data-centric data exchange within the network behind an identity-centric exchange used in the Internet.

It is by no means clear that such an application-level protocol exists that represents an actual simplification over just extending the actual sensor network protocols to the remote terminal, but there are some indications in this direction. For example, it is not necessary for the remote terminal to be concerned with maintaining multihop routes in the network nor should it be considered as "just another hop" as the characteristics of the Internet connection are quite different from a wireless hop.

In addition, there are some clear parallels for such an application-level protocol with so-called Web Service Protocols, which can explicitly describe services and the way they can be accessed. The Web Service Description Language (WSDL) [166], in particular, can be a promising starting point for extension with the required attributes for WSN service access – for example, required accuracy, energy trade-offs, or data-centric service descriptions. Moreover, the question arises as to how to

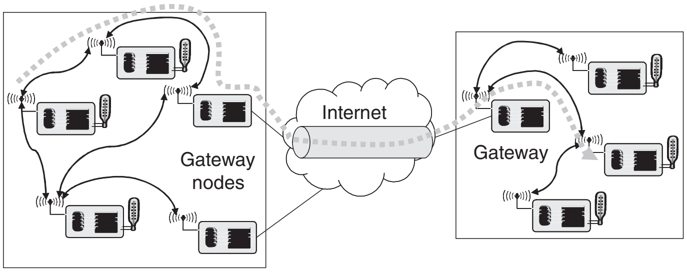

Figure 3.12 Connecting two WSNs with a tunnel over the Internet

integrate WSN with general middleware architectures [699] or how to make WSN services accessible from, say, a standard Web browser (which should be an almost automatic by-product of using WSDL and related standards in the gateway). However, research here is still in its early infancy [384, 508, 656]. Also, once a general-purpose service interface to WSNs is commonly accepted (such as [751]), this will have a clear impact on how to access WSN services from afar as well.

3.5.4 WSN tunneling

In addition to these scenarios describing actual interactions between a WSN and Internet terminals, the gateways can also act as simple extensions of one WSN to another WSN. The idea is to build a larger, "virtual" WSN out of separate parts, transparently "tunneling" all protocol messages between these two networks and simply using the Internet as a transport network (Figure 3.12) [751]. This can be attractive, but care has to be taken not to confuse the virtual link between two gateway nodes with a real link; otherwise, protocols that rely on physical properties of a communication link can get quite confused (e.g. time synchronization or localization protocols).

Such tunnels need not necessarily be in the form of fixed network connections; even mobile nodes carried by people can be considered as means for intermediate interconnection of WSNs [292]. Fall [252] also studies a similar problem in a more general setting.

3.6 Conclusion

The main conclusion to draw from this chapter is the fact that wireless sensor networks and their networking architecture will have many different guises and shapes. For many applications, but by no means all, multihop communication is the crucial enabling technology, and most of the WSN research as well as the following part of this book are focused on this particular form of wireless networking.

Four main optimization goals – WSN-specific forms of quality of service support, energy efficiency, scalability, and robustness – dominate the requirements for WSNs and have to be carefully arbitrated and balanced against each other. To do so, the design of WSNs departs in crucial aspects from that of traditional networks, resulting in a number of design principles. Most importantly, distributed organization of the network, the use of in-network processing, a data-centric view of the network, and the adaptation of result fidelity and accuracy to given circumstances are pivotal techniques to be considered for usage.

The large diversity of WSNs makes the design of a uniform, general-purpose service interface difficult; consequently, no final solutions to this problem are currently available. Similarly, the integration of WSNs in larger network contexts, for example, to allow Internet-based hosts a simple access to WSN services, is also still a fairly open problem.

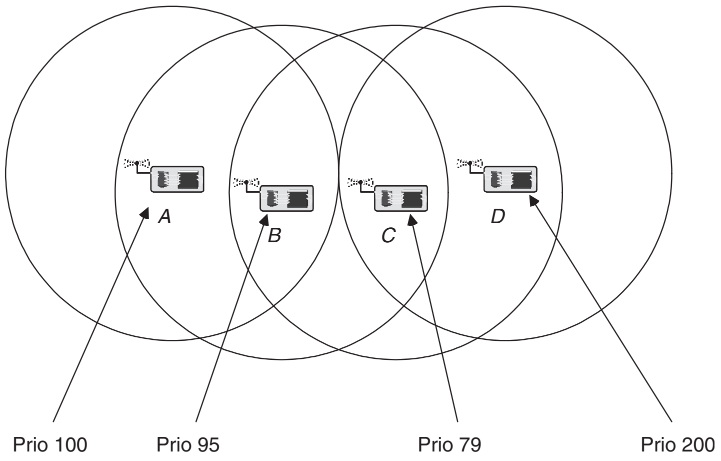

Figure 5.13 TRAMA: conflict situation

queuing delay of a packet waiting for transmission. Two scenarios are simulated: a single-hop scenario and a multihop scenario with one sink node and the sensors transmitting periodically to the sink. The underlying physical layer resembles the RF Monolithics TR1000 transceiver [690] (see also Section 2.1.4). As opposed to S-MAC, the energy savings of TRAMA depend on the load situation, while in S-MAC it depends on the duty cycle. The investigations confirmed also a well-known property of TDMA protocols stating that these have higher delays but also higher maximum throughput than contention-based protocols.

The TRAMA protocol needs significant computation and memory in dense sensor networks since the two-hop neighborhood of a node tends to be large in this case. Therefore, TRAMA is a feasible solution only if the sensor nodes have sufficient resources.

5.4.4 Further solutions

So far, we have mainly concentrated on protocols that assume almost no infrastructure, except some sink nodes. If we can assume the presence of infrastructure nodes, that is, nodes that are interconnected and not energy constrained, we can shift burdens to these nodes. For example, Shih et al. [762] assume an infrastructure setting (base stations close to the sensors) with tight latency requirements. As for the MAC, they investigate a combination of FDMA and TDMA (similar to GSM [848]) and derive a formula for the optimum number of FDMA channels for best energy efficiency under delay constraints.

