Principle of Wireless Sensor Network

Wireless Sensor Network is a network of autonomous sensor nodes that are capable of collecting, processing and transmitting data wirelessly. WSNs are used in a wide range of applications, including environmental monitoring, surveillance, industrial automation and healthcare. The principles of WSNs can be summarized as follows:

- Resource constraints: Sensor nodes in a WSN are typically low-power, and have limited computational and storage capabilities. They rely on battery power and are expected to operate for extended periods without maintenance or battery replacement.
- Communication: Sensor nodes communicate with each other wirelessly, typically using radio frequency (RF) transmissions. The communication range of each node is limited, and nodes must cooperate to rely data over longer distances.
- Data aggregation: To conserve energy and reduce bandwidth requirements, sensor nodes often perform local data aggregation before transmitting data to the base station or sink node.
- Routing: Sensor nodes must determine the best route for data transmission to the base station, taking into account the network topology, communication range, and energy constraints of each node
- Security: Sensor networks can be vulnerable to security threats, including eavesdropping, tampering and node compromise. Security mechanisms such as encryption, authentication and access control are necessary to orotect the network and its data.
- Data Processing: Sensor nodes may perform basic processing of the collected data, such as filtering or compression, before transmitting it to the base station. Data processing algorithms may also be distributed across multiple nodes to improve efficiency and reduce latency
- Application-specific functionality: WSN are designed for specific applications and the functionality of the network is tailored to meet the requirements of the application. Examples of applications include environmental monitoring, surveillance and industrial automation.

Application Types

Application of wireless sensor networks can be classified based on the interaction pattern between sources and sinks. The classification is as follows:

- Event detection : In this category of application, Sensor nodes should report to the sink once they have detected the occurrence of a specified event. The simplest event scan be detected locally by a single sensor node in isolation (e.g a temperature threshold is exceeded). If several different events can occur, event classification is necessary.
- Periodic Measurements: In this type of applications sensor has to periodically report the measured values. The reporting period is application dependent.
- Function Approximation and edge detection: Physical values like temperature which changes from one place to another can be regarded as a function of location. A WSN can be used to approximate this unknown function using a limited number of samples taken at each individual sensor node this is function approximation. Similarly, finding edges or structures in such functions along the boundaries of patterns is called edge detection.
- Tracking: This includes applications where the source of an event is mobile (e.g an intruder in surveillance scenarios). The WSN can be used to report updates on the event source's position to the sink, like speed and direction.
- Deployment option
    1. Fixed Deployment: Well planned deployment of sensor nodes
    2. Random Deployment: By dropping a large number of nodes from an aircraft over a forest fire.

Challenges For Wireless Sensor Networks

Major challenges for WSN is in its characteristic requirements and required mechanisms

- Characteristic Requirements
  - Types of service: WSN is expected to provide meaningful information or actions about a given task. The concept 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: The amount and quality of information that can be extracted at given sinks about the observed objects or area so adapted quality concepts like reliable detection of events or the approximation quality 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: 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. An alternative energy supplies might also be a 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. When quality of service investing more energy can increase quality but decrease lifetime. Concepts to harmonize these trade-offs are required.
  - 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. The density can vary over time and space because nodes fail or move, the density also does not have to homogeneous in the entire network and the network should adapt to such variations.
  - Programmable: These nodes should be programmable and their programming must be changeable during operation when new tasks become important.
  - Maintainability: The WSN has to monitor its own health and status to change operational parameters or to choose different trade-offs. 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

  To realize these requirements, innovative mechanisms for a communication network have to be found as well as new architectures and protocol concepts
  - Multihop wireless communication: 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, 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 or more importantly the energy- efficient determination of a requested information.
  - Auto-Configuration: A WSN will have no 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.
  - Collaboration and in-network processing: in some applications, a single sensor is not able to decide whether an event has happened but several sensors provide 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.
  - Locality: 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.

- Exploit trade-offs: 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. 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.

Comparison with AD HOC network vs Wireless Sensor Networks

An ad hoc network is a network that is setup, literally for a specific purpose to meet a quickly

Node Architecture

A basic sensor node comprises five main components such as

- Controller
- Memory
- Sensors and Actuators
- Communication Devices
- Power Supply Unit

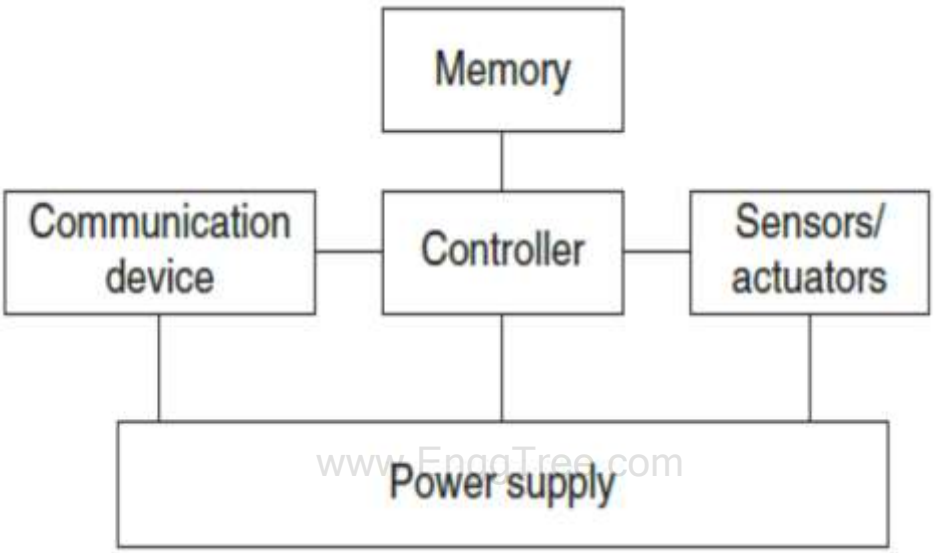


Fig 1.1 Hardware Components of Sensor Node

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 is the Central Processing Unit 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 processor is Digital Signal Processors. They are specifically geared, with respect to their architecture and their instruction set, for processing large amount of vectorial data, as is typically the case in signal processing applications.

Memory

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 ROM or more typically in Electrically Erasable Programmable Read –Only Memory(EEPROM) OR flash memory. 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.

Sensors and Actuators

Sensors

Sensors can be categorized into three

- 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 that is passive.
- Passive, narrow- beam sensors: These sensors are passive as well, but have well defined notion of direction of measurement.
- Active sensors: This sensor actively probes the environment Eg: A sonar or radar sensor or some types of seismic sensors, which generates shock waves by small explosions.

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

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
- 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.

- 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). 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. Some important elements of an RF front ends architecture are sketched in Figure:

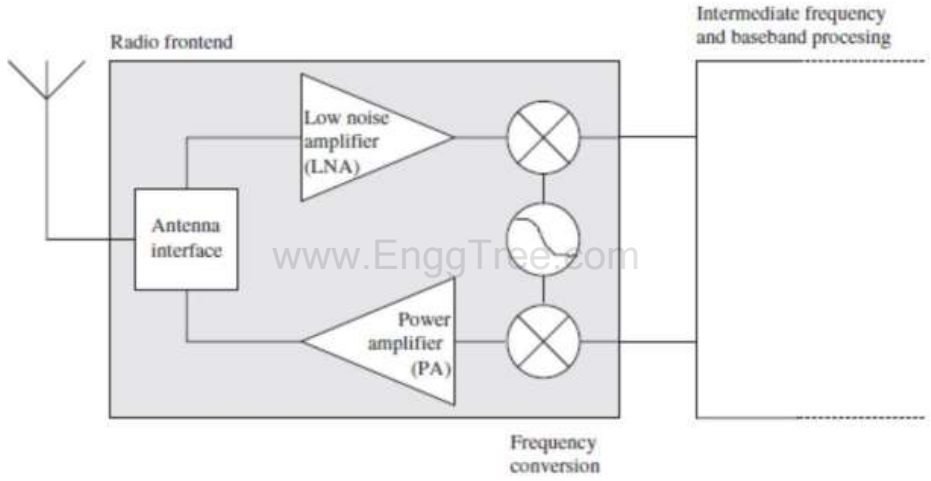


Fig 1.2.1 RF Front end

    - The Power Amplifier (PA): It accepts upconverted signals from the IF or baseband part and amplifies them for transmission over the antenna.
    - The Low Noise Amplifier (LNA): It 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.
- **Transceiver tasks and characteristics**: To select appropriate transceivers, a number of characteristics should be taken into account. The most important ones are:
    1. 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.

2. 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.
3. Carrier frequency and multiple channels: Transceivers are available for different carrier frequencies; evidently, it must match application requirements and regulatory restrictions.
4. 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.
5. Data rates: Carrier frequency and used bandwidth together with modulation and coding determine the gross data rate.

6. Modulations: The transceivers typically support one or several of on/off-keying, ASK, FSK, or similar modulations.
7. Coding: Some transceivers allow various coding schemes to be selected.
8. 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.
9. Noise figure: The noise figure, NF of an element is defined as the ratio of the Signal-to-Noise Ratio (SNR) ratio SNRI at the input of the element to the SNR ratio SNRO at the element's output. It describes the degradation of SNR due to the element's operation and is typically given in dB.
10. 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.
11. 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.

12. Receiver sensitivity: The receiver sensitivity (given in dBm) specifies the minimum signal power at the receiver needed to achieve a prescribed Eb/N0 or a prescribed bit/packet error rate.
13. Range: 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.
14. Blocking performance: The blocking performance of a receiver is its achieved bit error rate in the presence of an interferer.
15. 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.
16. 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
17. Voltage range: Transceivers should operate reliably over a range of supply voltages. Otherwise, inefficient voltage stabilization circuitry is required.

Power Supply Of Sensor Nodes

For untethered wireless sensor nodes, the power supply is a crucial system component. There are essentially two aspects:

a) First, storing energy and providing power in the required form
b) Second, attempting to replenish consumed energy by "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").
- **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 (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. 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. Also, the DC – DC converter does consume energy for its own operation, reducing overall efficiency.

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.

- 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 μW/cm2 indoors and 15 mW/cm2 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.
- Temperature gradients: Differences in temperature can be directly converted to electrical energy.
- 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 μW/cm3 up to 10, 000 μW/cm3 for some extreme cases (typical upper limits are lower).

- Pressure variations Somewhat akin to vibrations, a variation of pressure can also be used as a power source.

Energy Consumption of Sensor Nodes

Operation states with different power consumption

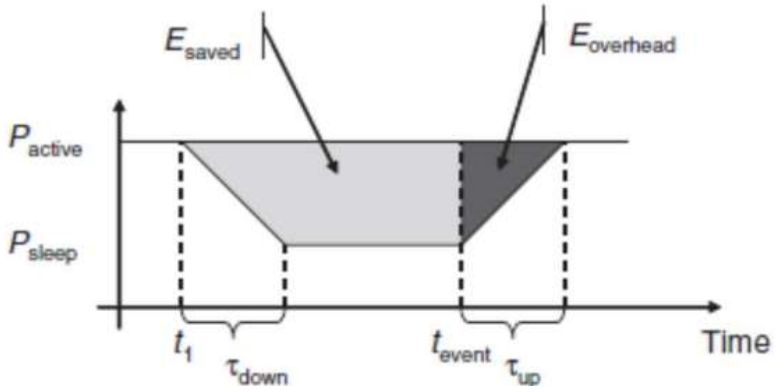


Fig 1.2.2 Energy Savings and Overheads for skeed Modes

Figure illustrates this notion based on a commonly used model. At time t1, 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 Pactive to Psleep. If it remains active and the next event occurs at time tevent, then a total energy of Eactive = Pactive (tevent − t1) has be spent uselessly idling. Putting the component into sleep mode, on the other hand, requires a time τdown until sleep mode has been reached; as a simplification, assume that the average power consumption during this phase is (Pactive + Psleep)/2. Then, Psleep is consumed until tevent. In total, τdown(Pactive + Psleep)/2 + (tevent − t1 − τdown)Psleep energy is required in sleep mode as opposed to (tevent − t1)Pactive when remaining active. The energy saving is thus

$$E_{saved} = (t_{event} - t_1) P_{active} - (\tau_{down}(P_{active} + P_{sleep})/2 + (t_{event} - t_1 - \tau_{down}) P_{sleep}).$$

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

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

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 Eoverhead < Esaved or, equivalently, if the time to the next event is sufficiently large:

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

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. 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).
- **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. Read times and read energy consumption tend to be quite similar between different types of FLASH memory. 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. 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.
- **Radio transceivers**: A radio transceiver has essentially two tasks: transmitting and receiving data between a pair of nodes. 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.

- **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. 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.

1.3 Network Architecture

Sensor Network Scenario

- Types of Sources and Sinks: 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.

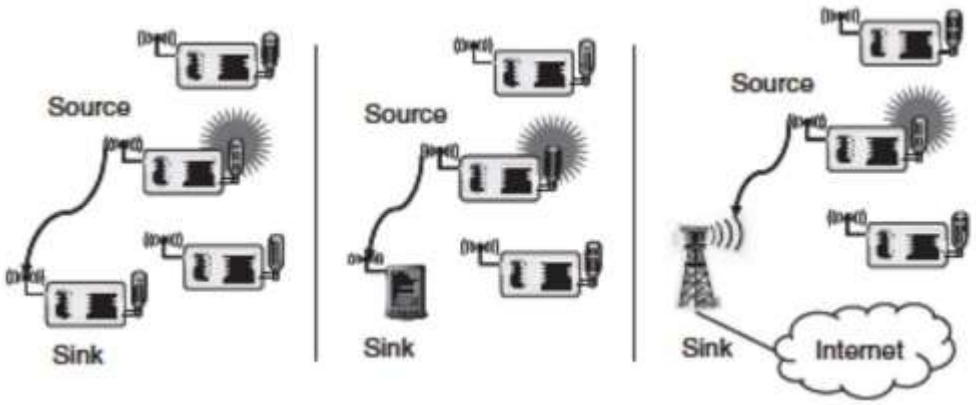


Fig: 1.3.1 Three types of sinks in a very simple, single-hop sensor network

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.

- 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 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).

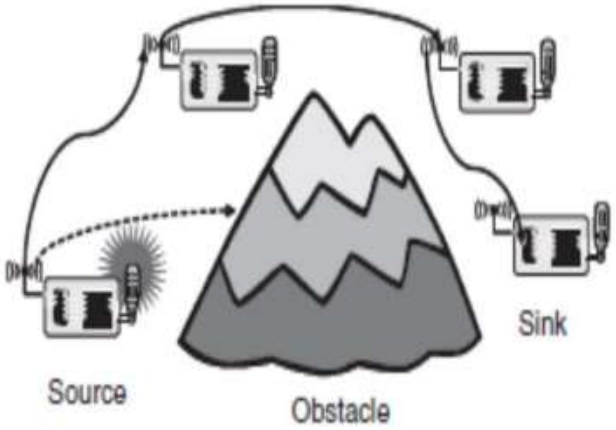


Fig: 1.3.2 Multihop networks: As direct communication is impossible because of distance and obstacles, multihop communication can circumvent the problem

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 1.3.2) 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. 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. The classification of 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.

