Module 4

Overview of wireless sensor network and architecture

Overview of Wireless Sensor Networks: Challenges for Wireless Sensor Networks, Enabling Technologies for Wireless Sensor Networks.

Architectures: Single-Node Architecture - Hardware Components, Energy Consumption of Sensor Nodes, Operating Systems and Execution Environments, Network Architecture-Sensor Network Scenarios, Optimization Goals and Figures of Merit, Design principles for WSNs, Service interfaces of WSNs Gateway Concepts.

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.
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 8 Introduction 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.

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

Required mechanisms

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 – socalled "self-location".

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.

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 10 Introduction 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.

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

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.

Enabling technologies for wireless sensor networks

Building a wireless sensor networks has only become possible with some fundamental advances in enabling technologies as in Figure 4.1. First and foremost among these technologies is the miniaturization of hardware. Smaller feature sizes in chips have driven down the power consumption of the basic components of a sensor node to a level that the constructions of WSNs can be contemplated. This is particularly relevant to microcontrollers and memory chips as such, but also, the radio modems, responsible for wireless communication, have become much more energy efficient. Reduced chip size and improved energy efficiency is accompanied by reduced cost, which is necessary to make redundant deployment of nodes affordable. Next to processing and communication, the actual sensing equipment is the third relevant technology. Here, however, it is difficult to generalize because of the vast range of possible sensors These three basic parts of a sensor node have to accompanied by power supply. This requires, depending on application, high

capacity batteries that last for long times, that is, have only a negligible self-discharge rate, and that can efficiently provide small amounts of current. Ideally, a sensor node also has a device for energy scavenging, recharging the battery with energy gathered from the environment – solar cells or vibration-based power generation are conceivable options.

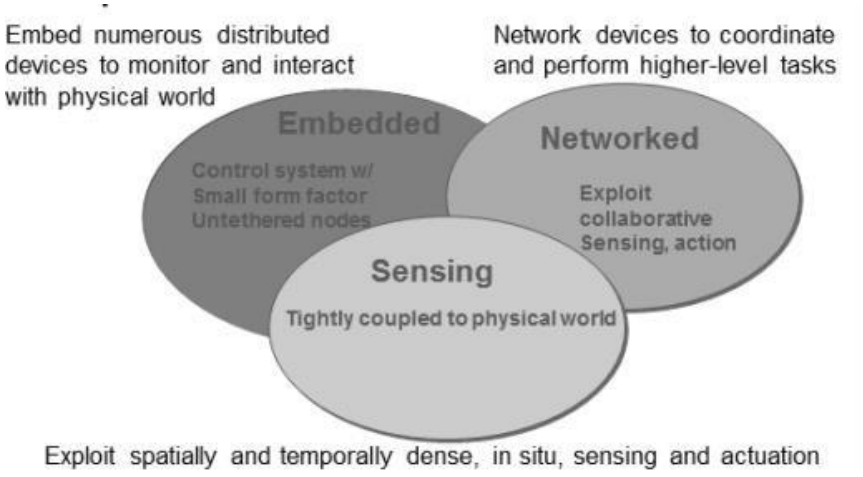

Figure 4.1: Enabling technologies of wireless Sensor Networks

Such a concept requires the battery to be efficiently chargeable with small amounts of current, which is not a standard ability. Both batteries and energy scavenging are still objects of ongoing research. The counterpart to the basic hardware technologies is software. The first question to answer here is the principal division of tasks and functionalities in a single node – the architecture of the operating system or runtime environment. This environment has to support simple retasking, cross-layer information exchange, and modularity to allow for simple maintenance. This software architecture on a single node has to be extended to a network architecture, where the division of tasks between nodes, not only on a single node, becomes the relevant question – for example, how to structure interfaces for application programmers. The third part to solve then is the question of how to design appropriate communication

Single-Node Architecture:

The single node architecture of a WSN is as shown in Figure 4.2.

Hardware Components: Choosing the hardware components for a wireless sensor node, obviously the applications has to consider size, costs, and energy consumption of the nodes. A basic sensor node comprises five main components such as Controller, Memory, Sensors and Actuators, Communication devices and Power supply Unit.