All the schemes discussed so far need a periodically recurring neighbor discovery/network setup phase to adapt their schedules to changing network topologies. It would be very interesting to have topology-invariant schemes, which do not need these phases; one example scheme is proposed in reference [160] and another can be found in [161]. However, these schemes still need the knowledge of global network parameters (number of nodes, maximum node degree) and contain no provisions to let nodes sleep. Another topology-independent scheme for schedule formation is described in reference [386].

5.5 The IEEE 802.15.4 MAC protocol

The Institute of Electrical and Electronics Engineers (IEEE) finalized the IEEE 802.15.4 standard in October 2003 ([468]; see also [929], [317], and [114]). The standard covers the physical layer and the MAC layer of a low-rate Wireless Personal Area Network (WPAN). Sometimes, people confuse IEEE 802.15.4 with ZigBee[5], an emerging standard from the ZigBee alliance. ZigBee uses the services offered by IEEE 802.15.4 and adds network construction (star networks, peer-to-peer/mesh networks, cluster-tree networks), security, application services, and more.

The targeted applications for IEEE 802.15.4 are in the area of wireless sensor networks, home automation, home networking, connecting devices to a PC, home security, and so on. Most of these applications require only low-to-medium bitrates (up to some few hundreds of kbps), moderate average delays without too stringent delay guarantees, and for certain nodes it is highly desirable to reduce the energy consumption to a minimum. The physical layer offers bitrates of 20 kbps (a single channel in the frequency range 868–868.6 MHz), 40 kbps (ten channels in the range between 905 and 928 MHz) and 250 kbps (16 channels in the 2.4 GHz ISM band between 2.4 and 2.485 GHz with 5-MHz spacing between the center frequencies). There are a total of 27 channels available, but the MAC protocol uses only one of these channels at a time; it is not a multichannel protocol. More details about the physical layer can be found in Section 2.1.4.

The MAC protocol combines both schedule-based as well as contention-based schemes. The protocol is asymmetric in that different types of nodes with different roles are used, which is described next.

5.5.1 Network architecture and types/roles of nodes

The standard distinguishes on the MAC layer two types of nodes:

- A Full Function Device (FFD) can operate in three different roles: it can be a **PAN coordinator** (PAN = Personal Area Network), a simple **coordinator** or a **device**.
- A Reduced Function Device (RFD) can operate only as a device.

A device must be associated to a coordinator node (which must be a FFD) and communicates only with this, this way forming a **star network**. Coordinators can operate in a peer-to-peer fashion and multiple coordinators can form a Personal Area Network (PAN). The PAN is identified by a 16-bit **PAN Identifier** and one of its coordinators is designated as a PAN coordinator.

A coordinator handles among others the following tasks:

- It manages a list of associated devices. Devices are required to explicitly associate and disassociate with a coordinator using certain signaling packets.
- It allocates short addresses to its devices. All IEEE 802.15.4 nodes have a 64-bit device address. When a device associates with a coordinator, it may request assignment of a 16-bit short address to be used subsequently in all communications between device and coordinator. The assigned address is indicated in the association response packet issued by the coordinator.
- In the beaconed mode of IEEE 802.15.4, it transmits regularly **frame beacon** packets announcing the PAN identifier, a list of outstanding frames, and other parameters. Furthermore, the coordinator can accept and process requests to reserve fixed time slots to nodes and the allocations are indicated in the beacon.
- It exchanges data packets with devices and with peer coordinators.

In the remainder of this section, we focus on the data exchange between coordinator and devices in a star network; a possible protocol for data exchange between coordinators is described in Section 5.2.3. We start with the beaconed mode of IEEE 802.15.4.

[5] see `http://www.zigbee.org/`; a brief slide set on ZigBee entitled "ZigBee Overview" can be found under `http://www.zigbee.org/en/resources`.

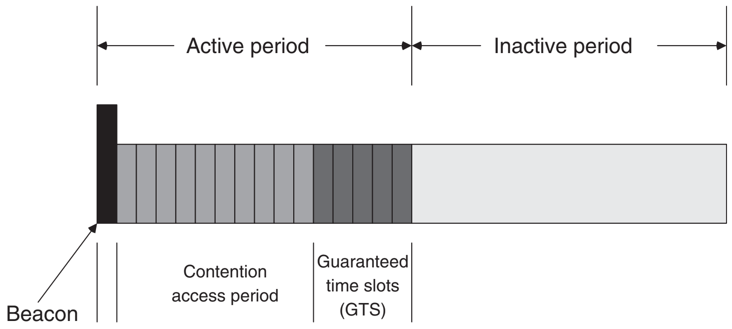

Figure 5.14 Superframe structure of IEEE 802.15.4

5.5.2 Superframe structure

The coordinator of a star network operating in the **beaconed mode** organizes channel access and data transmission with the help of a superframe structure displayed in Figure 5.14.

All superframes have the same length. The coordinator starts each superframe by sending a frame beacon packet. The frame beacon includes a **superframe specification** describing the length of the various components of the following superframe:

- The superframe is subdivided into an **active period** and an **inactive period**. During the inactive period, all nodes including the coordinator can switch off their transceivers and go into sleep state. The nodes have to wake up immediately before the inactive period ends to receive the next beacon. The inactive period may be void.
- The active period is subdivided into 16 time slots. The first time slot is occupied by the beacon frame and the remaining time slots are partitioned into a **Contention Access Period (CAP)** followed by a number (maximal seven) of contiguous **Guaranteed Time Slots (GTSs)**.

The length of the active and inactive period as well as the length of a single time slot and the usage of GTS slots are configurable.

The coordinator is active during the entire active period. The associated devices are active in the GTS phase only in time slots allocated to them; in all other GTS slots they can enter sleep mode. In the CAP, a device can shut down its transceiver if it has neither any own data to transmit nor any data to fetch from the coordinator.

It can be noted already from this description that coordinators do much more work than devices and the protocol is inherently asymmetric. The protocol is optimized for cases where energy-constrained sensors are to be attached to energy-unconstrained nodes.

5.5.3 GTS management

The coordinator allocates GTS to devices only when the latter send appropriate request packets during the CAP. One flag in the request indicates whether the requested time slot is a **transmit slot** or a **receive slot**. In a transmit slot, the device transmits packets to the coordinator and in a receive slot the data flows in the reverse direction. Another field in the request specifies the desired number of contiguous time slots in the GTS phase.