- 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.

  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.

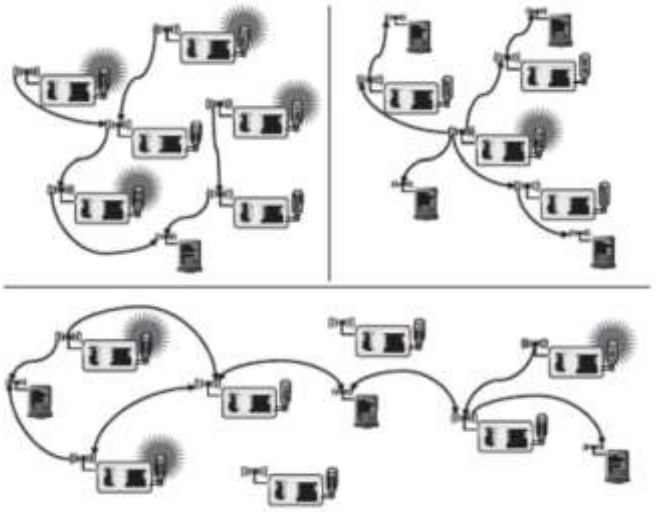


Fig 1.3.3 Multiple sources and multiple sinks

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. 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 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.

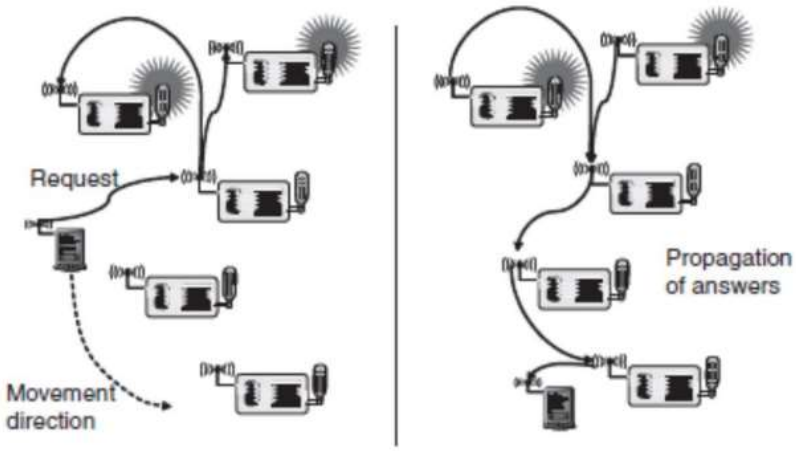


Fig 1.3.4 A mobile sink moves through a sensor network as information is being retrieved on its behalf

- 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 (which also describes algorithms for handling the "wakeup wavefront"). This notion is described by Figure 1.3.5, 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 1.3.5). 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.

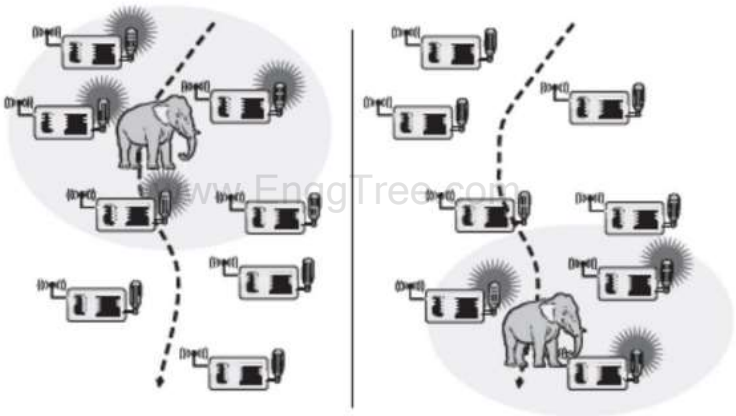


Fig 1.3.5 Area of sensor nodes detecting an event

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.

- **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? 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.

- **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 tradeoff 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:
    a) Time to first node death: When does the first node in the network run out of energy or fail and stop operating?
    b) Network half-life: When have 50% of the nodes run out of energy and stopped operating? Any other fixed percentile is applicable as well.
    c) 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.
    d) 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.

e) 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"

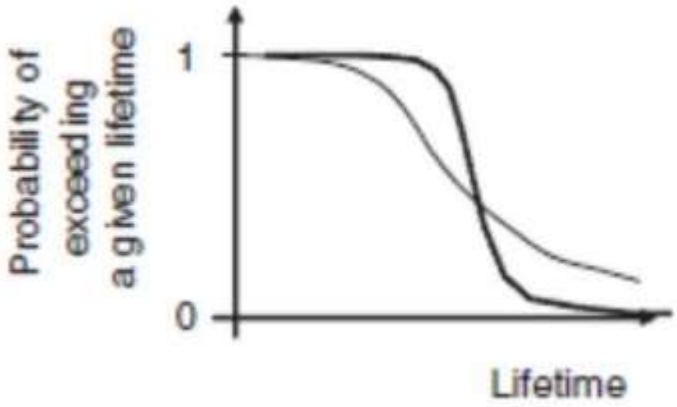


Fig: 1.3.6 Two probability curves of a node exceeding a given lifetime

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

- **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.

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. 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.

- **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 re-election vary considerably, depending on the purpose for which these hierarchies are used.

- **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.

- **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. The 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 of aggregate functions distinguishes duplicate-sensitive versus insensitive, summary versus exemplary, monotone versus non

monotone, and algebraic versus holistic. Functions like average, counting, or minimum can profit a lot from aggregation; holistic functions like the median are not amenable to aggregation at all.

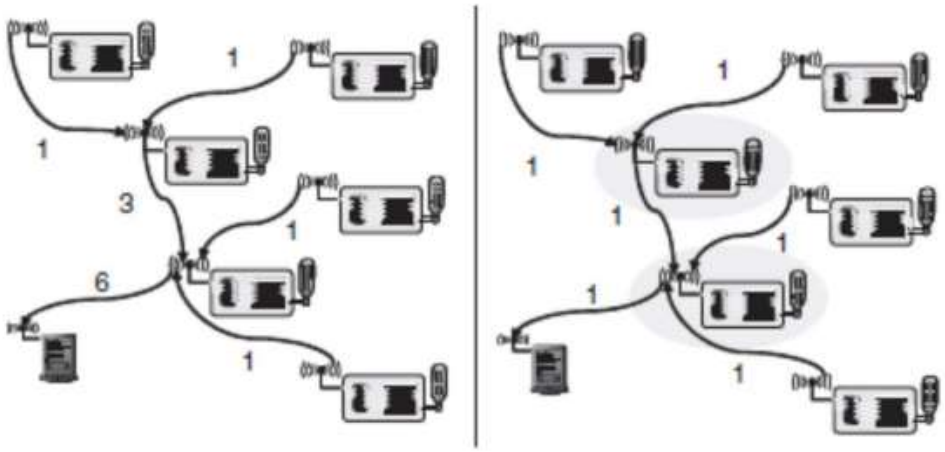


Fig : 1.3.7 Aggregation Example

- **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. 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. 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). 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.
- 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.

- **ADAPTIVE FIDELITY AND ACCURACY**: 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. 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.

- **DATA CENTRICITY**

Address Data, Not Nodes

In a wireless sensor network, 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. 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 innetwork 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, datacentric 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 neighbours. 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.

- **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). Once such information is available, it can simplify the design and operation of communication protocols and can improve their energy efficiency considerably.
- **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.
- **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).
- **COMPONENT-BASED PROTOCOL STACKS AND CROSS-LAYER OPTIMIZATION** Finally, a consideration about the implementation aspects of communication protocols in WSNs is necessary. 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. All wireless sensor networks will require some – even if only simple – form of physical, MAC and link layer 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. 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

1.4 IEEE 802.15.4 MAC

The Institute of Electrical and Electronics Engineers (IEEE) finalized the IEEE 802.15.4 standard in October 2003. 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 ZigBee5, 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.

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.

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

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 signalling 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. We start with the beaconed mode of IEEE 802.15.4.

Super frame structure

The coordinator of a star network operating in the beaconed mode organizes channel access and data transmission with the help of a super frame structure displayed in Figure 14. All super frames have the same length. The coordinator starts each super frame by sending a frame beacon packet. The frame beacon includes a super frame specification describing the length of the various components of the following super frame:

- The super frame 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.

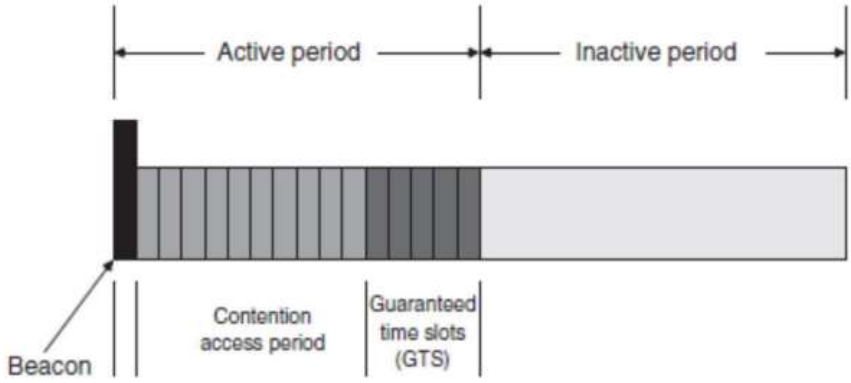


Fig 1.4.1 Super frame structure of IEEE 804.15.4

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. 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 super frame. 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 a GTS Desc Persistence Time 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 super frames, the slot is deallocated. The coordinator signals deallocation to the device by generating a GTS descriptor with start slot zero.

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 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 super frames and optionally switches off its transceiver until the next beacon.

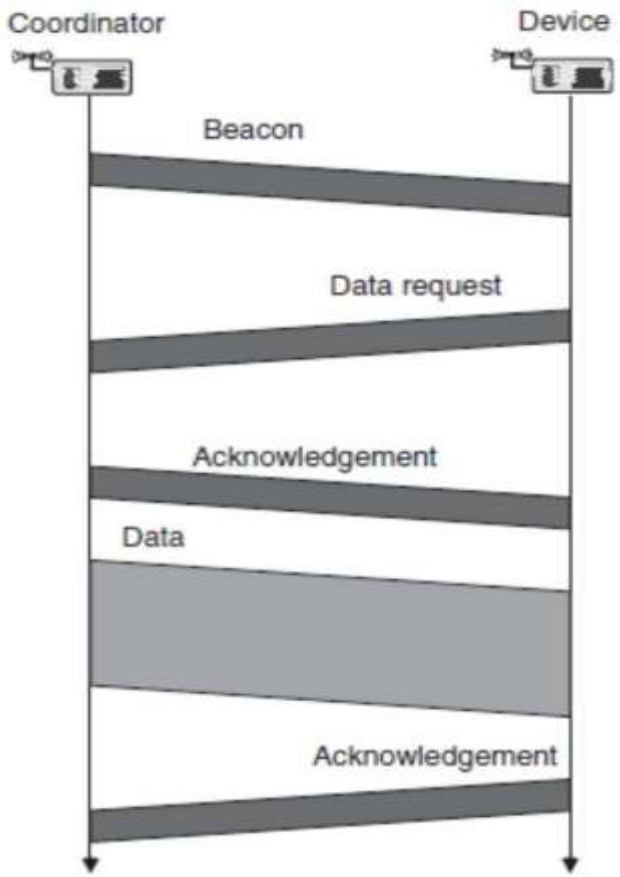


Fig 1.4.2 Handshake between coordinator and device when the device retrieves a packet

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

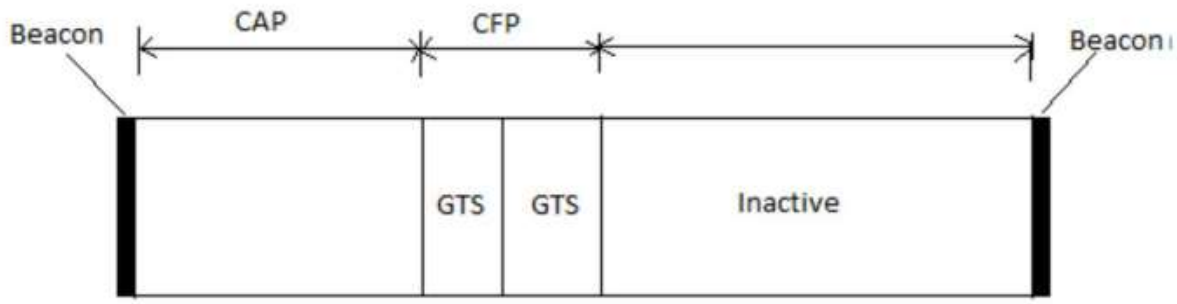
Nonbeaconed mode

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 back off 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:
  - To send a data/control packet to the coordinators,
  - 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.

ZigBee MAC PROTOCOL

ZigBee MAC protocol uses CSMA/CA or TDMA for accessing the shared medium. In ZigBee MAC data is packed into super frame. Super frame structure is shown in figure. The super frame may have an active and an inactive portion. During the inactive portion, the coordinator will not interact with its PAN and may enter a lowpower mode.



The active portion consists of contention access period (CAP) and contention free period (CFP). Any device that communicates during the CAP will compete with other devices using

a slotted CSMA/CA mechanism. On the other hand, the CFP contains guaranteed time slots (GTSs), a TDMA approach. The GTSs always placed at the end of the active super frame starting at a slot boundary just following the CAP. The network coordinator may allocate up to seven of these GTSs. A GTS can occupy more than one slot period. Synchronization is provided by beacon management. If TDMA mechanism is applied, device uses CFP field that contains GTSs. ZigBee MAC protocol while using CSMA/CA mechanism listen the channel continuously hence energy consumption is high. When it uses GTS management by providing a time slot to a device for transmission, only in period of time slot device has to transmit and for rest of the period it goes in sleep mode. Thus the energy consumption is reduced considerably.

After observing the simulation results it is obvious that ZigBee MAC protocol with GTS management is better if energy efficiency and throughput are more dominating factors. On the other hand T-MAC dominates ZigBee with GTS at low data rates in terms of energy consumption. But at low data rates throughput of T-MAC is lower than that of ZigBee with GTS. ZigBee with GTS has a problem of latency at higher data rates and synchronization, however many solution for resolving these problems are provided. So as per overall performance ZigBee MAC protocol is better for WBAN

1.5 PHYSICAL LAYER AND TRANSCEIVER DESIGN CONSIDERATIONS

The physical layer is mostly concerned with modulation and demodulation of digital data; this task is carried out by so-called transceivers. 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.