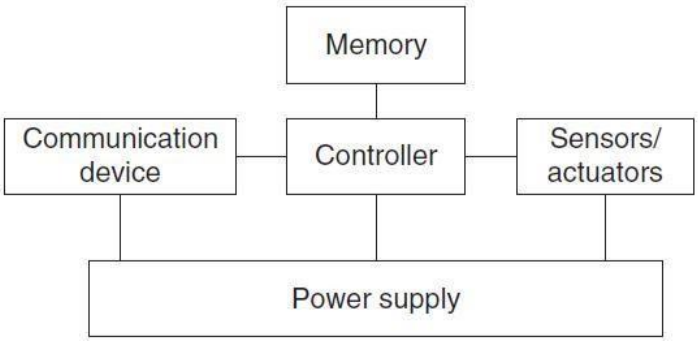

Figure4.2: Sensor node Hardware components

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

For General-purpose processors applications microcontrollers are used. These are highly overpowered, and their energy consumption is excessive. These are used in embedded systems. Some of the key characteristics of microcontrollers are particularly suited to embedded systems are their flexibility in connecting with other devices like sensors and they are also convenient in that they often have memory built in.

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.

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. *Examples:* Intel Strong ARM, Texas Instruments MSP 430, Atmel ATmega.

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

Communication Device:

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

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 networks. The case of wireless communication is considerably more interesting because it include radio frequencies. Radio Frequency (RF)-based communication is by far the most relevant one as it best fits the requirements of most WSN applications.

Transceivers: For Communication, both transmitter and receiver are required in a sensor node to convert a bit stream coming from a microcontroller and convert them to and from radio waves. For two tasks a combined device called transceiver is used.

Transceiver structure has two parts as shown in figure 4.3 Radio Frequency (RF) front end and the baseband part.

1. The radio frequency front end performs analog signal processing in the actual radio frequency Band.

2. The baseband processor performs all signal processing in the digital domain and communicates with a sensor node's processor or other digital circuitry.

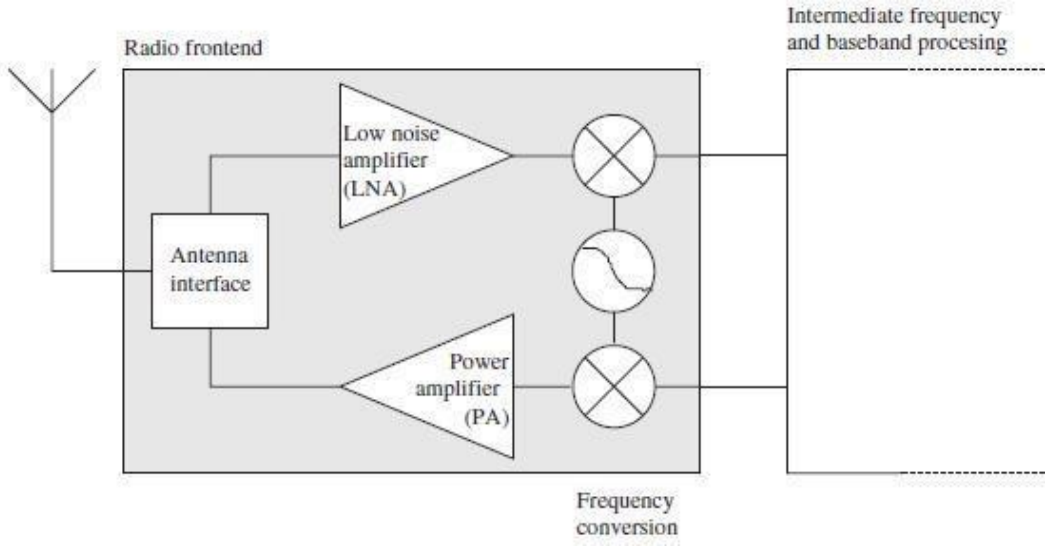

Figure 4.3 RF front end

- 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. 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.

- 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 fRF is multiplied in a mixer with a fixed-frequency signal from the local oscillator (frequency fLO). The resulting intermediate-frequency signal has frequency fLO − fRF. Depending on the RF front end architecture, other elements like filters are also present.

Transceiver tasks and characteristics:

- *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.

- *Power consumption and energy efficiency:* The simplest interpretation of energy efficiency is the energy required to transmit and receive a single bit.
- *Carrier frequency and multiple channels:* Transceivers are available for different carrier frequencies; evidently, it must match application requirements and regulatory restrictions.

- *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.

- *Data rates:* Carrier frequency and used bandwidth together with modulation and coding determine the gross data rate.

- *Modulations:* The transceivers typically support one or several of on/off-keying, ASK, FSK, or similar modulations.

- *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: NF= $\frac{SNR_I}{}$. It describes the degradation of SNR due to the element's operation and is typically given in dB: NF dB= SNR*I* dB − SNR*O* 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.