The coordinator answers the request packet in two steps: An immediate acknowledgment packet confirms that the coordinator has received the request packet properly but contains no information about success or failure of the request.

After receiving the acknowledgment packet, the device is required to track the coordinator's beacons for some specified time (called *aGTSDescPersistenceTime*). When the coordinator has sufficient resources to allocate a GTS to the node, it inserts an appropriate **GTS descriptor** into one of the next beacon frames. This GTS descriptor specifies the short address of the requesting node and the number and position of the time slots within the GTS phase of the superframe. A device can use its allocated slots each time they are announced by the coordinator in the GTS descriptor. If the coordinator has insufficient resources, it generates a GTS descriptor for (invalid) time slot zero, indicating the available resources in the descriptors length field. Upon receiving such a descriptor, the device may consider renegotiation. If the device receives no GTS descriptor within *aGTSDescPersistenceTime* time after sending the request, it concludes that the allocation request has failed.

A GTS is allocated to a device on a regular basis until it is explicitly deallocated. The deallocation can be requested by the device by means of a special control frame. After sending this frame, the device shall not use the allocated slots any further. The coordinator can also trigger deallocation based on certain criteria. Specifically, the coordinator monitors the usage of the time slot: If the slot is not used at least once within a certain number of superframes, the slot is deallocated. The coordinator signals deallocation to the device by generating a GTS descriptor with start slot zero.

5.5.4 Data transfer procedures

Let us first assume that a device wants to transmit a data packet to the coordinator. If the device has an allocated transmit GTS, it wakes up just before the time slot starts and sends its packet immediately without running any carrier-sense or other collision-avoiding operations. However, the device can do so only when the full transaction consisting of the data packet and an immediate acknowledgment sent by the coordinator as well as appropriate InterFrame Spaces (IFSs) fit into the allocated time slots. If this is not the case or when the device does not have any allocated slots, it sends its data packet during the CAP using a slotted CSMA protocol, described below. The coordinator sends an immediate acknowledgment for the data packet.

The other case is a data transfer from the coordinator to a device. If the device has allocated a receive GTS and when the packet/acknowledgment/IFS cycle fits into these, the coordinator simply transmits the packet in the allocated time slot without further coordination. The device has to acknowledge the data packet.

The more interesting case is when the coordinator is not able to use a receive GTS. The handshake between device and coordinator is sketched in Figure 5.15. The coordinator announces a buffered packet to a device by including the devices address into the **pending address field** of the beacon frame. In fact, the device's address is included as long as the device has not retrieved the packet or a certain timer has expired. When the device finds its address in the pending address field, it sends a special **data request** packet during the CAP. The coordinator answers this packet with an acknowledgment packet and continues with sending the data packet. The device knows upon receiving the acknowledgment packet that it shall leave its transceiver on and prepares for the incoming data packet, which in turn is acknowledged. Otherwise, the device tries again to send the data request packet during one of the following superframes and optionally switches off its transceiver until the next beacon.

5.5.5 Slotted CSMA-CA protocol

When nodes have to send data or management/control packets during the CAP, they use a slotted CSMA protocol. The protocol contains no provisions against hidden-terminal situations, for example

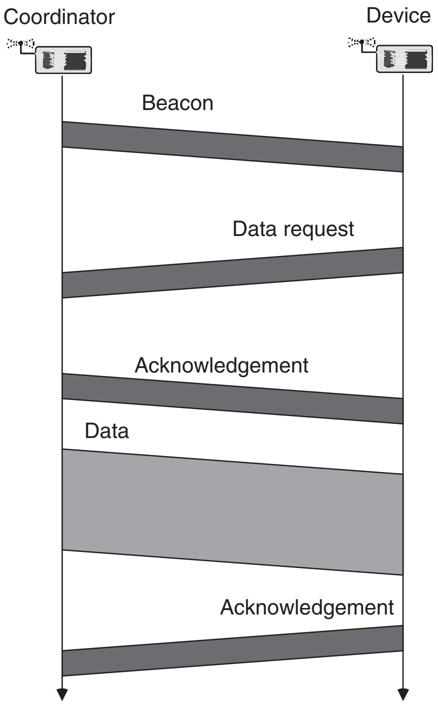

Figure 5.15 Handshake between coordinator and device when the device retrieves a packet [468, Fig. 8]

there is no RTS/CTS handshake. To reduce the probability of collisions, the protocol uses random delays; it is thus a CSMA-CA protocol (CSMA with Collision Avoidance). Using such random delays is also part of the protocols described in Section 5.3.1. We describe the protocol operation in some more detail; please refer to Figure 5.16 also.

The time slots making up the CAP are subdivided into smaller time slots, called **backoff periods**. One backoff period has a length corresponding to 20 channel symbol times and the slots considered by the slotted CSMA-CA protocol are just these backoff periods.

The device maintains three variables *NB*, *CW*, and *BE*. The variable *NB* counts the number of backoffs, *CW* indicates the size of the current congestion window, and *BE* is the current backoff exponent. Upon arrival of a new packet to transmit, these variables are initialized with $NB = 0$, $CW = 2$, and $BE = macMinBE$ (with *macMinBE* being a protocol parameter), respectively. The device awaits the next backoff period boundary and draws an integer random number r from the interval $[0, 2^{BE} - 1]$. The device waits for r backoff periods and performs a carrier-sense operation (denoted as Clear Channel Assessment (CCA) in the standard). If the medium is idle, the device decrements *CW*, waits for the next backoff period boundary, and senses the channel again. If the channel is still idle, the device assumes that it has won contention and starts transmission of its data packet. If either of the CCA operations shows a busy medium, the number of backoffs *NB* and the backoff exponent *BE* are incremented and *CW* is set back to $CW = 2$. If *NB* exceeds a threshold, the device drops the frame and declares a failure. Otherwise, the device again draws an integer r from $[0, 2^{BE} - 1]$ and waits for the indicated number of backoff slots. All subsequent steps are repeated.