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 1mW). On the other hand, the overall transceiver (RF front end and baseband part) consumes much more energy than is actually radiated; A transceiver working at frequencies beyond 1 GHz takes 10 to 100mW of power to radiate 1 mW. These numbers coincide well with the observation that many practical transmitter designs have efficiencies below 10% 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; depending on the transceiver architecture, the idle mode's power consumption can be less or in the same range as the receive power. 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.

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.

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. 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.

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, for example, to parallelize user bits into m-ary symbols.
- Many m-ary modulation schemes require for increasing m an increased Eb/N0 ratio and consequently an increased radiated power to achieve the same target BER; others become less and less bandwidth efficient. 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 Eb/N0 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 start-up time easily dominates overall energy consumption, rendering any efforts in reducing the transmission time by choosing m-ary modulation schemes irrelevant.
- 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. 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.

- The cost of increasing the radiated power depends on the efficiency of the power amplifier 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.

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, 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. 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.

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. 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 $\alpha = 4$. 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 non-isotropic 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).

2.1 MAC PROTOCOLS FOR WIRELESS SENSOR NETWORKS

- The medium access control(MAC) is a sublayer of the data link layer of the open system interconnections reference model for data transmission.
- It is responsible for flow control and multiplexing for transmission medium.

Characteristics of MAC protocols:

Transmission Delay: Transmission delay is defined as the amount of time that a single message spends in the MAC protocol

Throughput: Throughput is defined by the rate at which messages are served. The throughput can be measured in messages or symbols per second.

Fairness: A MAC protocol is considered fair if it allocates a channel among the competing nodes according to some fairness criteria.

Scalability: Scalability describes the ability of the communication system to meet performance characteristics despite of the size of the network and number of competing nodes.

Robustness: Robustness is referred to as a composition of reliability, availability and dependability.

Stability: Stability describes how good the protocol handles fluctuation of traffic load over a sustainable period of time.

Goals of MAC Protocols:

- Minimize Energy Consumption
- Overhearing: Unnecessarily receive a packet destined to another node
- Idle listening : Staying active to receive even if there is no sender.

Design considerations for MAC protocols in wireless sensor networks

a) Balance of requirements

- The importance of energy efficiency for the design of MAC protocols is relatively new and many of the “classical” protocols like ALOHA and CSMA contain no provisions toward this goal.
- Other typical performance figures like fairness, throughput, or delay tend to play a minor role in sensor networks.
- Further important requirements for MAC protocols are scalability and robustness against frequent topology changes.

- It is caused by nodes powering down temporarily to replenish their batteries by energy scavenging, mobility, deployment of new nodes, or death of existing nodes.

b) Energy problems on the MAC layer

- A nodes transceiver consumes a significant share of energy.
- The transceiver has four main states: transmitting, receiving, idling, or sleeping.
- Transmitting is costly, receive costs often have the same order of magnitude as transmit costs, idling can be significantly cheaper but also about as expensive as receiving, and sleeping costs almost nothing but results in a "deaf" node.
- Some energy problems and design goals are mentioned below:
  - ✓ **Collisions**:
    - ❖ Collisions incur useless receive costs at the destination node, useless transmit costs at the source node, and the prospect to expend further energy upon packet retransmission.
    - ❖ Hence, collisions should be avoided, either by design (fixed assignment/TDMA or demand assignment protocols) or by appropriate collision avoidance/hidden-terminal procedures in CSMA protocols.
  - ✓ **Overhearing**
    - ❖ Unicast frames have one source and one destination node.

    - ❖ However, the wireless medium is a broadcast medium and all the source's neighbors that are in receive state hear a packet and drop it when it is not destined to them; these nodes overhear the packet.
  - ✓ **Protocol overhead**
    - ❖ Protocol overhead is induced by MAC-related control frames like, RTS and CTS packets or request packets in demand assignment protocols
  - ✓ **Idle listening**
    - ❖ A node being in idle state is ready to receive a packet but is not currently receiving anything.

- ❖ This readiness is costly and useless in case of low network loads; the idle state still consumes significant energy.
- ❖ Switching off the transceiver is a solution.
- ❖ A design constraint somewhat related to energy concerns is the requirement for low complexity operation.
- ❖ Sensor nodes shall be simple and cheap and cannot offer plentiful resources in terms of processing power, memory, or energy.
- ❖ Therefore, computationally expensive operations like complex scheduling algorithms should be avoided.

Important classes of MAC Protocols

a) **Fixed assignment protocols**
b) **Demand assignment protocols**
c) **Random access protocols**

a) Fixed assignment protocols

- In this class of protocols, available resources are divided between the nodes such that the resource assignment is long term and each node can use its resources exclusively without the risk of collisions.
- Long term means that the assignment is for durations of minutes, hours or even longer as opposed to the short term case where assignments have a scope of a data burst, corresponding to a time horizon of perhaps milliseconds.
- Typical protocols of this class are TDMA, FDMA, CDMA and SDMA
    1. The Time Division Multiple Access(TDMA) scheme subdivides the time axis into fixed length super frames and each super frame is again subdivided into a fixed number of time slots. These time slots are assigned to nodes exclusively and hence the node can transmit in this time slot periodically in every super frame.TDMA requires tight time synchronization between nodes to avoid overlapping of signals in adjacent time slots.
    2. In Frequency Division Multiple Access(FDMA), The available frequency band is subdivided into a number of subchannels and these are assigned to nodes,

which can transmit exclusively on their channel. This scheme requires frequency synchronizations to renegotiate the assignment of resources to nodes.

3. In Code Division Multiple Access(CDMA) schemes, the nodes spread their signals over a much larger bandwidth than needed, using different codes to separate their transmissions. The receiver has to know the code used by the transmitter all parallel transmissions using other codes appear as noise.
4. In Space Division Multiple Access(SDMA), The spatial separation of nodes is used to separate their transmissions. SDMA requires arrays of antennas and sophisticated signal processing techniques and cannot be considered a candidate technology for WSNs

b) Demand assignment protocols

In demand assignment protocols, the exclusive allocation of resources to nodes is made on a short-term basis, typically the duration of a data burst. This class of protocols can be broadly subdivided into centralized and distributed protocols. In central control protocols (examples are the HIPERLAN/2 protocol, DQRUMA, or the MASCARA protocol; polling schemes can also be subsumed under this class), the nodes send out requests for bandwidth allocation to a central node that either accepts or rejects the requests. In case of successful allocation, a confirmation is transmitted back to the requesting node along with a description of the allocated resource, for example, the numbers and positions of assigned time slots in a TDMA system and the duration of allocation. The node can use these resources exclusively. The submission of requests from nodes to the central station is often done contention based, that is, using a random access protocol on a dedicated (logical) signalling channel. Another option is to let the central station poll its associated nodes. In addition, the nodes often piggyback requests onto data packets transmitted in their exclusive data slots, thus avoiding transmission of separate request packets. The central node needs to be switched on all the time and is responsible for resource allocation. Resource deallocation is often done implicitly: when a node does not use its time slots any more, the central node can allocate these to other nodes. This way, nodes do not need to send extra deallocation packets. Summarizing, the central node performs a lot of activities, it must be constantly awake, and thus needs lots of energy. This class of protocols is a good choice if a sufficient number of energy-unconstrained nodes are present and the duties of the central station can be moved to these.

An example of distributed demand assignment protocols are token-passing protocols like IEEE 802.4 Token Bus. The right to initiate transmissions is tied to reception of a small special token frame. The token frame is rotated among nodes organized in a logical ring on top of a broadcast medium. Special ring management procedures are needed to include and exclude nodes from the ring or to correct failures like lost tokens. Token-passing protocols have also been considered for wireless or error-prone media, but they tend to have problems with the maintenance of the logical ring in the presence of significant channel errors. In addition, since token circulation times are variable, a node must always be able to receive the token to avoid breaking the logical ring. Hence, a nodes transceiver must be switched on most of the time. In addition, maintaining a logical ring in face of frequent topology changes is not an easy task and involves significant signalling traffic besides the token frames themselves.

c) Random access protocols

The nodes are uncoordinated, and the protocols operate in a fully distributed manner. Random access protocols often incorporate a random element, for example, by exploiting random packet arrival times, setting timers to random values, and so on. One of the first and still very important random access protocols is the ALOHA or slotted ALOHA protocol, developed at the University of Hawaii. In the pure ALOHA protocol, a node wanting to transmit a new packet transmits it immediately. There is no coordination with other nodes and the protocol thus accepts the risk of collisions at the receiver. To detect this, the receiver is required to send an immediate acknowledgment for a properly received packet. The transmitter interprets the lack of an acknowledgment frame as a sign of a collision, backs off for a random time, and starts the next trial. ALOHA provides short access and transmission delays under light loads; under heavier loads, the number of collisions increases, which in turn decreases the throughput efficiency and increases the transmission delays. In slotted ALOHA, the time is subdivided into time slots and a node is allowed to start a packet transmission only at the beginning of a slot. A slot is large enough to accommodate a maximum-length packet. Accordingly, only contenders starting their packet transmission in the same slot can destroy a node's packet. If any node wants to start later, it has to wait for the beginning of the next time slot and has thus no chance to destroy the node's packet. In short, the synchronization reduces the probability of collisions and slotted ALOHA has a higher throughput than pure ALOHA.

In the class of CSMA protocols, a transmitting node tries to be respectful to ongoing transmissions. First, the node is required to listen to the medium; this is called carrier sensing. If the medium is

found to be idle, the node starts transmission. If the medium is found busy, the node defers its transmission for an amount of time determined by one of several possible algorithms. For example, in nonpersistent CSMA, the node draws a random waiting time, after which the medium is sensed again. Before this time, the node does not care about the state of the medium. In different persistent CSMA variants, after sensing that the medium is busy, the node awaits the end of the ongoing transmission and then behaves according to a backoff algorithm. In many of these backoff algorithms, the time after the end of the previous frame is subdivided into time slots. In p-persistent CSMA, a node starts transmission in a time slot with some probability p and with probability 1 – p it waits for another slot.3 If some other node starts to transmit in the meantime, the node defers and repeats the whole procedure after the end of the new frame. A small value of p makes collisions unlikely, but at the cost of high access delays. The converse is true for a large value of p.

In the backoff algorithm executed by the IEEE 802.11 Distributed Coordination Function (DCF), a node transmitting a new frame picks a random value from the current contention window and starts a timer with this value. The timer is decremented after each slot. If another node starts in the meantime, the timer is suspended and resumed after the next frame ends and contention continues. If the timer decrements to zero, the node transmits its frame. When a transmission error occurs (indicated, for example, by a missing acknowledgment frame), the size of the contention window is increased according to a modified binary exponential backoff procedure.4 While CSMA protocols are still susceptible to collisions, they have a higher throughput efficiency than ALOHA protocols, since ongoing packets are not destroyed when potential competitors hear them on the medium.

As explained above, carrier-sense protocols are susceptible to the hidden-terminal problem since interference at the receiver cannot be detected by the transmitter. This problem may cause packet collisions. The energy spent on collided packets is wasted and the packets have to be retransmitted. Several approaches have appeared to solve or at least to reduce the hidden-terminal problem; we present two important ones: the busy-tone solution and the RTS/CTS handshake.

In the original busy-tone solution, two different frequency channels are used, one for the data packets and the other one as a control channel. As soon as a node starts to receive a packet destined to it, it emits an unmodulated wave on the control channel and ends this when packet reception is finished. A node that wishes to transmit a packet first senses the control channel for the presence of a busy tone. If it hears something, the node backs off according to some algorithm, for example similar to nonpersistent CSMA. If it hears nothing, the node starts packet transmission on the data

channel. This protocol solves both the hidden- and exposed terminal problem, given that the busy-tone signal can be heard over the same distance as the data signal. If the busy tone is too weak, a node within radio range of the receiver might start data transmission and destroy the receiver's signal. If the busy tone is too strong, more nodes than necessary suppress their transmissions. The control channel does not need much bandwidth but a narrow bandwidth channel requires good frequency synchronization. A solution with two busy tones, one sent by the receiver and the other by the transmitter node. Another variant of the busy-tone approach is used by PAMAS.

The RTS/CTS handshake as used in IEEE 802.11 is based on the MACAW protocol and is illustrated in Figure 2.2. It uses only a single channel and two special control packets. Suppose that node B wants to transmit a data packet to node C. After B has obtained channel access (for example after sensing the channel as idle), it sends a Request. To Send (RTS) packet to C, which includes a duration field indicating the remaining length of the overall transaction (i.e., until the point where B would receive the acknowledgment for its data packet). If C has properly received the RTS packet, it sends a Clear. To Send (CTS) packet, which again contains a duration field. When B receives the CTS packet, it starts transmission of the data packet and finally C answers with an acknowledgment packet. The acknowledgment is used to tell B about the success of the transmission; lack of acknowledgment is interpreted as collision (the older MACA protocol lacks the acknowledgment). Any other station A or D hearing either the RTS, CTS, data or acknowledgment packet sets an internal timer called Network Allocation Vector (NAV) to the remaining duration indicated in the respective frame and avoids sending any packet as long as this timer is not expired. Specifically, nodes A and D send no CTS answer packets even when they have received a RTS packet correctly. This way, the ongoing transmission is not distorted.

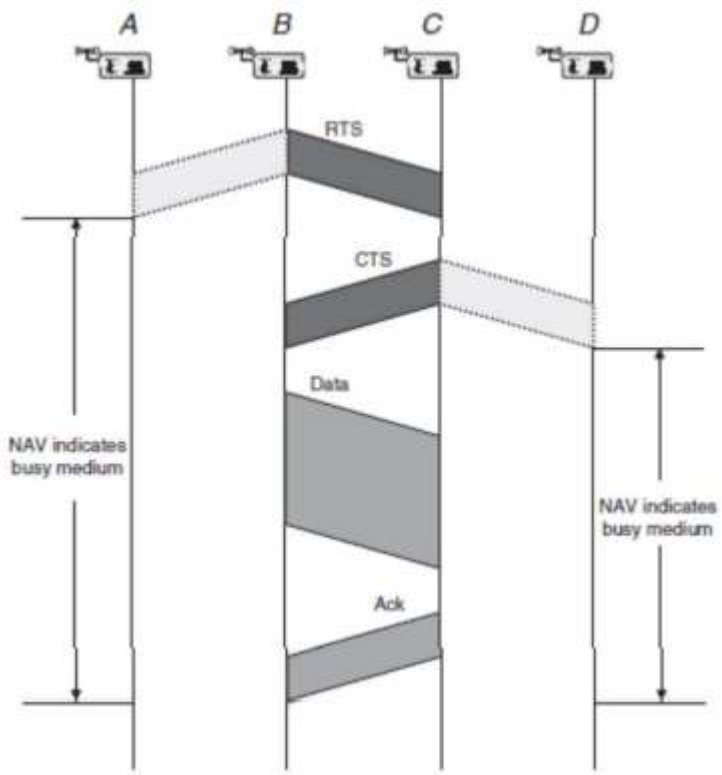


Fig 2.2 RTS/CTS handshake in IEEE 802.11

Does this scheme eliminate collisions completely? No, there still exist some collision scenarios. First, in the scenario described above, nodes A and C can issue RTS packets to B simultaneously. However, in this case, only the RTS packets are lost and no long data frame has been transmitted. Two further problems are illustrated in Figure 3: In the left part of the figure, nodes A and B run the RTS-CTS-Data-Ack sequence, and B's CTS packet also reaches node C. However, at almost the same time, node D sends an RTS packet to C, which collides at node C with B's CTS packet. This way, C has no chance to decode the duration field of the CTS packet and to set its NAV variable accordingly. After its failed RTS packet, D sends the RTS packet again to C and C answers with a CTS packet. Node C is doing so because it cannot hear A's ongoing transmission and has no proper NAV entry. C's CTS packet and A's data packet collide at B. In the figure's right part, the problem is created by C starting its RTS packet to D immediately before it can sense B's CTS packet, which C consequently cannot decode properly. One solution approach is to ensure that CTS packets are longer than RTS packets. For an explanation, consider the right part of Figure 3. Here, even if B's CTS arrives at C immediately after C starts its RTS, it lasts long enough that C has a chance to turn its transceiver into receive mode and to sense B's signal. An additional protocol rule

states that in such a case node C has to defer any further transmission for a sufficiently long time to accommodate one maximum-length data packet. Hence, the data packet between A and B can be transmitted without distortion.