SNR_O

- *Receiver sensitivity*: The receiver sensitivity (given in dBm) specifies the minimum signal power at the receiver needed to achieve a prescribed *Eb/N*0 or a prescribed bit/packet error rate.

- *Range:* The range of a transmitter is clear. The range is considered in absence of interference; it evidently depends on the maximum transmission power, on the antenna characteristics.

- *Blocking performance:* The blocking performance of a receiver is its achieved bit error rate in the presence of an interferer.

- *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 signal strength at which an incoming data packet has been received can provide useful information 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.

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

Sensors and actuators:

The actual interface to the physical world: devices that can observe or control physical parameters of the environment. **Sensors** can be roughly categorized into three categories as

- *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.

- **Passive, narrow-beam sensors** These sensors are passive as well, but have a well-defined notion of direction of measurement.

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

Actuators: Actuators are just about as diverse as sensors, yet for the purposes of designing a WSN that converts electrical signals into physical phenomenon.

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). There are essentially two aspects: Storing energy and Energy scavenging.

Storing energy: Batteries

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

Table 4.1Energy densities for various primary and secondary battery types

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

Upon these batteries the requirements are

- *Capacity:* They should have high capacity at a small weight, small volume, and low price. The main metric is energy per volume, J/cm3.

- *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.

- *Self-discharge:* Their self-discharge should be low. Zinc-air batteries, for example, have only a very short lifetime (on the order of weeks).

- *Efficient recharging:* Recharging should be efficient even at low and intermittently available recharge power.

- *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.

- *DC–DC Conversion:* Unfortunately, batteries 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. 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. The DC – DC converter does consume energy for its own operation, reducing overall efficiency.

Energy scavenging: Depending on application, high capacity batteries that last for long times, that is, have only a negligible self-discharge rate, and that can efficiently provide small amounts of current. Ideally, a sensor node also has a device for **energy scavenging**, recharging the battery with energy gathered from the environment – solar cells or vibration-based power generation are conceivable options.

ENERGY CONSUMPTION OF SENSOR NODES:

In previous section we discussed about energy supply for a sensor node through batteries that 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, the memory, and type of the sensors. One method to reduce power consumption of these components is designing low-power chips, it is the best starting point for an energy-efficient sensor node. But any advantages gained by such designs can easily be squandered/ wasted when the components are improperly operated. Second method for energy efficiency in wireless sensor node is reduced functionality by using multiple states of operation with reduced energy consumption. These modes can be introduced for all components of a sensor node, in particular, for controller, radio front end, memory, and sensors.

Microcontroller energy consumption: For a controller, typical states are "active", "idle", and "sleep". A radio modem could turn transmitter, receiver, or both on or off. At time *t*1, the microcontroller is to be put into sleep mode should be taken to reduce power consumption from Pactive to Psleep as shown in Figure 4.4. If it remains active and the next event occurs at time tevent, then a total energy is Eactive = Pactive (tevent − t1). On the other hand, requires a time τdown until sleep mode has been reached. Let the average power consumption during this phase is (Pactive + Psleep)/2. Then, Psleep is consumed until tevent. The energy saving is given by

$$E_{saved} = (t_{event} - t1)P_{active} - (\tau_{down}(P_{active} + P_{sleep})/2 + (t_{event} - t1 - \tau_{down})P_{sleep) \quad \text{----------} (4)$$

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

$$E_{overhead} = \tau_{Up}(P_{active} + P_{sleep})/2 \quad \text{----------} (5)$$

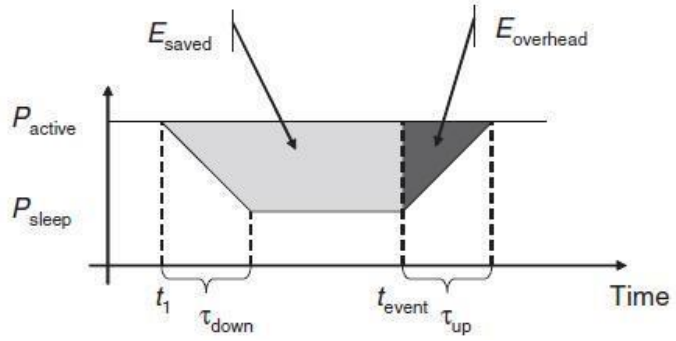

Figure 4.4 Energy savings and overheads for sleep modes

Switching to a sleep mode is only beneficial if *E*overhead < *E*saved or,

equivalently, if the time to the next event is sufficiently large:

$$(t_{event} - t_1) > \frac{1}{2}\left(\tau_{down} + \frac{P_{active} + P_{sleep}}{P_{active} - P_{sleep}}\tau_{up}\right) \quad \text{-- (6)}$$

Examples:

Intel StrongARM

The Intel StrongARM 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 features a wider range of operation modes: One fully operational mode, which consumes about 4.2 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 till running, which can be used for scheduled wake ups, and still consumes only about 6 µW.