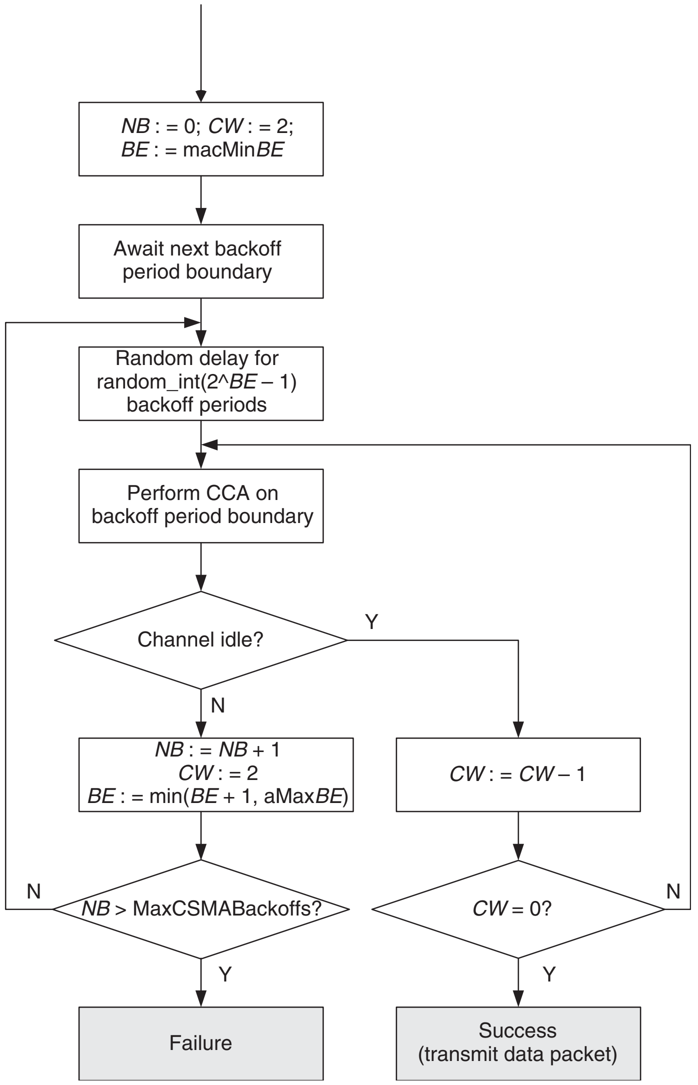
NB : = 0; CW : = 2;
BE : = macMinBE
Await next backoff period boundary
Random delay for random_int(2^BE – 1) backoff periods
Perform CCA on backoff period boundary
Channel idle?
Y
N
NB : = NB + 1
CW : = 2
BE : = min(BE + 1, aMaxBE)
CW : = CW – 1
N
NB > MaxCSMABackoffs?
CW = 0?
N
Y
Y
Failure
Success (transmit data packet)

Figure 5.16 Schematic of the slotted CSMA-CA algorithm (simplified version of [468, Fig. 61])

5.5.6 Nonbeaconed mode

The IEEE 802.15.4 protocol offers a **nonbeaconed mode** besides the beaconed mode. Some important differences between these modes are the following:

- In the nonbeaconed mode, the coordinator does *not* send beacon frames nor is there any GTS mechanism. The lack of beacon packets takes away a good opportunity for devices to acquire time synchronization with the coordinator.
- All packets from devices are transmitted using an unslotted (because of the lack of time synchronization) CSMA-CA protocol. As opposed to the slotted CSMA-CA protocol, there is no

synchronization to backoff period boundaries and, in addition, the device performs only a single CCA operation. If this indicates an idle channel, the device infers success.
- Coordinators must be switched on constantly but devices can follow their own sleep schedule. Devices wake up for two reasons: (i) to send a data/control packet to the coordinators, or (ii) to fetch a packet destined to itself from the coordinator by using the data request/acknowledgment/data/acknowledgment handshake (fetch cycle) discussed above. The data request packet is sent through the unslotted CSMA-CA mechanism and the following acknowledgment is sent without any further ado. When the coordinator has a data packet for the device, it transmits it using the unslotted CSMA-CA access method and the device sends an immediate acknowledgment for the data. Therefore, the device must stay awake for a certain time after sending the data request packet. The rate by which the device initiates the fetch cycle is application dependent.

5.5.7 Further reading

Performance evaluations of certain aspects of IEEE 802.15.4 are presented in [521] and [880]. Lu et al. [521] focus on the case of beacon-enabled networks and investigate some of the throughput/energy/delay trade-offs. Some attention is paid to the aspect of synchronizing devices to the beacons, for example, to initiate fetch cycles. In the tracking mode, the device scans the channel for the first beacon, learns the timing from this, and tries to wake up immediately before the next beacon, and so on. Accordingly, nodes have a precise idea when the next beacon will arrive and when they can learn about packets destined to them. In the nontracking mode, a node simply wakes up at some time, seeks the next beacon, and goes back into sleep mode for longer time when no packet is stored at the coordinator. Willig [880] investigates theoretical throughput bounds for the two CSMA-CA variants. The unslotted CSMA-CA variant has a higher throughput because of its lower overhead (no waiting for backoff period boundaries, only one carrier-sense operation). One key parameter for the achievable throughput is the transceiver turnover times since these directly impact the collision rates. In the unslotted version, the number of collisions can be decreased when the turnover times are decreased. This is, however, not true for the slotted version because of the synchronization of devices to the coordinator: when two backlogged devices choose the same number r of backoff periods as their random delay, they will sense the channel at the same time, see the same idle channel, and start to transmit at the same time.

5.6 How about IEEE 802.11 and bluetooth?

An obvious question is the following: Given that there are already a number of proven wireless MAC protocols and wireless products out there, why not simply use them? Specifically: There are two popular and commercially available systems, Bluetooth and IEEE 802.11. What is “wrong” about them?