A further problem of the RTS/CTS handshake is its significant overhead of two control packets per data packet, not counting the acknowledgment packet. If the data packet is small, this overhead might not pay off and it may be simpler to use some plain CSMA variant.

For long packets, the overhead of the RTS/CTS handshake can be neglected, but long packets are more likely to be hit by channel errors and must be retransmitted entirely, wasting precious energy (channel errors often hit only a few bits).

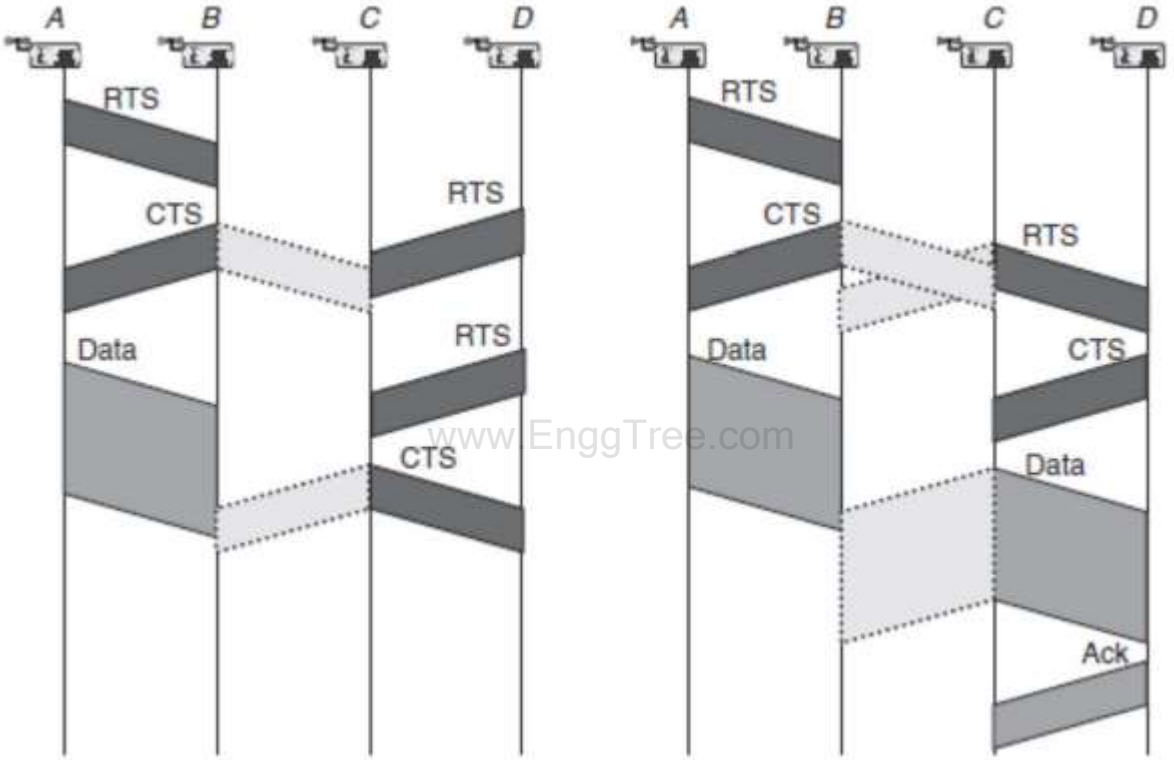


Fig: Two problems in RTS/CTS handshake

2.2 Low duty cycle protocols and wakeup concepts

- Low duty cycle protocols try to avoid spending time in the idle state and to reduce the communication activities of a sensor node to a minimum.
- In an ideal case, the sleep state is left only when a node is about to transmit or receive packets.
- A concept for achieving this is called wakeup radio.
- In several protocols, a periodic wakeup scheme is used. Such schemes exist in different flavors. One is the cycled receiver approach is illustrated in below Figure

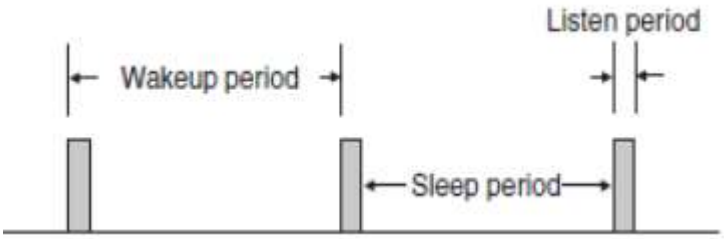


Fig 2.2.1 Periodic Wakeup Scheme

- In this approach, nodes spend most of their time in the sleep mode and wake up periodically to receive packets from other nodes.
- Specifically, a node A listens onto the channel during its listen period and goes back into sleep mode when no other node takes the opportunity to direct a packet to A.
- A potential transmitter B must acquire knowledge about A's listen periods to send its packet at the right time – this task corresponds to a rendezvous.
- This rendezvous can be accomplished by letting node A transmit a short beacon at the beginning of its listen period to indicate its willingness to receive packets.
- Another method is to let node B send frequent request packets until one of them hits A's listen period and is really answered by A.
- However, in either case, node A only receives packets during its listen period.
- If node A itself wants to transmit packets, it must acquire the target's listen period.
- A whole cycle consisting of sleep period and listen period is also called a wakeup period.
- The ratio of the listen period length to the wakeup period length is also called the node's duty cycle.
- By choosing a small duty cycle, the transceiver is in sleep mode most of the time, avoiding idle listening and conserving energy.

- By choosing a small duty cycle, the traffic directed from neighboring nodes to a given node concentrates on a small time window (the listen period) and in heavy load situations significant competition can occur.
- Choosing a long sleep period induces significant per-hop latency. In the multihop case, the per-hop latencies add up and create significant end-to-end latencies.
- Sleep phases should not be too short lest the start-up costs outweigh the benefits.
- In other protocols like S-MAC, there is also a periodic wakeup but nodes can both transmit and receive during their wakeup phases.
- When nodes have their wakeup phases at the same time, there is no necessity for a node wanting to transmit a packet to be awake outside these phases to rendezvous its receiver.

S-MAC

- The S-MAC (Sensor-MAC) protocol provides mechanisms to circumvent idle listening, collisions, and overhearing.
- S-MAC adopts a periodic wakeup scheme, that is, each node alternates between a fixedlength listen period and a fixed-length sleep period according to its schedule.
- The listen period of S-MAC can be used to receive and transmit packets.
- S-MAC attempts to coordinate the schedules of neighboring nodes such that their listen periods start at the same time.

Phases in listen period:

- A node x's listen period is subdivided into three different phases:

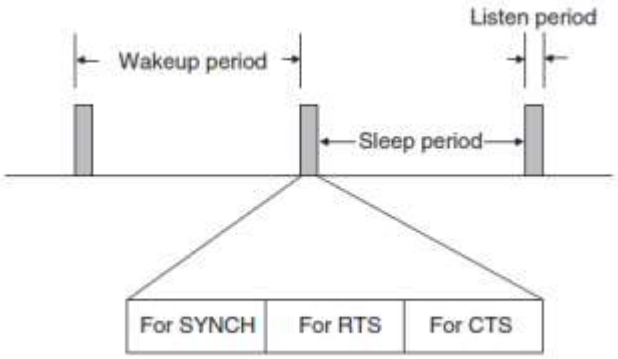


Fig 2.2.1 S-MAC Principle

First phase

- In the first phase (SYNCH phase), node x accepts SYNCH packets from its neighbors.

- In these packets, the neighbors describe their own schedule and x stores their schedule in a table (the schedule table).
- Node x's SYNCH phase is subdivided into time slots and x's neighbors contend according to a CSMA scheme with additional backoff.
- Each neighbor y wishing to transmit a SYNCH packet picks one of the time slots randomly and starts to transmit if no signal was received in any of the previous slots.
- In the other case, y goes back into sleep mode and waits for x's next wakeup. ϖ In the other direction, since x knows a neighbor y's schedule, x can wake at appropriate times and send its own SYNCH packet to y (in broadcast mode).
- It is not required that x broadcasts its schedule in every of y's wakeup periods.
- However, for reasons of time synchronization and to allow new nodes to learn their local network topology, x should send SYNCH packets periodically. The according period is called synchronization period.

Second phase

- In the second phase (RTS phase), x listens for RTS packets from neighboring nodes.
- In S-MAC, the RTS/CTS handshake is used to reduce collisions of data packets due to hidden-terminal situations.

- Again, interested neighbors contend in this phase according to a CSMA scheme with additional backoff.

Third Phase

- In the third phase (CTS phase), node x transmits a CTS packet if an RTS packet was received in the previous phase. After this, the packet exchange continues, extending into x's nominal sleep time. Working of S-MAC Protocol
- When competing for the medium, the nodes use the RTS/CTS handshake, including the virtual carrier-sense mechanism.
- When transmitting in a broadcast mode (for example SYNCH packets), the RTS and CTS packets are dropped and the nodes use CSMA with backoff.

Working of S-MAC Protocol

- When competing for the medium, the nodes use the RTS/CTS handshake, including the virtual carrier-sense mechanism.

- When transmitting in a broadcast mode (for example SYNCH packets), the RTS and CTS packets are dropped and the nodes use CSMA with backoff.
- If we can arrange that the schedules of node x and its neighbors are synchronized, node x and all its neighbors wake up at the same time and x can reach all of them with a single SYNCH packet.
- The S-MAC protocol allows neighboring nodes to agree on the same schedule and to create virtual clusters.
- The clustering structure refers solely to the exchange of schedules; the transfer of data packets is not influenced by virtual clustering.
- The S-MAC protocol proceeds as follows to form the virtual clusters:
  - ✓ A node x, newly switched on, listens for a time of at least the synchronization period.
  - ✓ If x receives any SYNCH packet from a neighbor, it adopts the announced schedule and broadcasts it in one of the neighbors' next listen periods.
  - ✓ In the other case, node x picks a schedule and broadcasts it.
  - ✓ If x receives another node's schedule during the broadcast packet's contention period, it drops its own schedule and follows the other one.
  - ✓ It might also happen that a node x receives a different schedule after it already has chosen one, for example, because bit errors destroyed previous SYNCH packets.
  - ✓ If node x already knows about the existence of neighbors who adopted its own schedule, it keeps its schedule and in the future has to transmit its SYNCH and data packets according to both schedules.
  - ✓ On the other hand, if x has no neighbor sharing its schedule, it drops its own and adopts the other one.
  - ✓ Since there is always a chance to receive SYNCH packets in error, node x periodically listens for a whole synchronization period

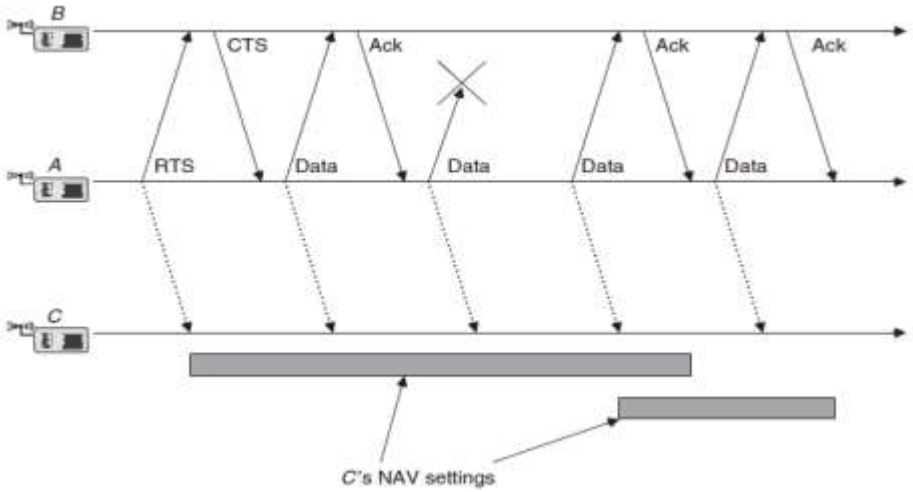


Fig:2.2.1 S-MAC Fragmentation and NAV Settings

S-MAC includes a fragmentation scheme

- ✓ A series of fragments is transmitted with only one RTS/CTS exchange between the transmitting node A and receiving node B.
- ✓ After each fragment, B has to answer with an acknowledgment packet.
- ✓ All the packets (data, ack, RTS, CTS) have a duration field and a neighboring node C is required to set its NAV field accordingly.

- ✓ In S-MAC, the duration field of all packets carries the remaining length of the whole transaction, including all fragments and their acknowledgments. Therefore, the whole message shall be passed at once.
- ✓ If one fragment needs to be retransmitted, the remaining duration is incremented by the length of a data plus ack packet, and the medium is reserved for this prolonged time.
- ✓ However, there is the problem of how a nonparticipating node shall learn about the elongation of the transaction when he has only heard the initial RTS or CTS packets.

Drawbacks:

- It is hard to adapt the length of the wakeup period to changing load situations, since this length is essentially fixed, as is the length of the listen period.