Atmel ATmega

The Atmel ATmega 128L 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.

Memory energy consumption: The most relevant kinds of memory are on-chip memory and FLASH memory. Off-chip RAM is rarely 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 energy consumption. Read times and read energy consumption tend to be quite similar between different types of FLASH memory. Energy consumption necessary for reading and writing to the Flash memory is used on the Mica nodes. Hence, writing to FLASH memory can be a time- and energy-consuming task that is best avoided if somehow possible.

Radio transceivers energy consumption: 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.

The energy consumed by a transmitter is due to two sources one part is due to RF signal generation, which mostly depends on chosen modulation and target distance. Second part is due to electronic components necessary for frequency synthesis, frequency conversion, filters, and so on. The transmitted power is generated by the amplifier of a transmitter. Its own power consumption Pamp depends on its architecture Pamp = αamp + βampPtx. where αamp and βamp are constants depending on process technology and amplifier architecture. 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.

$$E_{\text{tx}}(n, R_{\text{code}}, P_{\text{amp}}) = T_{\text{start}} P_{\text{start}} + \frac{n}{R R_{\text{code}}}(P_{\text{txElec}} + P_{\text{amp}}) \quad \text{--------- (7)}$$

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*start*P*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 $\frac{n}{RR_{code}}$. During this time of actual reception, receiver circuitry has to be powered up, requiring a (more or less constant) power of *P*rxElec.

$$E_{\text{rcvd}} = T_{\text{start}} P_{\text{start}} + \frac{n}{R R_{\text{code}}} P_{\text{rxElec}} + n E_{\text{decBit}} \quad \text{---------- (8)}$$

Power consumption of sensor and actuators:

Providing any guidelines about the power consumption of the actual sensors and actuators is impossible because of the wide variety of these devices. For example, passive light or temperature sensors – the power consumption can possibly be ignored in comparison to other devices on a wireless node. For others, active devices like sonar(A measuring instrument that sends out an acoustic pulse in water and measures distances in terms of time for the echo of the pulse to return), power consumption can be quite considerable in the dimensioning of power sources on the sensor node, not to overstress batteries.

OPERATING SYSTEMS AND EXECUTION ENVIRONMENTS:

Embedded operating systems:

- An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs.
- For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware.
- An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, that is specifically designed for a particular function.
- Embedded operating systems are designed to be used in embedded computer systems. They are able to operate with a limited number of resources. They are very compact and extremely efficient by design.

TinyOS:

- TinyOS is an open-source, flexible and application-specific operating system for wireless sensor networks.

- Wireless sensor network consists of a large number of tiny and low-power nodes, each of which executes simultaneous and reactive programs that must work with strict memory and power constraints.

- TinyOS meets these challenges and has become the platform of choice for sensor network such as limited resources and low-power operation.

Salient features of TinyOS are

- A simple event-based concurrency model and split-phase operations that influence the development phases and techniques when writing application code.

- It has a component-based architecture which provides rapid innovation and implementation while reducing code size as required by the difficult memory constraints inherent in wireless sensor networks.

- TinyOS's component library includes network protocols, distributed services, sensor drivers, and data acquisition tools.

- TinyOS's event-driven execution model enables fine grained power management, yet allows the scheduling flexibility made necessary by the unpredictable nature of wireless communication and physical world interfaces.

Programming paradigms and application programming interfaces:

- **Concurrent Programming:** Concurrent processing is a computing model in which multiple processors execute instructions simultaneously for better performance. Concurrent means something that happens at the same time as something else. Tasks are broken down into subtasks that are then assigned to separate processors to perform simultaneously, instead of sequentially as they would have to be carried out by a single processor. Concurrent processing is sometimes said to be synonymous with parallel processing. This is depicted in Figure 4.5.