The Bluetooth system is designed as a Wireless Personal Area Network (WPAN) with one major application, the connection of devices to a personal computer [318]. It already has been used as a means for prototyping wireless sensor network applications [72]. The PHY is based on a FHSS scheme having a hopping frequency of 1.6 kHz and an appropriate allocation of hopping sequences. The nodes are organized into **piconets** with one master and up to seven active slave nodes. The master chooses the hopping sequence, which the slaves have to follow. Furthermore, there can be several passive slave nodes in a piconet. The master polls the active slaves continuously. Two major drawbacks of Bluetooth are the need to constantly have a master node, spending much energy on polling his slaves, and the rather limited number of active slaves per piconet.[6] This is not compatible with the case of dense wireless sensor networks where a huge number of master nodes would be

[6] A newer version of Bluetooth remedies this limitation.

needed. An active slave must always be switched on since it cannot predict when it will be polled by the master. A passive slave has to apply at the master to become an active slave. This fails if there are seven active nodes already. Furthermore, it is required that each node is able to take the role of masters or slaves and thus bears considerable complexity. Also, the fast frequency hopping operations require tight synchronization between the nodes in a piconet.

In the IEEE 802.11 family of protocols, several physical layers are specified sharing a single MAC protocol [284, 815], the DCF, and on top of it the Point Coordination Function (PCF). Without further provisions, IEEE 802.11 requires any node x to constantly be in listen mode since another node y may attempt to transmit a frame to x at any time. Secondly, nodes are required to overhear RTS and CTS packets to adjust their NAV timers properly. IEEE 802.11 has some power-saving functionalities [691], but, in general, the system is targeted towards high bitrates and the available transceivers require orders of magnitude more energy than acceptable in low-bitrate sensor network applications. Furthermore, IEEE 802.11 is a single-hop protocol for both the infrastructure and ad hoc network scenarios and, in general, is targeted at letting a number of independent and competing users share a common channel in a fair manner. These goals do not match the goals of wireless sensor networks.

5.7 Further reading

MAC protocols is a popular issue in the sensor networks literature and many protocols are missing. The following list of references provides some further starting points.

- There are some proposals to add power-control elements to MAC protocols. A recurring theme here is the augmentation of protocols using RTS/CTS handshakes with methods for adapting the transmit power to the minimum level necessary to reach the intended neighbor with a given BER target or packet-loss probability; see, for example, [895] or the BASIC scheme discussed in [389]. In BASIC, the RTS and CTS packets are transmitted at highest power, the data and acknowledgment packets at just the right power, inferred from the signal strengths of either RTS or CTS. The same authors Jung and Vaidya [389] propose a variation of BASIC in which the transmitter increases transmit power periodically *within* the data packet to keep contenders away. In the PCMA protocol described by Monks et al. [572], the transmitter issues a RTS packet and the receiver measures the signal strength of this packet. Using this information, the receiver can derive bounds on acceptable interference from hidden nodes and this bound is transmitted along with the CTS answer. Any neighbor hearing the CTS is allowed to transmit as long as the interference it creates to the receiver is below the given bound. Further references are [446], [573], [48], and [577].
- Chlamtac et al. [163] propose different access protocols for RF ID applications. One of these protocols is based on TDMA, a second one is a random access protocol, and a third one is a directory protocol.
- The DE-MAC protocol presented by Kalidindi et al. [393] is a TDMA-based scheme, which aims to regulate slot assignment such that energy-poor nodes can go into sleep mode more often.
- Tay et al. [813] present a nonpersistent CSMA variant with carefully chosen backoff times specifically targeted for achieving small delays for the first few packets after a sensor network starts to report data about a triggering event.
- The capacity of MAC protocols based on the RTS/CTS virtual carrier-sensing approach in ad hoc networks is discussed in [43].
- One reference focusing on quality-of-service aspects in wireless multihop networks instead of energy is [398].

Forward error correction (FEC) The transmitter accepts a stream or a block of user data bits or source bits, adds suitable redundancy, and transmits the result to the receiver. Depending on the amount and structure of the redundancy, the receiver might be able to correct some bit/symbol errors. It is known that AWGN channels have a higher capacity than Rayleigh fading channels and many coding schemes achieve better BER performance on AWGN than on fading channels with their bursty errors [79]. The operation of **interleaving** applies a permutation operation to a block of bits, hoping to distribute bursty errors smoothly and letting the channel "look" like an AWGN channel. FEC is discussed in some more detail in Section 6.2.3.

ARQ The basic idea of ARQ protocols [322, 511] can be described as follows: The transmitter prepends a header and appends a checksum to a data block. The resulting packet is then transmitted. The receiver checks the packet's integrity with the help of the checksum and provides some feedback to the transmitter regarding the success of packet transmission. On receiving negative feedback, the transmitter performs a retransmission. ARQ protocols are discussed in Section 6.2.2.

4.3 Physical layer and transceiver design considerations in WSNs

So far, we have discussed the basics of the PHY without specific reference to wireless sensor networks. Some of the most crucial points influencing PHY design in wireless sensor networks are:

- Low power consumption.
- As one consequence: small transmit power and thus a small transmission range.
- As a further consequence: low duty cycle. Most hardware should be switched off or operated in a low-power standby mode most of the time.
- Comparably low data rates, on the order of tens to hundreds kilobits per second, required.
- Low implementation complexity and costs.
- Low degree of mobility.
- A small form factor for the overall node.

In this section, we discuss some of the implications of these requirements.

In general, in sensor networks, the challenge is to find modulation schemes and transceiver architectures that are simple, low-cost but still robust enough to provide the desired service.

4.3.1 Energy usage profile

The choice of a small transmit power leads to an energy consumption profile different from other wireless devices like cell phones. These pivotal differences have been discussed in various places already but deserve a brief summary here.

First, the radiated energy is small, typically on the order of 0 dBm (corresponding to 1 mW). On the other hand, the overall transceiver (RF front end and baseband part) consumes much more energy than is actually radiated; Wang et al. [855] estimate that a transceiver working at frequencies beyond 1 GHz takes 10 to 100 mW of power to radiate 1 mW. In reference [115, Chap. 3], similar numbers are given for 2.4-GHz CMOS transceivers: For a radiated power of 0 dBm, the transmitter uses actually 32 mW, whereas the receiver uses even more, 38 mW. For the Mica motes, 21 mW are consumed in transmit mode and 15 mW in receive mode [351]. These numbers coincide well with the observation that many practical transmitter designs have efficiencies below 10 % [46] at low radiated power.