Wakeup radio concepts

- If a node were always in the receiving state when a packet is transmitted to it, in the transmitting state when it transmits a packet, and in the sleep state at all other times; the idle state should be avoided

- The wakeup radio concept strives to achieve this goal by a simple, “powerless” receiver that can trigger a main receiver if necessary.
- One proposed wakeup MAC protocol assumes the presence of several parallel data channels, separated either in frequency (FDMA) or by choosing different codes in a CDMA schemes.
- A node wishing to transmit a data packet randomly picks one of the channels and performs a carrier sensing operation.
- If the channel is busy, the node makes another random channel choice and repeats the carrier-sensing operation.
- After a certain number of unsuccessful trials, the node backs off for a random time and starts again.
- If the channel is idle, the node sends a wakeup signal to the intended receiver, indicating both the receiver identification and the channel to use.
- The receiver wakes up its data transceiver, tunes to the indicated channel, and the data packet transmission can proceed. Afterward, the receiver can switch its data transceiver back into sleep mode.
- It has the significant advantage that only the low-power wakeup transceiver has to be switched on all the time while the much more energy consuming data transceiver is nonsleeping if and only if the node is involved in data transmissions.
- Furthermore, this scheme is naturally traffic adaptive, that is, the MAC becomes more and more active as the traffic load increases.

- Periodic wakeup schemes do not have this property. However, there are also some drawbacks.
- First, there is no real hardware yet for such an ultralow power wakeup transceiver.
- Second, the range of the wakeup radio and the data radio should be the same.
- If the range of the wakeup radio is smaller than the range of the data radio, possibly not all neighbor nodes can be woken up.
- On the other hand, if the range of the wakeup radio is significantly larger, there can be a problem with local addressing schemes.
- These schemes do not use globally or network wide-unique addresses but only locally unique addresses, such that no node has two or more one-hop neighbors with the same address.
- Since the packets exchanged in the neighbor discovery phase have to use the data channel, the two hop neighborhood as seen on the data channel might be different from the two-hop neighborhood on the wakeup channel.

- Third, this scheme critically relies on the wakeup channel's ability to transport useful information like node addresses and channel identifications;
- This might not always be feasible for transceiver complexity reasons and additionally requires methods to handle collisions or transmission errors on the wakeup channel.
- If the wakeup channel does not support this feature, the transmitter wakes up all its neighbors when it emits a wakeup signal, creating an overhearing situation for most of them.
- If the transmitting node is about to transmit a long data packet, it might be worthwhile to prepend the data packet with a short filter packet announcing the receiving node's address.
- All the other nodes can go back to sleep mode after receiving the filter packet. Instead of using an extra packet, all nodes can read the bits of the data packet until the destination address appeared.
- If the packet's address is not identical to its own address, the node can go back into sleep mode.

Contention-based protocols

- In a given transmit opportunity toward a receiver node can in principle be taken by any of its neighbors.
- If only one neighbor tries its luck, the packet goes through the channel.
- If two or more neighbors try their luck, these have to compete with each other and in unlucky cases due to hidden-terminal situations, a collision might occur, wasting energy for both transmitter and receiver.

PAMAS

- The PAMAS protocol (Power Aware Multiaccess with Signaling) originally designed for ad hoc networks.
- It provides a detailed overhearing avoidance mechanism while it does not consider the idle listening problem.
- The protocol combines the busy-tone solution and RTS/CTS handshake similar to the MACA protocol

Features of PAMAS:

- It uses two channels: a data channel and a control channel.
- All the signaling packets (RTS, CTS, busy tones) are transmitted on the control channel, while the data channel is reserved for data packets.

Protocol operation of PAMAS:

- Let us consider an idle node x to which a new packet destined to a neighboring node y arrives.
- First, x sends an RTS packet on the control channel without doing any carrier sensing. This packet carries both x's and y's MAC addresses.
- If y receives this packet, it answers with a CTS packet if y does not know of any ongoing transmission in its vicinity.
- Upon receiving the CTS, x starts to transmit the packet to y on the data channel. When y starts to receive the data, it sends out a busy-tone packet on the control channel.
- If x fails to receive a CTS packet within some time window, it enters the backoff mode, where a binary exponential backoff scheme is used.
- The backoff time is uniformly chosen from a time interval that is doubled after each failure to receive a CTS.

- Now, let us look at the nodes receiving x's RTS packet on the control channel. There is the intended receiver y and there are other nodes; let z be one of them.
- If z is currently receiving a packet, it reacts by sending a busy-tone packet, which overlaps with y's CTS at node x and effectively destroys the CTS.
- Therefore, x cannot start transmission and z's packet reception is not disturbed. Since the busy-tone packet is longer than the CTS, we can be sure that the CTS is really destroyed. Next, we consider the intended receiver y. If y knows about an ongoing transmission in its vicinity, it suppresses its CTS, causing x to back off.
- Node y can obtain this knowledge by either sensing the data channel or by checking whether there was some noise on the control channel immediately after receiving the RTS.
- This noise can be an RTS or CTS of another node colliding at y.
- In the other case, y answers with a CTS packet and starts to send out a busy-tone packet as soon as x's transmission has started.
- Furthermore, y sends out busy-tone packets each time it receives some noise or a valid packet on the control channel, to prevent its neighborhood from any activities.

Schedule-based protocols

- Schedule-based protocols that do not explicitly address idle listening avoidance but do so implicitly, for example, by employing TDMA schemes, which explicitly assign transmission and reception opportunities to nodes and let them sleep at all other times.
- In schedule-based protocols is that transmission schedules can be computed such that no collisions occur at receivers and hence no special mechanisms are needed to avoid hidden-terminal situations.

Disadvantages:

- First, the setup and maintenance of schedules involves signaling traffic, especially when faced to variable topologies.
- Second, if a TDMA variant is employed, time is divided into comparably small slots, and both transmitter and receiver have to agree to slot boundaries to actually meet each other and to avoid overlaps with other slots, which would lead to collisions.
- However, maintaining time synchronization involves some extra signaling traffic.

- Third drawback is that such schedules are not easily adapted to different load situations on small timescales. Specifically, in TDMA, it is difficult for a node to give up unused time slots to its neighbors.

- Fourth drawback is that the schedule of a node may require a significant amount of memory, which is a scarce resource in several sensor node designs.

- Finally, distributed assignment of conflict-free TDMA schedules is a difficult problem in itself.

LEACH

The LEACH protocol (Low-energy Adaptive Clustering Hierarchy) assumes a dense sensor network of homogeneous, energy-constrained nodes, which shall report their data to a sink node.

- In LEACH, a TDMA based MAC protocol is integrated with clustering and a simple "routing" protocol.
- LEACH partitions the nodes into **clusters** and in each cluster a dedicated node, the **clusterhead**, is responsible for creating and maintaining a TDMA schedule; all the other nodes of a cluster are **member nodes**.
- To all member nodes, TDMA slots are assigned, which can be used to exchange data between the member and the clusterhead; there is no peer-to-peer communication. With the exception of their time slots, the members can spend their time in sleep state.
- The clusterhead aggregates the data of its members and transmits it to the sink node or to other nodes for further relaying.
- Since the sink is often far away, the clusterhead must spend significant energy for this transmission.
- For a member, it is typically much cheaper to reach the clusterhead than to transmit directly to the sink.
- The clusterheads role is energy consuming since it is always switched on and is responsible for the long-range transmissions.
- If a fixed node has this role, it would burn its energy quickly, and after it died, all its members would be "headless" and therefore useless.
- Therefore, this burden is rotated among the nodes. Specifically, each node decides independent of other nodes whether it becomes a clusterhead, and therefore there is no signaling traffic related to clusterhead election.

- This decision takes into account when the node served as clusterhead the last time, such that a node that has not been a clusterhead for a long time is more likely to elect itself than a node serving just recently.
- The protocol is round based, that is, all nodes make their decisions whether to become a clusterhead at the same time and the nonclusterhead nodes have to associate to a clusterhead subsequently.
- The nonclusterheads choose their clusterhead based on received signal strengths.
- The network partitioning into clusters is time variable and the protocol assumes global time synchronization.
- After the clusters have been formed, each clusterhead picks a random CDMA code for its cluster, which it broadcasts and which its member nodes have to use subsequently.
- This avoids a situation where a border node belonging to clusterhead *A* distorts transmissions directed to clusterhead *B*.

Fig 2.3.1 Intercluster Interferences

Stages of LEACH protocol:

- The protocol is organized in **rounds** and each round is subdivided into a setup phase and a steady-state phase.

Setup Phase:

- ✓ The **setup phase** starts with the self-election of nodes to clusterheads.
- ✓ In the following **advertisement phase**, the clusterheads inform their neighborhood with an advertisement packet.
- ✓ The clusterheads contend for the medium using a CSMA protocol with no further provision against the hidden-terminal problem.

- ✓ The nonclusterhead nodes pick the advertisement packet with the strongest received signal strength.
- ✓ In the following cluster-setup phase, the members inform their clusterhead ("join"), again using a CSMA protocol.
- ✓ After the cluster setup-phase, the clusterhead knows the number of members and their identifiers.
- ✓ It constructs a TDMA schedule, picks a CDMA code randomly, and broadcasts this information in the broadcast schedule subphase.

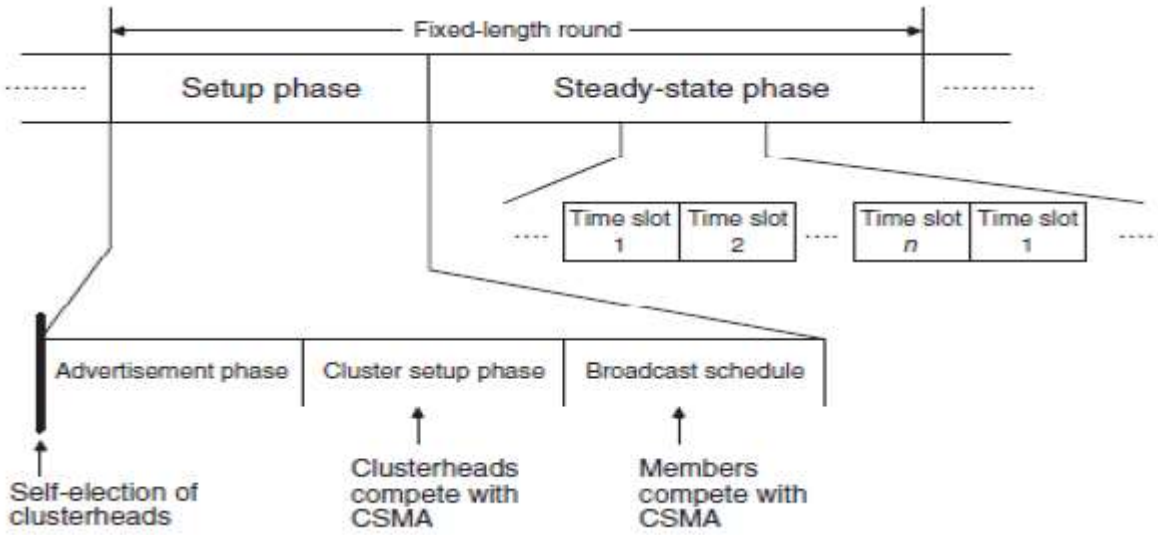


Fig 2.3.2 Organization of Leach round

Steady state phase:

- ❖ After this, the TDMA steady-state phase begins. Because of collisions of advertisement or join packets, the protocol cannot guarantee that each non clusterhead node belongs to a cluster.
- ❖ However, it can guarantee that nodes belong to at most one cluster.
- ❖ The clusterhead is switched on during the whole round and the member nodes have to be switched on during the setup phase and occasionally in the steady-state phase, according to their position in the cluster's TDMA schedule.

Drawback:

- ➢ unable to cover large geographical areas because a clusterhead two miles away from the sink likely does not have enough energy to reach the sink at all, not to mention achieving a low BER.

Solution:

- ➢ If it can be arranged that a clusterhead can use other clusterheads for forwarding, this limitation can be mitigated.

3.1 **Applications of 6LoWPAN**

The reason why there are such a large number of technical solutions in the wireless embedded networking market is that the requirements, scale and market of embedded applications vary wildly. Applications can range from personal health sensor monitoring to large scale facility monitoring, which differ greatly. This is in contrast to PC information technology, which is fairly homogeneous and mainly aimed at home and office environments. The ideal use of 6LoWPAN is in applications where:

- embedded devices need to communicate with Internet-based services,
- low-power heterogeneous networks need to be tied together,
- the network needs to be open, reusable and evolvable for new uses and services
- scalability is needed across large network infrastructures with mobility.

Connecting the Internet to the physical world enables a wide range of interesting applications where 6LoWPAN technology may be applicable, for example:

1. home and building automation
2. healthcare automation and logistics

3.2 The 6LoWPAN Architecture

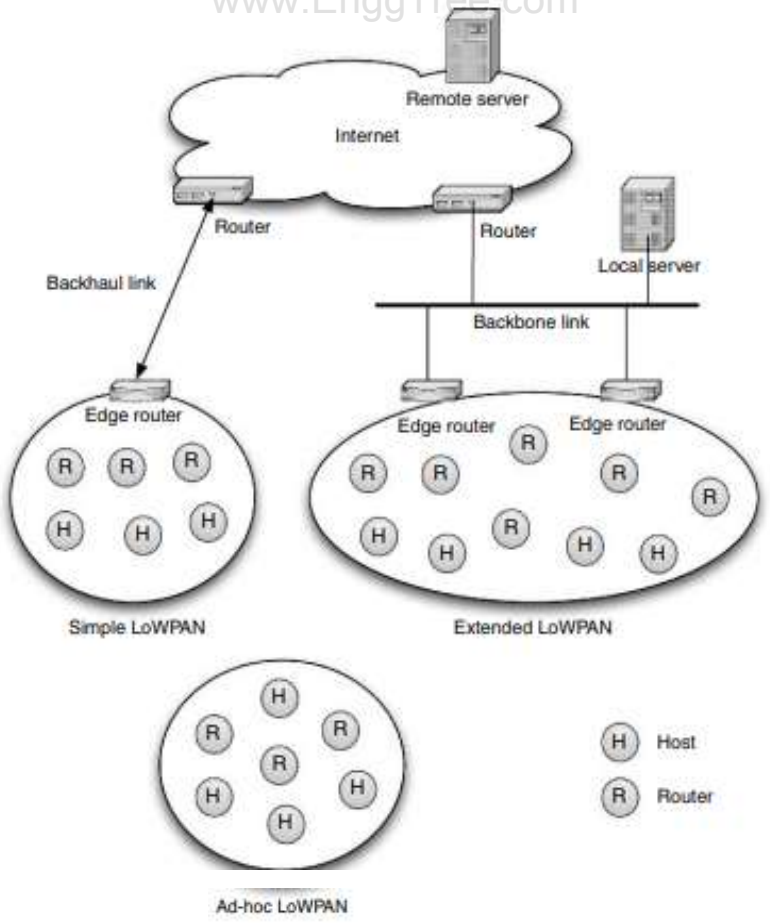


Fig 3.1 The 6LoWPAN architecture.