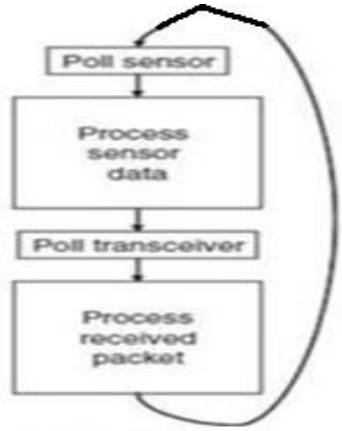

Figure4.5: Sequential Programming module

- **Process-based concurrency:** Most modern, general-purpose operating systems support concurrent (seemingly parallel) execution of multiple processes on a single CPU. Using processes you are forced to deal with communication through messages, which is the Erlang (A unit of traffic intensity in telephone system) way of doing communication. Data is not shared, so there is no risk of data corruption. Fault-tolerance and scalability is the main advantages of using processes vs. threads. Another advantage of processes is that they can crash and you are perfectly ok with that, because you just restart them (even across network hosts). If thread crashes, it may crash the entire process, which may bring down your entire application. The mechanism of process based programming is as shown in Figure 4.6.

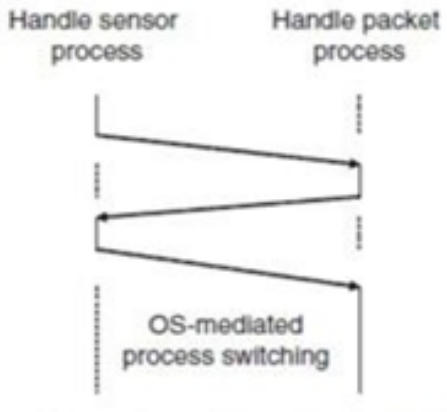

Figure 4.6: Process based programming

- **Event-based programming:**

In computer programming, event-driven programming is a programming paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs/threads. Event-driven programming is the dominant paradigm used in Graphical User Interfaces (GUI-type of user interface that allows users to interact with electronic devices through graphical icons) and other applications. It is as shown in Figure 4.7 The system essentially waits for any event to happen, where an event typically can be the availability of data from a sensor, the arrival of a packet, or the expiration of a timer. Such an event is then handled by a short sequence of instructions that only stores the fact that this event has occurred and stores the necessary information.

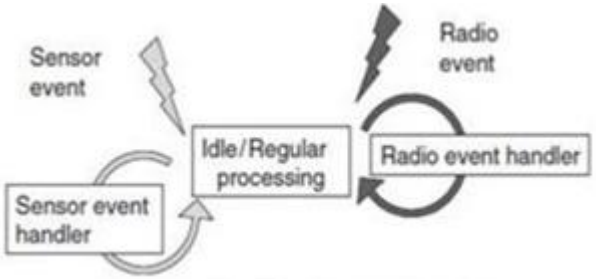

Figure 4.7 Event Based Programming

Interfaces to the operating system: A boundary across which two independent systems meet and act on or communicate with each other. In computer technology, there are several types of interfaces. User interface - the keyboard, mouse, menus of a computer system. The user interface allows the user to communicate with the operating system. Stands for "Application Programming Interface." An API is a set of commands, functions, protocols, and objects (wireless links, nodes) that programmers can use to create software or interact with an external system (sensors, actuators, transceivers). It provides developers with standard commands for performing common operations so they do not have to write the code from scratch.

Structure of operating system and protocol stack:

The traditional approach to communication protocol structuring is to use layering: individual protocols are stacked on top of each other, each layer only using functions of the layer directly. This layered approach has great benefits in keeping the entire protocol stack manageable, in containing complexity, and in promoting modularity and reuse. For the purposes of a WSN, however, it is not clear whether such a strictly layered approach will serve. A protocol stack refers to a group of protocols that are running concurrently that are employed for the implementation of network protocol suite. The protocols in a stack determine the interconnectivity rules for a layered network model such as in the OSI or TCP/IP models.

Dynamic energy and power management:

Switching individual components into various sleep states or reducing their performance by scaling down frequency and supply voltage and selecting particular modulation and coding are prominent examples for improving energy efficiency. To control these possibilities, decisions have to be made by the operating system, by the protocol stack, or potentially by an application when to switch into one of these states. Dynamic Power Management (DPM) on a system level is the problem at hand. One of the complicating factors to DPM is the energy and time required for the transition of a component between any two states. If these factors were negligible, clearly it would be optimal to always

& immediately go into the mode with the lowest power consumption possible.

NETWORK ARCHITECTURE:

It introduces the basic principles of turning individual sensor nodes into a wireless sensor network. In this optimization goals of how a network should function are discussed as

- Sensor network scenarios
- Optimization goals and figures of merit
- Gateway concepts

SENSOR NETWORK SCENARIOS:

Types of sources and sinks:

Source is any unit in the network that can provide information (sensor node). A sink is the unit where information is required, it could belong to the sensor network or outside this network to interact with another network or a gateway to another larger Internet**.** Sinks are illustrated by Figure, showing sources and sinks in direct communication.