A second key observation is that for small transmit powers the transmit and receive modes consume more or less the same power; it is even possible that reception requires more power than transmission [670, 762]; depending on the transceiver architecture, the idle mode's power consumption can be less or in the same range as the receive power [670]. To reduce average power consumption in a low-traffic wireless sensor network, keeping the transceiver in idle mode all the time would consume significant amounts of energy. Therefore, it is important to put the transceiver into sleep state instead of just idling. It is also important to explicitly include the received power into energy dissipation models, since the traditional assumption that receive energy is negligible is no longer true.

However, there is the problem of the **startup energy/startup time**, which a transceiver has to spend upon waking up from sleep mode, for example, to ramp up phase-locked loops or voltage-controlled oscillators. During this startup time, no transmission or reception of data is possible [762]. For example, the μAMPS-1 transceiver needs a startup time of 466 µs and a power dissipation of 58 mW [561, 563]. Therefore, going into sleep mode is unfavorable when the next wakeup comes fast. It depends on the traffic patterns and the behavior of the MAC protocol to schedule the transceiver operational state properly. If possible, not only a single but multiple packets should be sent during a wakeup period, to distribute the startup costs over more packets. Clearly, one can attack this problem also by devising transmitter architectures with faster startup times. One such architecture is presented in reference [855].

A third key observation is the relative costs of communications versus computation in a sensor node. Clearly, a comparison of these costs depends for the communication part on the BER requirements, range, transceiver type, and so forth, and for the computation part on the processor type, the instruction mix, and so on. However, in [670], a range of energy consumptions is given for Rockwell's WIN nodes, UCLA's WINS NG 2.0 nodes, and the MEDUSA II nodes. For the WIN nodes, 1500 to 2700 instructions can be executed per transmitted bit, for the MEDUSA II nodes this ratio ranges from 220:1 up to 2900:1, and for the WINS NG nodes, it is around 1400:1. The bottom line is that computation is cheaper than communication!

4.3.2 Choice of modulation scheme

A crucial point is the choice of modulation scheme. Several factors have to be balanced here: the required and desirable data rate and symbol rate, the implementation complexity, the relationship between radiated power and target BER, and the expected channel characteristics.

To maximize the time a transceiver can spend in sleep mode, the transmit times should be minimized. The higher the data rate offered by a transceiver/modulation, the smaller the time needed to transmit a given amount of data and, consequently, the smaller the energy consumption.

A second important observation is that the power consumption of a modulation scheme depends much more on the symbol rate than on the data rate [115, Chap. 3]. For example, power consumption measurements of an IEEE 802.11b Wireless Local Area Network (WLAN) card showed that the power consumption depends on the modulation scheme, with the faster Complementary Code Keying (CCK) modes consuming more energy than DBPSK and DQPSK; however, the relative differences are below 10 % and all these schemes have the same symbol rate. It has also been found that for the μAMPS-1 nodes the power consumption is insensitive to the data rate [762].

Obviously, the desire for "high" data rates at "low" symbol rates calls for m-ary modulation schemes. However, there are trade-offs:

- m-ary modulation requires more complex digital and analog circuitry than 2-ary modulation [762], for example, to parallelize user bits into m-ary symbols.

Table 4.3 Bandwidth efficiency η_{BW} and E_b/N_0[dB] required at the receiver to reach a BER of 10^{-6} over an AWGN channel for m-ary orthogonal FSK and PSK (adapted from reference [682, Chap. 6])

m	2	4	8	16	32	64
m-ary PSK:η_{BW}	0.5	1.0	1.5	2.0	2.5	3.0
m-ary PSK:E_b/N_0	10.5	10.5	14.0	18.5	23.4	28.5
m-ary FSK:η_{BW}	0.40	0.57	0.55	0.42	0.29	0.18
m-ary FSK:E_b/N_0	13.5	10.8	9.3	8.2	7.5	6.9

- Many m-ary modulation schemes require for increasing m an increased E_b/N_0 ratio and consequently an increased radiated power to achieve the same target BER; others become less and less bandwidth efficient. This is exemplarily shown for coherently detected m-ary FSK and PSK in Table 4.3, where for different values of m, the achieved bandwidth efficiencies and the E_b/N_0 required to achieve a target BER of 10^{-6} are displayed. However, in wireless sensor network applications with only low to moderate bandwidth requirements, a loss in bandwidth efficiency can be more tolerable than an increased radiated power to compensate E_b/N_0 losses.
- It is expected that in many wireless sensor network applications most packets will be short, on the order of tens to hundreds of bits. For such packets, the startup time easily dominates overall energy consumption, rendering any efforts in reducing the transmission time by choosing m-ary modulation schemes irrelevant.

Let us explore the involved trade-offs a bit further with the help of an example.

Example 4.1 (Energy efficiency of m-ary modulation schemes) Our goal is to transmit data over a distance of $d = 10$ m at a target BER of 10^{-6} over an AWGN channel having a path-loss exponent of $\gamma = 3.5$ (corresponding to the value determined in reference [563]). We compare two families of modulations: coherently detected m-ary PSK and coherently detected orthogonal m-ary orthogonal FSK. For these two families we display in Table 4.3, the bandwidth efficiencies η_{BW} and the E_b/N_0 in dB required at the receiver to reach a BER of 10^{-6} over an AWGN channel.

From the discussion in Section 4.2.3, the relationship between E_b/N_0 and the received power at a distance d is given as:

$$\begin{aligned}\frac{E_b}{N_0} &= \text{SNR} \cdot \frac{1}{R} = \frac{P_{\text{rcvd}}(d)}{N_0} \cdot \frac{1}{R} \\ &= \frac{1}{N_0 \cdot R} \cdot \frac{P_{\text{tx}} \cdot G_t \cdot G_r \cdot \lambda^2}{(4\pi)^2 \cdot d_0^{\gamma} \cdot L} \cdot \left(\frac{d_0}{d}\right)^{\gamma},\end{aligned} \tag{4.8}$$

which can be easily solved for P_{tx} given a required E_b/N_0 value and data rate R. We denote the solution as $P_{\text{tx}}\left(\frac{E_b}{N_0}, R\right)$. One example: From Table 4.3 we obtain that 16-PSK requires an E_b/N_0 of 18.5 dB to reach the target BER. When fixing the parameters $G_t = G_r = L = 1$, $\lambda = 12.5$ cm (according to a 2.4 GHz transceiver), reference distance $d_0 = 1$ m, distance $d = 10$ m, a data rate of $R = 1$ Mbps, and a noise level of $N_0 = -180$ dB this corresponds to $P_{\text{tx}}(18.5\,\text{dB}, R) \approx 2.26$ mW.