- The 6LoWPAN architecture is made up of low-power wireless area networks (LoWPANs)2, which are IPv6 stub networks. The overall 6LoWPAN architecture is presented in Figure 3.1. Three different kinds of LoWPANs have been defined:
  - Simple LoWPANs
  - Extended LoWPANs
  - Ad hoc LoWPANs.
- A LoWPAN is the collection of 6LoWPAN Nodes which share a common IPv6 address prefix (the first 64 bits of an IPv6 address), meaning that regardless of where a node is in a LoWPAN its IPv6 address remains the same.
- An Ad hoc LoWPAN is not connected to the Internet, but instead operates without an infrastructure. A Simple LoWPAN is connected through one LoWPAN Edge Router to another IP network. A backhaul link (point-to-point, e.g. GPRS) is shown in the figure, but this could also be a backbone link (shared). An Extended LoWPAN encompasses the LoWPANs of multiple edge routers along with a backbone link (e.g. Ethernet) interconnecting them.
- LoWPANs are connected to other IP networks through edge routers. The edge router plays an important role as it routes traffic in and out of the LoWPAN, while handling 6LoWPAN compression and Neighbor Discovery for the LoWPAN. If the LoWPAN is to be connected to an IPv4 network, the edge router will also handle IPv4 interconnectivity. Edge routers typically have management features tied into overall IT management solutions. Multiple edge routers can be supported in the same LoWPAN if they share a common backbone link.
- A LoWPAN consists of nodes, which may play the role of host or router, along with one or more edge routers. The network interfaces of the nodes in a LoWPAN share the same IPv6 prefix which is distributed by the edge router and routers throughout the LoWPAN. In order to facilitate efficient network operation, nodes register with an edge router. These operations are part of Neighbor Discovery (ND), which is an important basic mechanism
- of IPv6. Neighbor Discovery defines how hosts and routers interact with each other on the same link. LoWPAN Nodes may participate in more than one LoWPAN at the same time (called multi-homing), and fault tolerance can be achieved between edge routers. LoWPAN Nodes are free to move throughout the LoWPAN, between edge routers, and even between LoWPANs. Topology change may also be caused by wireless channel conditions, without physical movement. A multihop mesh topology within the LoWPAN is achieved either through link-layer forwarding (called Mesh-Under) or using IP routing (called Route-Over). Both techniques are supported by 6LoWPAN.

- Communication between LoWPAN Nodes and IP nodes in other networks happens in an end-to-end manner, just as between any normal IP nodes. Each LoWPAN Node is identified. by a unique IPv6 address, and is capable of sending and receiving IPv6 packets. Typically LoWPAN Nodes support ICMPv6 traffic such as “ping”, and use the user datagram protocol (UDP) as a transport. the Simple LoWPAN and Extended LoWPAN Nodes can communicate with either of the servers through their edge router. As the payload and processing capabilities of LoWPAN Nodes are extremely limited, application protocols are usually designed using a simple binary format in a UDP payload.
- The main difference between a Simple LoWPAN and an Extended LoWPAN is the existence of multiple edge routers in the LoWPAN, which share the same IPv6 prefix and a common backbone link. Multiple LoWPANs can overlap each other (even on the same channel). When moving from one LoWPAN to another, a node’s IPv6 address will change. A LoWPAN Edge Router is typically connected to the Internet over a backhaul link such as cellular or DSL [ID-6lowpan-nd]. A network deployment may also choose to use multiple Simple LoWPANs rather than an Extended LoWPAN on a shared backbone link, e.g. for management reasons. This is not a problem if there is low mobility between LoWPANs in the network, or the application does not assume stable IPv6 addresses for nodes.
- In an Extended LoWPAN configuration, multiple edge routers share a common backbone link and collaborate by sharing the same IPv6 prefix, offloading most Neighbor Discovery messaging to the backbone link [ID-6lowpan-nd]. This greatly simplifies LoWPAN Node operation as IPv6 addresses are stable throughout the Extended LoWPAN and movement between edge routers is very simple. Edge routers also handle IPv6 forwarding on behalf of the nodes. To IP nodes outside the LoWPAN, the LoWPAN Nodes are always reachable regardless of their attachment point in the Extended LoWPAN. This enables large enterprise 6LoWPAN infrastructures to be built, functioning similar to a WLAN (WiFi) access point infrastructure (but at layer 3 instead of layer 2).
- 6LoWPAN does not require an infrastructure to operate, but may also operate as an Ad hoc LoWPAN [ID-6lowpan-nd]. In this topology, one router must be configured to act as a simplified edge router, implementing two basic functionalities: unique local unicast address (ULA) generation [RFC4193] and handling 6LoWPAN Neighbor Discovery registration functionality. From the LoWPAN Node point of view the network operates just like a Simple LoWPAN, except the prefix advertised is an IPv6 local prefix rather than a global one, and there are no routes outside the LoWPAN.

The protocol stack

- A simple IPv6 protocol stack with 6LoWPAN (also called a 6LoWPAN protocol stack) is almost identical to a normal IP stack with the following differences.
- 6LoWPAN only supports IPv6, for which a small adaptation layer (called the LoWPAN adaptation layer) has been defined to optimize IPv6 over IEEE 802.15.4

- 6LoWPAN stack implementations in embedded devices often implement the LoWPAN adaptation layer together with IPv6, thus they can alternatively be shown together as part of the network layer.
- The most common transport protocol used with 6LoWPAN is the user datagram protocol (UDP), which can also be compressed using the LoWPAN format.
- The transmission control protocol (TCP) is not commonly used with 6LoWPAN for performance, efficiency and complexity reasons. The Internet control message protocol v6 (ICMPv6) is used for control messaging, for example ICMP echo, ICMP destination unreachable and Neighbor Discovery messages.
- Application protocols are often application specific and in binary format, although more standard application protocols are becoming available.
- Adaptation between full IPv6 and the LoWPAN format is performed by routers at the edge of 6LoWPAN islands, referred to as edge routers. This transformation is transparent, efficient and stateless in both directions. LoWPAN adaptation in an edge router typically is performed as part of the 6LoWPAN network interface driver and is usually transparent to the IPv6 protocol stack itself.

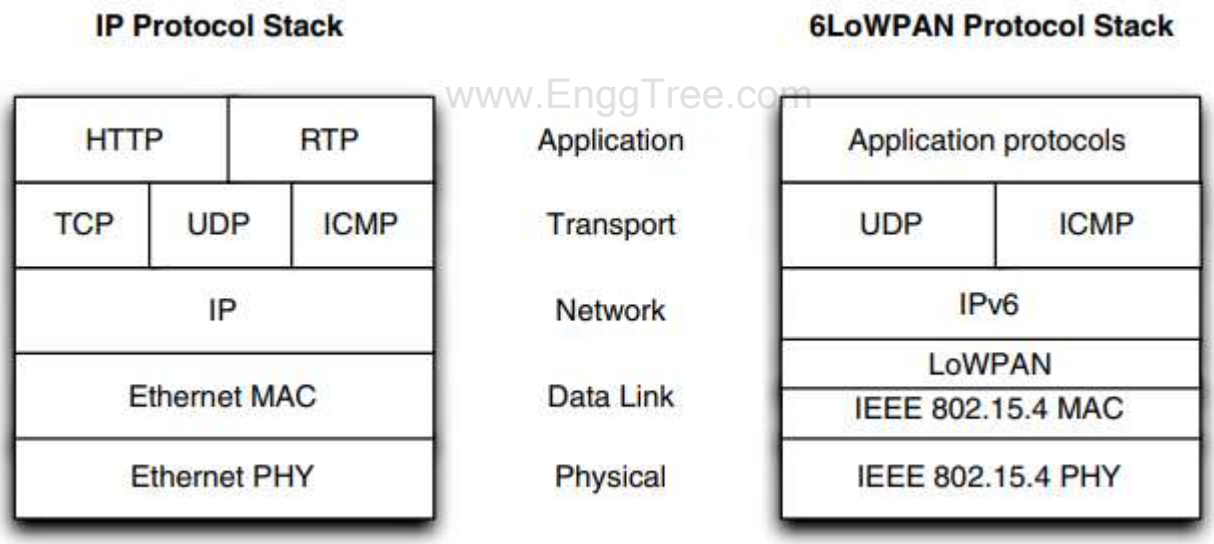


Fig 3.2 IP and 6loWPAN protocol stacks

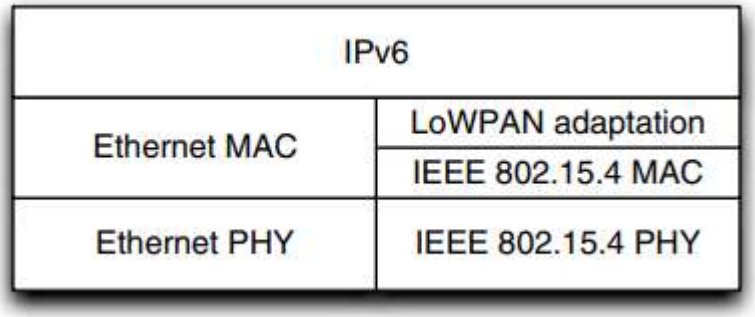


Fig: 3.3 IPv6 edge router with 6LoWPAN support.

- Figure 3.3 illustrates one realization of an edge router with 6LoWPAN support. See Section 6.4 for edge router implementation considerations. Inside the LoWPAN, hosts and routers do not actually need to work with full IPv6 or UDP header formats at any point as all compressed fields are implicitly known by each node.

Link layers for 6LoWPAN

- The most basic requirements for a link layer to support 6LoWPAN are framing, unicast transmission and addressing.
- Addressing is required to differentiate between nodes on a link, and to form IPv6 addresses which are then elided by 6LoWPAN compression.
- It is highly recommended that a link supports unique addresses by default (e.g. a 64-bit extended unique identifier [EUI-64]), to allow for stateless autoconfiguration.
- Multi-access links should provide a broadcast service. Multicast service is required by standard IPv6, but not by 6LoWPAN (broadcast is sufficient). IPv6 requires a maximum transmission unit (MTU) of 1280 bytes from a link, which 6LoWPAN fulfills by supporting fragmentation at the LoWPAN adaptation layer.
- A link should provide payload sizes at least 30 bytes in length to be useful (and preferably larger than 60 bytes). Although UDP and ICMP include a simple 16-bit checksum, it is recommended that the link layer also provides strong error checking.
- Finally, as IPsec may not always be practical for 6LoWPAN, it is highly recommended that links include strong encryption and authentication.

Addressing

- IPv6 addresses are typically formed automatically from the prefix of the LoWPAN and the link-layer address of the wireless interfaces. The difference in a LoWPAN is with the way low-power wireless technologies support link-layer addressing; a direct mapping between the link-layer address and the IPv6 address is used for achieving compression.
- IPv6 addresses are 128 bits in length, and (in the cases relevant here) consist of a 64-bit prefix part and a 64-bit interface identifier (IID).
- 6LoWPAN networks assume that the IID has a direct mapping to the linklayer address, therefore avoiding the need for address resolution. The IPv6 prefix is acquired through Neighbor Discovery Router Advertisement (RA) messages [ID-6lowpan-nd] as on a normal IPv6 link. The construction of

IPv6 addresses in 6LoWPAN from known prefix information and known link-layer addresses, is what allows a high header compression ratio.

Header format

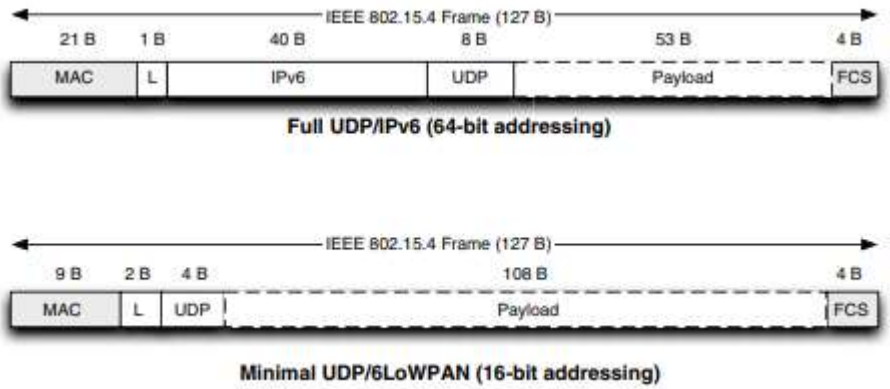


Fig: 3.4 6LoWPAN header compression example (L = LoWPAN header).

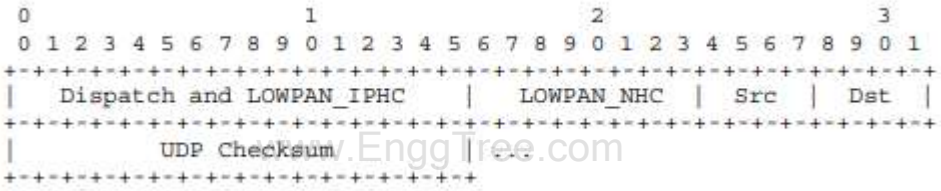


Fig:3.5 6LoWPAN/UDP compressed headers (6 bytes).

- The LoWPAN header consists of a dispatch value identifying the type of header, followed by an IPv6 header compression byte indicating which fields are compressed, and then any in-line IPv6 fields. If, for example, UDP or IPv6 extension headers follow IPv6, then these headers may also be compressed using what is called next-header compression [ID-6lowpan-hc].
- The LoWPAN header consists of a dispatch value identifying the type of header, followed by an IPv6 header compression byte indicating which fields are compressed, and then any in-line IPv6 fields. If, for example, UDP or IPv6 extension headers follow IPv6, then these headers may also be compressed using what is called next-header compression [ID-6lowpan-hc].
- An example of 6LoWPAN compression is given in Figure 3.4. In the upper packet a one-byte LoWPAN dispatch value is included to indicate full IPv6 over IEEE 802.15.4. Figure 3.5 gives an example of 6LoWPAN/UDP in its simplest form (equivalent to the lower packet in Figure 3.4), with a dispatch value and IPv6 header compression (LOWPAN_IPHC) as per [ID-6lowpan-hc] (2 bytes), all IPv6 fields compressed, then followed by a UDP next-header compression byte (LOWPAN_NHC)

with compressed source and destination port fields and the UDP checksum (4 bytes). Therefore in the likely best case the 6LoWPAN/UDP header is just 6 bytes in length.