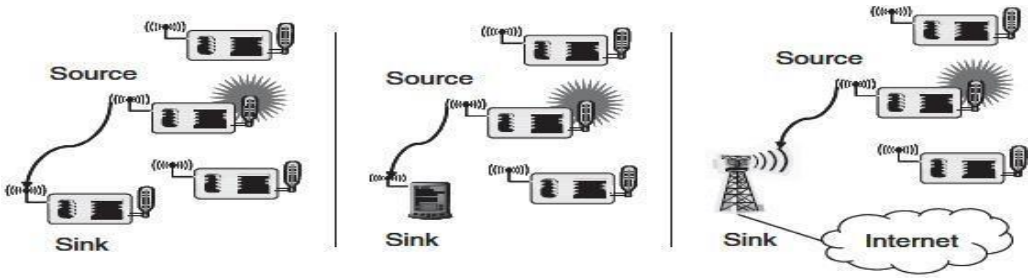

Figure 4.8 Three types of sinks in a very simple, single-hop sensor network

Single-hop versus multi-hop networks:

Because of limited distance the direct communication between source and sink is not always possible. In WSNs, to cover a lot of environment the data packets taking multi hops from source to the sink. To overcome such limited distances it better to use relay stations, The data packets taking multi hops from source to the sink as shown in Figure 4.8, Depending on the particular application of having an intermediate sensor node at the right place is high.

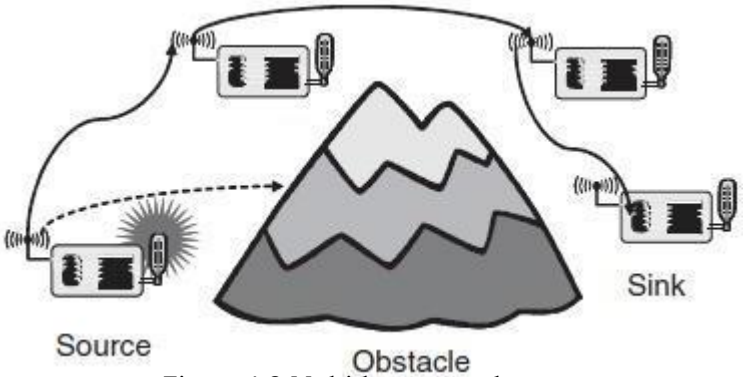

Figure 4.9 Multi-hop networks

Multi-hopping also to improves the energy efficiency of communication as it consumes less energy to use relays instead of direct communication, the radiated energy required for direct communication over a distance d is cdα (c some constant, α ≥ 2 the path loss coefficient) and using a relay at distance d/2 reduces this energy to 2c(d/2)α. The same is depited in Figure 4.9.
This calculation considers only the radiated energy. It should be pointed out that only multi-hop 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. Cooperative relaying (reconstruction in case of erroneous packet reception) techniques are not considered here.

Multiple sinks and sources:

In many cases, multiple sources and multiple sinks present. Multiple sources should send information to multiple sinks. Either all or some of the information has to reach all or some of the sinks. This is illustrated in Figure4.10.

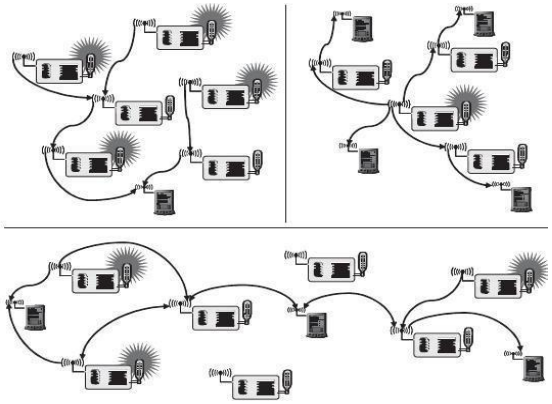

Figure4.10: Multiple sources and/or multiple sinks

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.

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

a. Node mobility
b. Sink mobility
c. Event mobility

a) **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 to function correctly.

b) **Sink Mobility:** The information sinks can be mobile. The scenario is shown in Figure 4.11. 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 separate, unrelated requests.