We next utilize a transceiver energy consumption model developed in references [762, 855] that incorporates startup energy and transmit energy. In this model, it is assumed that during the startup time mainly a frequency synthesizer is active, consuming energy P_{FS}, while during the actual waveform transmission power is consumed by the frequency synthesizer, the modulator (using P_{MOD}), and the radiated energy $P_{\mathrm{tx}}(\cdot,\cdot)$. The power amplifier is not explicitly considered. Using reference [855], we assume $P_{\mathrm{FS}} = 10\,\mathrm{mW}$, $P_{\mathrm{MOD}} = 2\,\mathrm{mW}$ and a symbol rate of $B = 1$ M symbols/sec. The duration of the startup time is T_{start}. For the case of binary modulation, we assume the following energy model:

$$E_{\mathrm{binary}}\left(\frac{E_b}{N_0}, B\right) = P_{\mathrm{FS}} \cdot T_{\mathrm{start}} + \left(P_{\mathrm{MOD}} + P_{\mathrm{FS}} + P_{\mathrm{tx}}\left(\frac{E_b}{N_0}, B\right)\right) \cdot \frac{n}{B},$$

where n is the number of data bits to transmit in a packet. For the case of m-ary modulation, it is assumed that the power consumption of the modulator and the frequency synthesizer are increased by some factors $\alpha \geq 1$, $\beta \geq 1$, such that the overall energy expenditure is:

$$E_{m\text{-ary}}\left(\frac{E_b}{N_0}, B \cdot \log_2 m\right) = \beta \cdot P_{\mathrm{FS}} \cdot T_{\mathrm{start}} + \left(\alpha \cdot P_{\mathrm{MOD}} + \beta \cdot P_{\mathrm{FS}} + P_{\mathrm{tx}}\left(\frac{E_b}{N_0}, B \cdot \log_2 m\right)\right) \cdot \frac{n}{B \cdot \log_2(m)}.$$

Accepting the value $\beta = 1.75$ from reference [855] for both PSK and FSK modulation, one can evaluate the ratio $\frac{E_{m\text{-ary}}(\cdot,\cdot)}{E_{\mathrm{binary}}(\cdot,\cdot)}$ to measure the energy advantage or disadvantage of m-ary modulation over binary modulation. As an example, we show this ratio in Figure 4.10 for varying $m \in \{4, 8, 16, 32, 64\}$, with $\alpha = 2.0$, a startup time of 466 µs, and two different packet sizes, 100 bits and 2000 bits. The two upper curves correspond to a packet size of 100 bits; the two lower curves correspond to the packet size of 2000 bits. Other results obtained with a shorter startup time of 100 µs or $\alpha = 3.0$ look very similar. One can see that for large packet sizes m-ary FSK modulation is favorable, since the actual packet transmission times are shortened and furthermore the required E_b/N_0 *decreases* for increasing m, at the expense of a reduced bandwidth efficiency, which translates into a wider required spectrum (the FSK scheme is orthogonal FSK). For m-ary PSK, only certain values of m give an energy advantage; for larger m the increased E_b/N_0 requirements outweigh the gains due to reduced transmit times. For small packet sizes, the binary modulation schemes are more energy efficient for both PSK and FSK, because the energy costs are dominated by the startup time. If one reduces β to $\beta = 1$ (assuming no extra energy consumption of the frequency synthesizer due to m-ary modulation), then m-ary modulation would, for all parameters under consideration, be truly better than binary modulation. The results presented in reference [855] indicate that the advantage of m-ary modulation increases as the startup time decreases. For shorter startup times also the packet lengths required to make m-ary modulation pay out are smaller.

Can we conclude from this that it is favorable to use large packets? Unfortunately, the answer is: it depends. As we will see in Chapter 6, longer packets at the same bit error rate and without employing error-correction mechanisms lead to higher packet error rates, which in turn lead to retransmitted packets, easily nullifying the energy gains of choosing m-ary modulation. A careful joint consideration of modulation and other schemes for increasing transmission robustness (FEC or ARQ schemes) is needed.

But it can be beneficial to transmit multiple short packets during a single wakeup period, thus achieving a lower relative influence of the startup costs per packet [562].

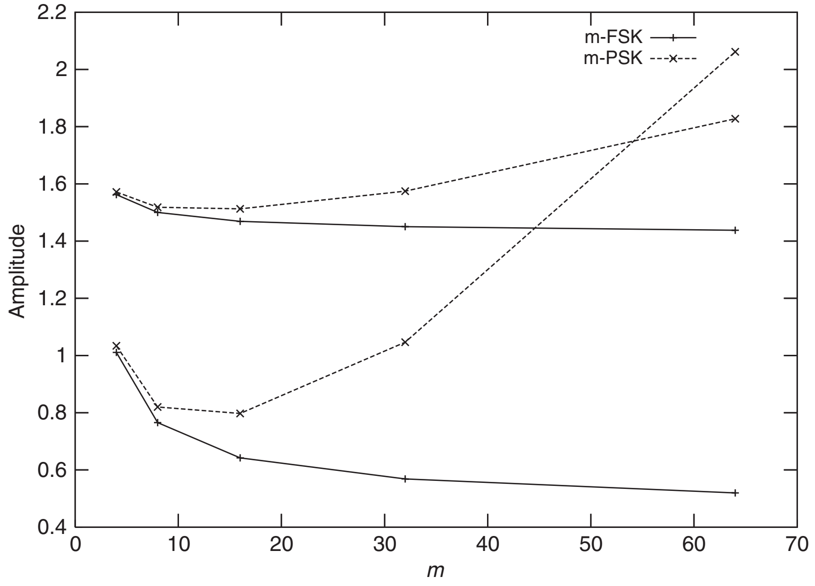

Figure 4.10 Comparison of the energy consumption of m-ary FSK/PSK to binary FSK/PSK for $\alpha = 2.0$ and startup time of 466 µs.