Mesh topologies

Mesh topologies are common in applications of 6LoWPAN such as automatic meter reading and environmental monitoring. A mesh topology extends the coverage of the network, and reduces the cost of needed infrastructure. In order to achieve a mesh topology, multihop forwarding is required from one node to another. In 6LoWPAN this can be done in three different ways: link-layer mesh, LoWPAN mesh or IP routing. Link-layer mesh and LoWPAN mesh are referred to as Mesh-Under as the mesh forwarding is transparent to the Internet Protocol. IP routing is referred to as Route-Over

Internet integration

When connecting a LoWPAN to another IP network or to the Internet, there are several issues to be considered. 6LoWPAN enables IPv6 for simple embedded devices over low-power wireless networks by efficiently compressing headers and simplifying IPv6 requirements. Issues to be considered when integrating LoWPANs with other IP networks include:

1. **Maximum transmission unit**: In order to comply with the 1280 byte MTU size requirement of IPv6, 6LoWPAN performs fragmentation and reassembly. Applications designed for the Wireless Embedded Internet should however try to minimize packet sizes if possible. This is to avoid forcing a LoWPAN to fragment IPv6 packets, as this incurs a performance penalty
2. **Application protocols:** Application protocols on the Web today depend on payloads of HTML, XML or SOAP carried over HTTP and TCP. This results in payloads ranging in size from hundreds of bytes to several kilobytes. This is far too large for use with 6LoWPAN Nodes. End-to-end application protocols should make use of UDP and compact payload formats (preferably binary) Technologies which are capable of the transparent compression of web services into a format suitable for 6LoWPAN Nodes are especially interesting.
3. **Firewalls and NATs:** In real network deployments firewalls and network address translators (NATs) are a reality. When connecting 6LoWPAN through these there may be several problems that need to be dealt with, for example the blocking of compressed UDP ports and non-standard application protocols used for 6LoWPAN applications, along with the unavailability of static IP addresses.
4. **IPv4 interconnectivity:** 6LoWPAN natively supports only IPv6, however often it will be necessary for 6LoWPAN Nodes to interact with IPv4 nodes or across IPv4 networks. There are several ways to deal with IPv4 interconnectivity, including IPv6-in-IPv4 tunneling and address translation. These mechanisms are typically collocated on LoWPAN Edge Routers, on a local gateway router, or on a node configured for that purpose on the Internet.

5. **Security:** When connecting embedded devices to the public Internet, security should always be a major concern as embedded devices are limited in resources and are autonomous. This is very much so with 6LoWPAN as node and network limitations prevent the use of the full IPsec suite, transport layer ("socket") security or the use of sophisticated firewalls on each node. Although link-layer security inside a LoWPAN (employing the 128-bit AES encryption in IEEE 802.15.4) provides some protection, communication beyond LoWPAN Routers is still vulnerable. This increases the need for end-to-end security at the application layer.

3.2 Adaption layer

An "IP-over-X" adaptation layer needs to map IP datagrams to the services provided by the subnetwork, which is usually considered to be at layer 2 (L2) of a layered reference model.
A number of problems may need to be solved here:

- In a wireless network such as IEEE 802.15.4, a packet may be overheard by multiple receivers, not all of which may need to act on it; the L2 address provides an efficient way to make that decision. Once the IP layer has decided on the IP address of the next hop for a packet, one of the tasks of the adaptation layer is to find out to which link-layer address the packet needs to be addressed to so that it advances on its way to the intended IP-layer destination.
- The subnetwork may not immediately provide a path for packets to proceed to the next IP node. For instance, when mapping IP to connection-oriented networks such as ISDN or ATM (asynchronous transfer mode, a cell-switched link layer based on virtual circuits of 53-byte equal-size cells), the adaptation layer may need to set up connections (and may have to decide when to close them down again). While LoWPANs are not connection-oriented, in a Mesh-Under situation the adaptation layer may have to figure out the next L2 hop and may need to provide that hop with information about the further direction to forward the packet on.
- The IP packet needs to be packaged (*encapsulated*) in the subnetwork in such a way that the subnetwork can transport it and the L2 receiver can extract the IP packet again. This leads to a number of sub problems:
    - Links may be able to carry packets of other types than just IP datagrams. Also, there may be a need to distinguish different kinds of encapsulation. Most link layers provide some form of next-layer packet type information, such as the 16-bit ether type in Ethernet or the PID (protocol ID) in PPP.
    - IP packets may not fit into the data units that layer 2 can transport. An IP network interface is characterized by the maximum packet size that can be sent using that interface, the MTU (maximum transmission unit). Ethernet interfaces most often have an MTU of 1500 bytes. IPv6 defines a minimum value for the MTU of 1280 bytes, i.e. any maximum packet size imposed by the adaptation layer cannot be smaller than that, and at least 1280 byte or smaller packets have to go through. IEEE 802.15.4 can only transport L2 packets of up to 127 bytes (and a significant part of this can be consumed for L2 purposes). In order to be able to transport larger IPv6 packets, there needs to be a way to carve up the L3 packets and put their contents into multiple L2 packets. The next IP node then needs to put those parts of a packet together again and reconstruct the IP packet. This process is often called segmentation and reassembly; 6LoWPAN calls it *fragmentation.*
    - IP was designed so that each packet stands completely on its own. This leads to a header that may contain a lot of information that could be inferred from its context. In a LoWPAN, the typical IP/UDP header size of 48 bytes already consumes a significant part of the payload space available in a single IEEE 802.15.4 packet, leaving little for applications before fragmentation has to set in. The obvious fix may be to redesign (or avoid the use of) IP. A better approach is to eliminate large parts of the redundancy at the L3–L2 interface, and this has turned out to be a good architectural position to provide *header compression*. Existing IETF standards for header compression (such as ROHC mentioned above) are too heavyweight for LoWPAN Nodes

Link Layer

6LoWPAN attempts to be very modest in its requirements on the link layer. The basic service required of the link layer is for one node to be able to send packets of a limited size to another node within radio reach (i.e. a unicast packet). n a LoWPAN, a node A may be barely (or not at all) in radio range from another node C while both have reasonable error rates to and from a node B. Instead, 6LoWPAN's requirements on a link are relaxed to an assumption that, with respect to a node A and during a period of time, there is a set of nodes relatively

likely to be reachable from A. In this book, we call this set the *one-hop neighborhood* of A. In addition, there is an assumption that A can send a local *broadcast* packet that could be (but is not necessarily always) received by all nodes in A's one-hop neighborhood.

The IEEE 802.15.4 MAC layer defines four types of frames:

Data frames for the transport of actual data, such as IPv6 frames packaged according to the 6LoWPAN format specification;

Acknowledgment frames that are meant to be sent back by a receiver immediately after successful reception of a data frame, if requested by the acknowledgment request bit in the data frame MAC header

MAC layer command frames, used to enable various MAC layer services such as asso-ciation to and disassociation from a coordinator, and management of synchronized transmission;

Beacon frames, used by a coordinator to structure the communication with its associated nodes.

Link-layer addressing

The link layer must have some concept of globally unique addressing. 6LoWPAN assumes that there is a very low likelihood of two devices coming up in the network with the same link-layer address. The fact that an address uniquely identifies a node does not mean that it is by itself useful for locating the node globally, i.e., the link-layer address is not *routable*, and it is not by itself useful for determining if a node is on the same or a different network. Data frames carry both a source and a destination address. The destination address is used by a receiver to decide whether the frame was actually intended for this receiver or for a different one. The source address is mainly used to look up the keying material for link-layer security, but may also play a role in mesh forwarding. IEEE 802.15.4 nodes are permanently identified by EUI-64 identifiers, which weigh in at 8 bytes. As a pair of 64-bit source and destination addresses already consumes one eighth of the usable space in a packet, IEEE 802.15.4 also defines a *short address* format These 16-bit addresses can be dynamically assigned during the bootstrapping of the network.

Link-layer management and operation

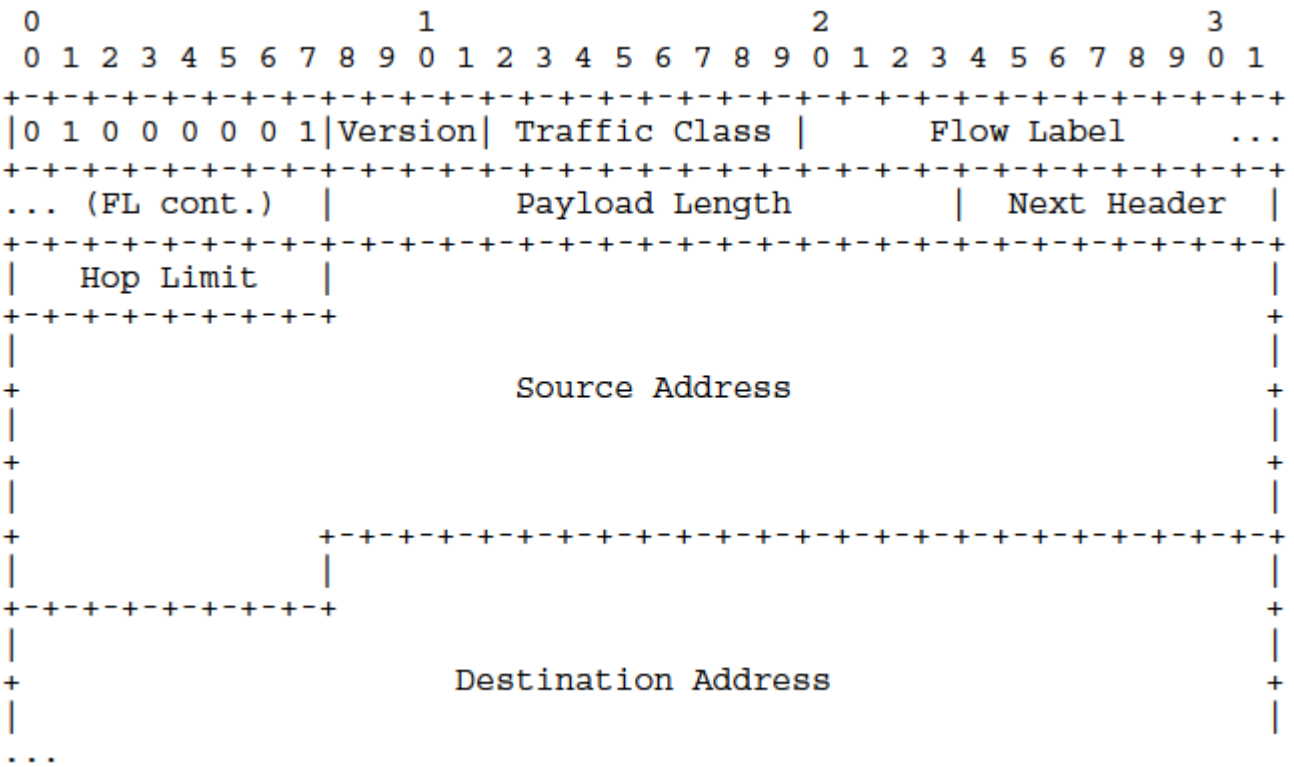


Figure 3.2.1 Uncompressed IPv6 packet with 6LoWPAN header.

Some of the formats defined by 6LoWPAN are designed to carry further 6LoWPAN PDUs as their payload. When multiple headers need to be present, the question is which header should be transported as the payload

of which other header, i.e., in which order the headers should be nested. To make this work reliably, 6LoWPAN specifies a well-defined nesting order. If present, the various 6LoWPAN headers should be used in the following order:

- **Addressing:** the mesh header (10nnnnnn, see Section 2.5), carrying L2 original source and final destination addresses and a hop count, followed by a 6LoWPAN PDU;
- **Hop-by-hop processing:** headers that essentially are L2 hop-by-hop options such as the broadcast header (LOWPAN_BC0, 01010000, that carries a sequence number to be checked at each forwarding hop), followed by a 6LoWPAN PDU;
- **Destination processing:** the fragmentation header (11nnnnnn), carrying fragments that, after possibly having been carried through multiple L2 hops, need to be reassembled to a 6LoWPAN PDU on the destination node;
- **Payload:** headers carrying L3 packets such as IPv6 (01000001), LOW-PAN_HC1 (01000010, see Section 2.6.1), or LOWPAN_IPHC (011nnnnn).

Forwarding and Routing

Packets will often have to traverse multiple radio hops on their way through the LoWPAN.
This involves two related processes: *forwarding* and *routing*. Both can be performed at layer 2 or at layer 3.

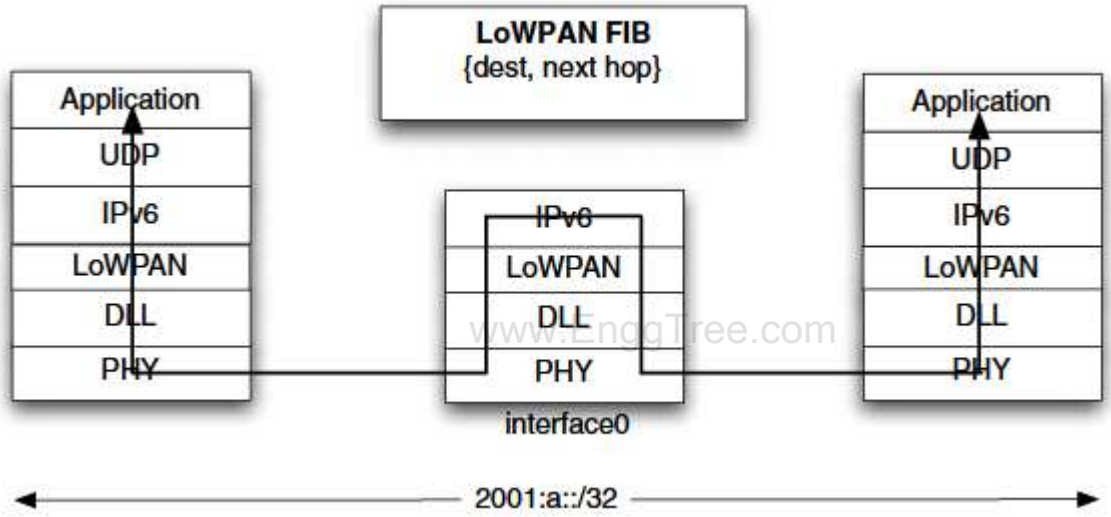


Figure 3.2.2 The LoWPAN routing model (L3 routing, "Route-Over").

L2 forwarding ("Mesh-Under")

When routing and forwarding happen at layer 2, they are performed based on layer-2 addresses, i.e., 64-bit EUI-64 or 16-bit short addresses. The IETF is not usually working on layer-2 routing ("mesh routing") protocols. To forward the packet to its eventual layer-2 destination, the node needs to know its address, the *final destination address*.

Also, to perform a number of services including reassembly, nodes need to know the address of the original layer-2 source, the *originator address*. Since each forwarding step overwrites the link-layer destination address by the address of the next hop and the link-layer source address by the address of the node doing the forwarding, this information needs to be stored somewhere else. 6LoWPAN defines the *mesh header* for this. In addition to the addresses, the mesh header stores a layer-2 equivalent of an IPv6 Hop Limit. Since the diameters of useful wireless multihop networks are usually small, the format is optimized for *hops left* values below 15 by only allocating four bits to that value. If a value of 15 or larger is needed, the 6LoWPAN packet encoder needs to insert an extension byte. This value must be decremented by a forwarding node before sending the packet on its next hop; if the value reaches zero, the packet is discarded silently. (Note that the lack of any error message means that the *traceroute* functionality that was one of the success factors of IP networking cannot be implemented for 6LoWPAN Mesh-Under.) In an implementation, there is no need to remove the extension byte when the hops left value drops to 14 or less by the decrementing process; the packet can be sent on as is or it can be optimized by removing the byte and moving the value into the dispatch byte.