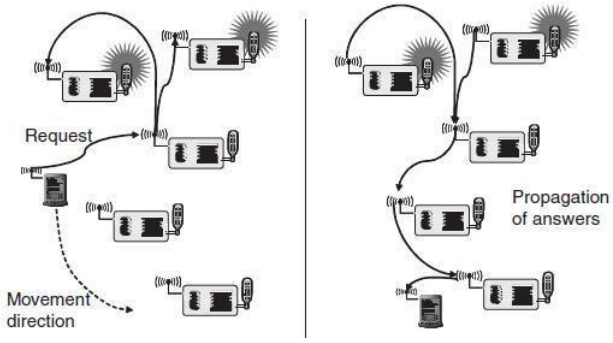

Figure4.11 Sink mobility: A mobile sink moves through a sensor network as information is being retrieved *on its behalf*

Event Mobility: 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. 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. This notion is described by Figure 4.12, where the task is to detect a moving elephant and to observe it as it moves around Area of sensor nodes detecting an event – an elephant– 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).

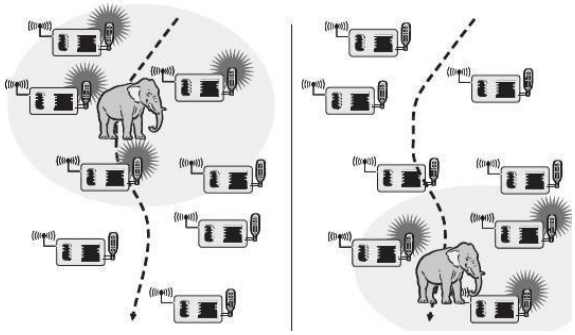

Figure4.12 Frisbee model

OPTIMIZATION GOALS AND FIGURES OF MERIT:

For all WSN scenarios and application types have to face the challenges such as

- How to optimize a network and How to compare these solutions?
- How to decide which approach is better?
- How to turn relatively inaccurate optimization goals into measurable figures of merit?

For all the above questions the general answer is obtained from

- Quality of Service
- Energy efficiency
- Scalability
- Robustness

Quality of service: WSNs differ from other conventional communication networks in the type of service they offer. These networks essentially only move bits from one place to another. Some generic possibilities are

- **Event detection/reporting probability-** The probability that an event that actually occurred is not detected or 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.

- **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 -**It 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, the average/maximum absolute or relative error with respect to the actual function.

- **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.

Energy efficiency: Energy efficiency should be optimization goal. The most commonly considered aspects are:

- **Energy per correctly received bit-**How much energy is spent on average to transport one bit of information (payload) from the transmitter to the receiver.

- **Energy per reported (unique) event-**What is the average energy spent to report one event

- **Delay/energy trade-offs-**"urgent" events increases energy investment for a speedy reporting events. Here, the trade-off between delay and energy overhead is interesting
- **Network lifetime** The time for which the network is operational
- **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
- **Time to partition-**When does the first partition of the network in two (or more) disconnected parts occur?

- **Time to loss of coverage** the time when for the first time any spot in the deployment region is no longer covered by any node's observations.

- **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.

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 obviously essential requirement. 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. 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.

Robustness: 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 by finding other routes.

GATE WAY CONCEPTS:

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 to read the temperature sensors in one's home while traveling and accessing the Internet via a wireless.
- Wireless sensor networks should also exhibit an appropriate robustness
- They should not fail just because of a limited number of nodes run out of energy or because of their environment changes and breaks existing radio links between two nodes.
- If possible, these failures have to be compensated by finding other routes.

Figure 4.13 shows this networking scenario, 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. The WSN support standard wireless communication technologies such as IEEE 802.14. The design of gateways becomes much more challenging when considering their logical design. One option is to regard a gateway as a simple router between Internet and sensor network.

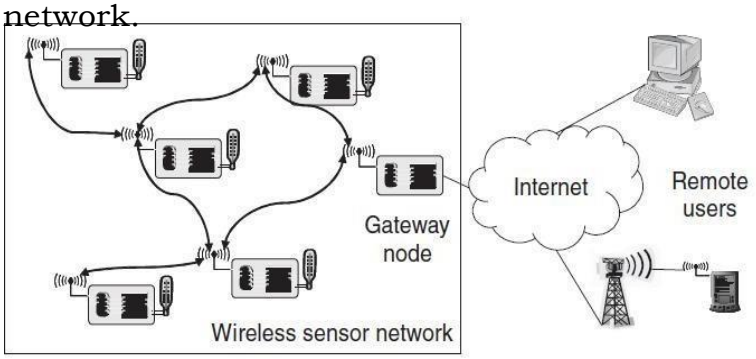

Figure 4.13: A wireless sensor network with gateway node, enabling access to remote clients via the Internet

WSN to Internet communication:

Assume that the initiator of a WSN – Internet communication resides in the WSN.

- For example, a sensor node wants to deliver an alarm message to some Internet host.