Clearly, this example provides only a single point in the whole design space. The bottom line here is that the choice of modulation scheme depends on several interacting aspects, including technological factors (in the example: α, β), packet size, target error rate, and channel error model (in reference [855], a similar example is carried out for the case of Rayleigh fading). The optimal decision would have to properly balance the modulation scheme and other measures to increase transmission robustness, since these also have energy costs:

- With retransmissions, entire packets have to be transmitted again.
- With FEC coding, more bits have to be sent and there is additional energy consumption for coding and decoding. While coding energy can be neglected, and the receiver needs significant energy for the decoding process [563]. This is especially cumbersome if the receiver is a power-constrained node. Coding and retransmission schemes are discussed in more detail in Chapter 6.
- The cost of increasing the radiated power [855] depends on the efficiency of the power amplifier (compare Section 2.2.4), but the radiated power is often small compared to the overall power dissipated by the transceiver, and additionally this drives the PA into a more efficient regime.[14]

In [670], a similar analysis as in our example has been carried out for m-ary QAM. Specifically, the energy-per-bit consumption (defined as the overall energy consumption for transmitting a packet of n bits divided by n) of different m-ary QAM modulation schemes has been investigated for different packet sizes, taking startup energy and the energy costs of power amplifiers as well as PHY and MAC packet overheads explicitly into account. For the particular setup used in this

[14] Of course, one disadvantage of using an increased transmit power is an increased interference for other transmissions and thus a decreased overall network capacity. However, this plays no role during low-load situations, which prevail in wireless sensor networks – unless event storms or other correlated traffic models are present.

investigation, 16-QAM seems to be the optimum modulation schemes for all different sizes of the user data.

4.3.3 Dynamic modulation scaling

Even if it is possible to determine the optimal scheme for a given combination of BER target, range, packet sizes and so forth, such an optimum is only valid for short time; as soon as one of the constraints changes, the optimum can change, too. In addition, other constraints like delay or the desire to achieve high throughput can dictate to choose higher modulation schemes.

Therefore, it is interesting to consider methods to *adapt* the modulation scheme to the current situation. Such an approach, called **dynamic modulation scaling**, is discussed in reference [738]. In particular, for the case of m-ary QAM and a target BER of 10^{-5}, a model has been developed that uses the symbol rate B and the number of levels per symbol m as parameters. This model expresses the energy required per bit and also the achieved delay per bit (the inverse of the data rate), taking into account that higher modulation levels need higher radiated energy. Extra startup costs are not considered. Clearly, the bit delay decreases for increasing B and m. The energy per bit depends much more on m than on B. In fact, for the particular parameters chosen, it is shown that both energy per bit and delay per bit are minimized for the maximum symbol rate. With modulation scaling, a packet is equipped with a delay constraint, from which directly a minimal required data rate can be derived. Since the symbol rate is kept fixed, the approach is to choose the smallest m that satisfies the required data rate and which thus minimizes the required energy per bit. Such delay constraints can be assigned either explicitly or implicitly. One approach explored in the paper is to make the delay constraint depend on the packet backlog (number of queued packets) in a sensor node: When there are no packets present, a small value for m can be used, having low energy consumption. As backlog increases, m is increased as well to reduce the backlog quickly and switch back to lower values of m. This modulation scaling approach has some similarities to the concept of **dynamic voltage scaling** discussed in Section 2.2.2.

4.3.4 Antenna considerations

The desired small form factor of the overall sensor nodes restricts the size and the number of antennas. As explained above, if the antenna is much smaller than the carrier's wavelength, it is hard to achieve good antenna efficiency, that is, with ill-sized antennas one must spend more transmit energy to obtain the same radiated energy.

Secondly, with small sensor node cases, it will be hard to place two antennas with suitable distance to achieve receive diversity. As discussed in Section 4.2.7, the antennas should be spaced apart at least 40–50 % of the wavelength used to achieve good effects from diversity. For 2.4 GHz, this corresponds to a spacing of between 5 and 6 cm between the antennas, which is hard to achieve with smaller cases.

In addition, radio waves emitted from an antenna close to the ground – typical in some applications – are faced with higher path-loss coefficients than the common value $\alpha = 2$ for free-space communication. Typical attenuation values in such environments, which are also normally characterized by obstacles (buildings, walls, and so forth), are about $\alpha = 4$ [245, 648].

Moreover, depending on the application, antennas must not protrude from the casing of a node, to avoid possible damage to it. These restrictions, in general, limit the achievable quality and characteristics of an antenna for wireless sensor nodes.

Nodes randomly scattered on the ground, for example, deployed from an aircraft, will land in random orientations, with the antennas facing the ground or being otherwise obstructed. This can lead to nonisotropic propagation of the radio wave, with considerable differences in the strength of the emitted signal in different directions. This effect can also be caused by the design of an antenna, which often results in considerable differences in the spatial propagation characteristics (so-called lobes of an antenna).

Antenna design is an issue in itself and is well beyond the scope of this book. Some specific considerations on antenna design for wireless sensor nodes are discussed in [115, Chap. 8].

4.4 Further reading

Jointly optimizing coding and modulation Biglieri et al. [79] consider coding and modulation from an information-theoretic perspective for different channel models, including the AWGN, flat fading channels and block fading channels. Specifically, the influence of symbol-by-symbol power control at the transmitter in the presence of channel-state information such that deep fades are answered with higher output powers ("channel inversion"), of receiver diversity and interleaving and of coding schemes with unequal protection (i.e., user bits of different importance are encoded differently) on the channel capacity are discussed. One particularly interesting result is that the capacity of a Rayleigh fading channel with power control can be higher than the capacity of an AWGN channel with the same average radiated power.

DSSS in WSN Some efforts toward the construction of DSSS transceivers for wireless sensor networks with their space and power constraints are described in references [155, 280, 281]. In addition, Myers et al. [580] discuss low-power spread-spectrum transceivers for IEEE 802.11.

Energy efficiency in GSM Reducing energy consumption is an issue not only in wireless sensor networks but also in other types of systems, for example, cellular systems. For the interested: advanced signal processing algorithms for reducing power consumption of GSM transceivers are discussed in references [525].