- The first problem to solve is 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
- 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

- 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.

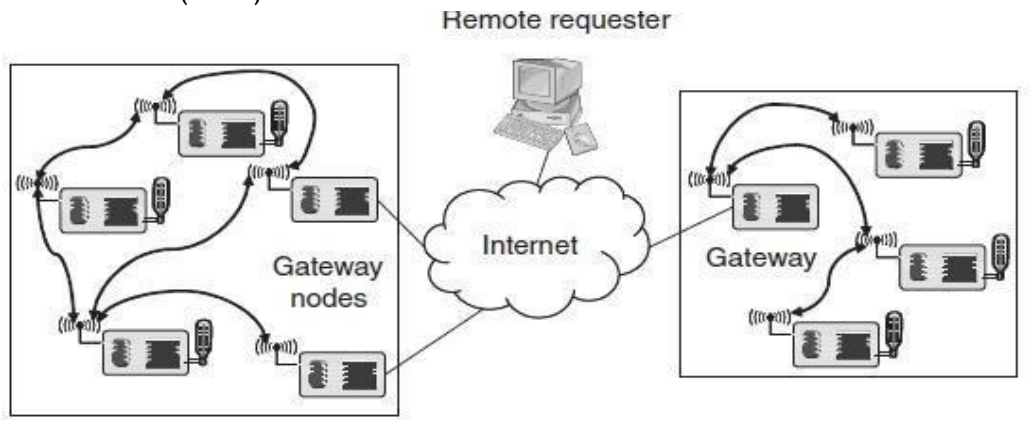

Figure4.14: A wireless Sensor Network with gateway node, enabling access to remote clients via the WSN

Internet to WSN communication: The case of an Internet-based entity trying to access services of a WSN is even more challenging.

- This is fairly simple if this requesting terminal is able to directly communicate with the WSN.

- 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 is 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.
- The requesting terminal can instead send a properly formatted request to this gateway, which acts as an application-level gateway
- The gateway translates this request into the proper intra sensor network protocol interactions
- The gateway can then mask, for example, a data-centric data exchange within the network behind an identity-centric exchange used in the Internet as in Figure 4.15.
- 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
- 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

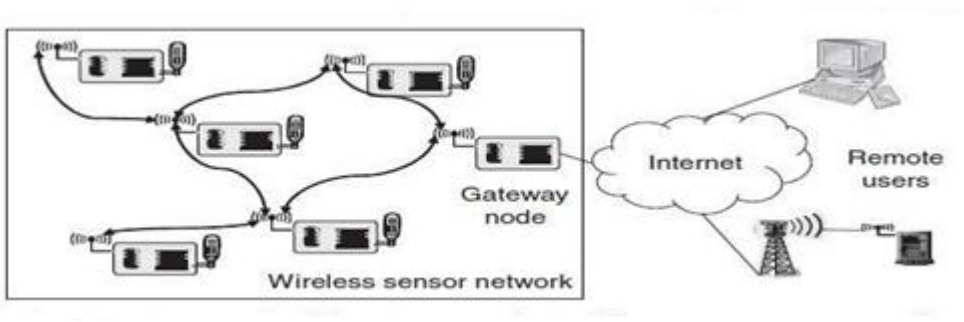

Figure 4.15: A wireless sensor network with gateway node enabling access to remote clients via internet

WSN tunneling:

- 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.
- 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).
- Wireless tunneling is depicted in Figure 4.16.

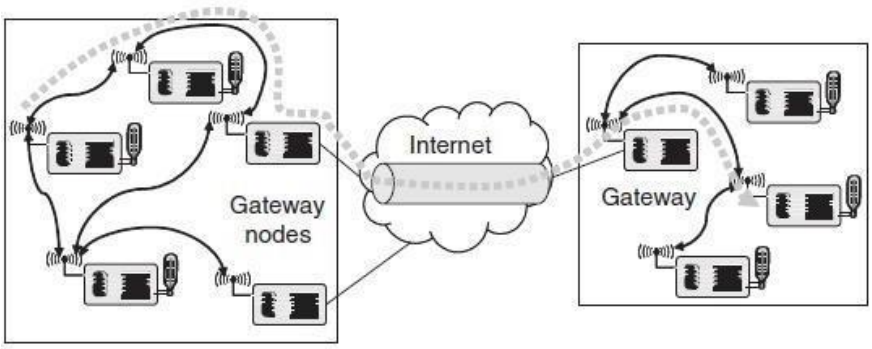

Figure 4.16: Connecting two WSNs with a tunnel over the Internet