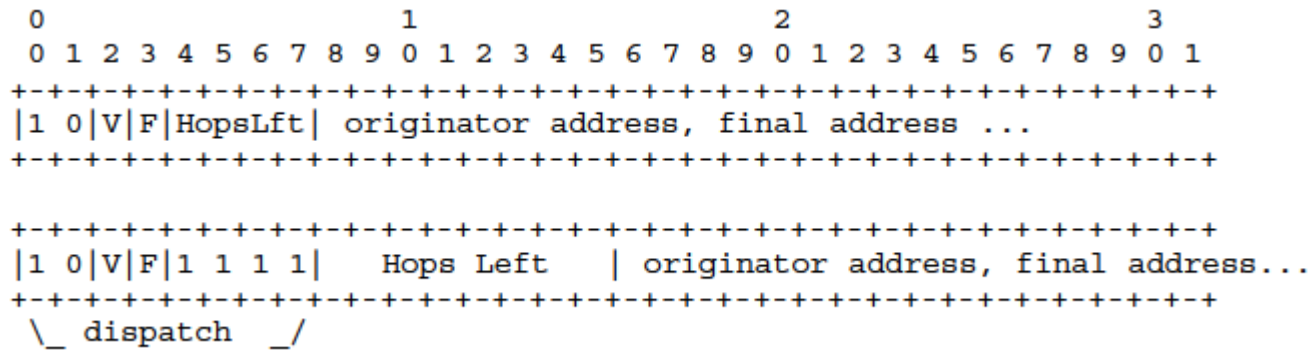


Figure 3.2.3 Mesh addressing type and header.

L3 routing ("Route-Over")

Layer-3 Route-Over forwarding is illustrated in Figure 3.2.2. In contrast to layer-2 mesh forwarding, layer-3 Route-Over forwarding does not require any special support from the adaptation layer format. Before the layer-3 forwarding engine sees the packet, the adaptation layer has done its work and decapsulated the packet – at least conceptually (implementations may be able to perform some optimizations by keeping the encapsulated form if they know how to rewrite it into the proper encapsulated form for the next layer-3 hop). Note that this in particular means that fragmentation and reassembly are performed at each hop in Route-Over forwarding – it is hard to imagine otherwise, as the layer-3 addresses are part of the initial bytes of the IPv6 header, which is present only in the first fragment of a larger packet. Again, implementations may be able to optimize this process by keeping virtual reassembly buffers that remember just the IPv6 header including the relevant addresses (and the contents of any fragments that arrived out of order before the addresses).

Fragmentation and Reassembly

IPv4 requires each node originating host or router) to be able to *fragment* packets into several smaller ones, and each final destination must be able to *reassemble* these fragments. To distinguish the whole (unfragmented) packet from its fragments, the former is often called the *datagram*. Note that the very small minimum MTU of IPv4 is often confused with the larger minimum *reassembly* buffer size: every IPv4 destination *must be able to receive a datagram of 576 [bytes] either in one piece or in fragments to be reassembled*

The fragmentation format

An 8-bit *datagram_offset* indicates the position of the fragment in the reassembled IPv6 packet; as in the IP layer, this counts in 8-byte units, so eight bits can cover the entire 2047 bytes. As with all 6LoWPAN frames, the first byte (dispatch byte) of a fragment indicates the type of frame; eight of the possible values for that byte (11100nnn) have been allocated for fragments with *datagram_size* spilling over into the dispatch byte so that no padding is required to accommodate the other bits. The resulting format, shown in Figure 3.2.4, is used for all but the initial fragment of a 6LoWPAN fragment sequence.

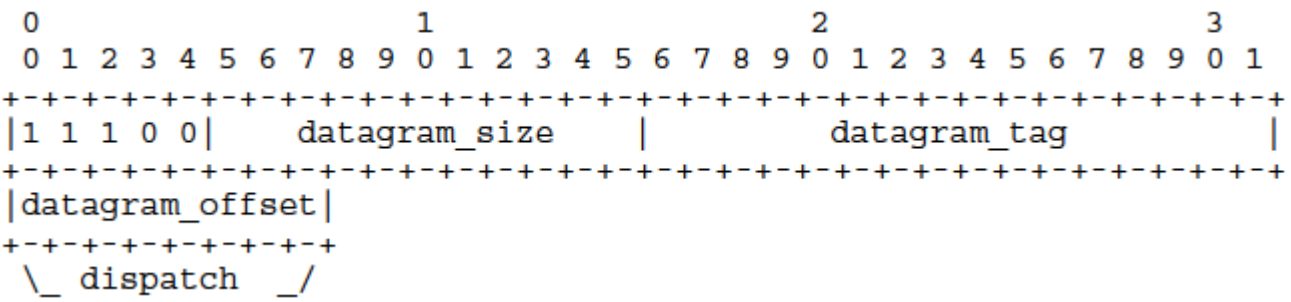


Figure 3.2.4 Non-initial 6LoWPAN fragment.

Assuming that most packets sent in a LoWPAN will be relatively small even if fragmented, a significant part of the fragments will be initial fragments with a fragment offset of all zeros. An optimization allows eliding that number; another eight possible dispatch values (11000nnn) were consumed for an alternative fragment format that implies a *datagram_offset* of zero (Figure 3.2.2).

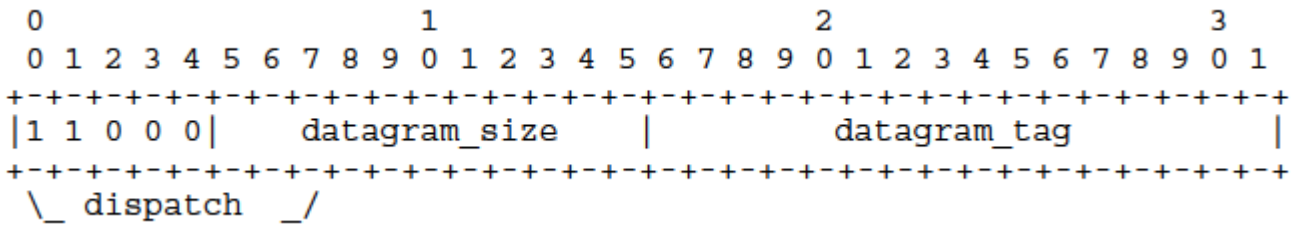


Figure 3.2.5 Initial 6LoWPAN fragment.

A node that needs to send a 6LoWPAN PDU that is too big to fit into a link-layer frame might use the following procedure:

Set variable *packet_size* to the size of the IPv6 packet (header and payload), and *header_size* to the size of the 6LoWPAN headers that need to be in the first fragment only, such as the dispatch byte and uncompressed or compressed IPv6 headers (including non-compressed fields in the latter case). If part of the IPv6 packet header or payload (such as the UDP header) is compressed away into the compressed header, adjust *header_size* down by that amount (which will usually make *header_size* negative!). In summary, *header_size* + *packet_size* is the size of the 6LoWPAN PDU that would result if it could be sent unfragmented.

Set variable *max_frame* to the space left in the link-layer frame after accounting for PHY, MAC, address and security headers and trailers, as well as any 6LoWPAN headers that may need to be prepended to each fragment or full packet (such as mesh headers for Mesh-Under). Note that *max_frame* may depend on the actual next-hop destination and the security and address size settings applicable for that. (Unless *header_size* + *packet_size* > *max_frame*, no fragmentation is needed and the PDU can simply be packaged into a link-layer frame.)

Increment a global *datagram_tag* variable to a new value that will be used in all fragments of this PDU.

Now the tricky part is to send just so much data that the first fragment is nicely filled but ends on a multiple of 8 bytes *within the IPv6 packet*. Set variable *max_frag_initial* to ^ *(max_frame* − 4 − *header_size)/*8^ ∗ 8 (leaving four bytes of space for the initial fragment header).

Send the first *max_frag_initial* + *header_size* bytes of the 6LoWPAN PDU in an initial fragment, prepending the four-byte initial fragment header to those bytes.

Set variable *position* to *max_frag_initial*.

Set variable *max_frag* to ^*(max_frame* − 5*)/*8^ ∗ 8 (leaving five bytes of space for each non-initial fragment header).

As long as *packet_size* − *position* > *max_frame* − 5:

- Send the next *max_frag* bytes in a non-initial fragment, prepending the five-byte non-initial fragment header to those bytes, filling in the value of *datagram_offset* from *position/*8.
- Increment *position* by *max_frag*.

Send the remaining bytes in a non-initial fragment, filling in the *datagram_offset* from *position*/8.

Fragment reception and reassembly might operate by this procedure:

- Build a *four-tuple* consisting of:
  – the source address,
  – the destination address
  – the *datagram_size*, and
  – the *datagram_tag*.
- If no reassembly buffer has been created for this four-tuple, create one, using the *datagram_size* as the buffer size, and initialize as empty a corresponding list of fragments received.
- For the initial fragment:
  - ✓ set variable *datagram_offset* to zero;
  - ✓ discard the four-byte fragment header;
  - ✓ perform any decoding and decompression on the contained dispatch byte and any compressed headers, as if this were a full packet, but using the full *datagram_size* for the reconstruction of length fields such as the IPv6 payload length and the UDP length;
  - ✓ set temporary variables *data* to the contents and *frag_size* to the size of the resulting decompressed packet.
- For non-initial fragments:
  - ✓ set variable *datagram_offset* to the value of the field from the packet;
  - ✓ discard the five-byte fragment header;
  - ✓ set temporary variables *data* to the contents and *frag_size* to the size of the data portion of the fragment received, i.e., minus the size of the five-byte header.
- Set variable *byte_offset* to *datagram_offset* ∗ 8.
- Check that *frag_size* either:
  - ✓ is a multiple of 8 (allowing additional fragments to line up with the end), or
  - ✓ *byte_offset* + *frag_size* = *datagram_size* (i.e., this is the final fragment);
  - ✓ if neither is true, fail (there would be no way to fill in the remaining bytes).
- If any of the entries in the list of fragments received before overlaps the interval [*byte_offset*, *byte_offset* + *frag_size)*:
  - ✓ If the overlapping entry is identical, discard the current fragment as a duplicate.
  - ✓ If not, fail.
- Otherwise, add the interval to the list.
- Copy the contents of *data* to the buffer positions starting from *byte_offset*.
- If the list of intervals now covers the whole span, the reassembly is complete, and the buffer contains a reassembled IPv6 packet of size *datagram_size*. Perform any final processing that requires the whole packet such as reconstructing a compressed-away UDP checksum.

Avoiding the fragmentation performance penalty

Fragmentation is undesirable for a number of reasons, ,the problem most often discussed is the decoupling between unit of loss (the fragment) and unit of retransmission (the entire packet), with the related inefficiencies. Possibly even more important in a resource-constrained embedded environment, the uncertainty of when the remaining fragments for a reassembly buffer will be received makes management of the resources assigned to reassembly buffers very difficult. This is probably less of an issue for more resource-heavy edge routers, but can make reception of fragmented packets by battery-operated systems with limited RAM quite unreliable. As a rule of thumb, fragmentation is marginally acceptable for packets originating from such devices (where the main problem is the reduced probability of the whole reassembled packet arriving intact), but should be avoided for packets being sent to them.

3.3 Mobility

Mobility in IP networks technically is the act of a node changing its topological point of attachment. Koodli and Perkins distinguish the following two kinds of mobility

- **Roaming:** A process in which a mobile node moves from one network to another, typically with no existing packet streams.
- **Handover:** A process in which a mobile node disconnects from its existing point of attachment and attaches itself to a new point of attachment. Handover may include operations at specific link layers as well as at the IP layer in order for the mobile node to be able to communicate again. One or more application packet streams typically accompany the mobile node as it undergoes handover

Mobility can alternatively be described with the terms micro-mobility and macro-mobility. Micro-mobility refers to mobility that occurs within a network domain. In 6LoWPAN we can consider micro-mobility to refer to the mobility of a node within a LoWPAN where the IPv6 prefix does not change, which is the definition used here. Macro-mobility on the other hand refers to mobility between networks. In 6LoWPAN we consider macro-mobility to refer to mobility between LoWPANs, in which the IPv6 prefix changes. In relation to the previously defined terms, we can consider micro-mobility to require only handover, whereas macro-mobility is a process of roaming and handover

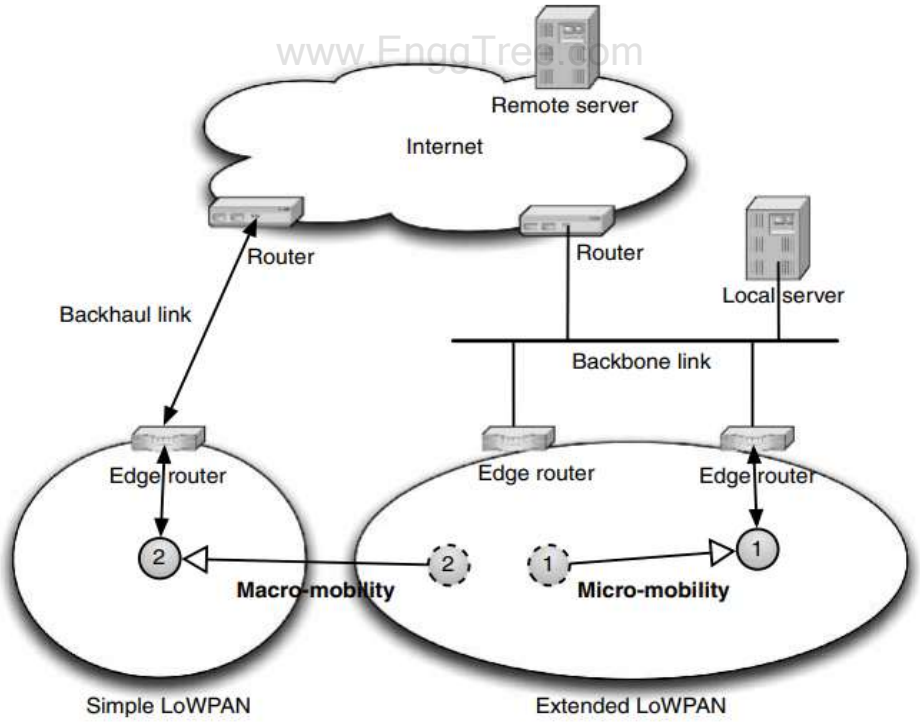


Figure 3.3.1 The difference between micro-mobility and macro-mobility.

Before looking at solutions for dealing with mobility, it helps to understand why mobility happens in the first place. In wireless networks there are a number of things that may cause a network to make a change in topology. The causes of

topology change can be categorized simply as physical movement, radio channel changes, network performance, sleep schedules and node failure:

- **Physical movement**: The most evident reason for mobility is when nodes in a network physically move in relation to each other, which changes the wireless connectivity between pairs of nodes. This may cause nodes to change their point of attachment.
- **Radio channel:** Changes in the environment cause changes in radio propagation, called fading. These changes often require topology change even without physical movement, especially with simple radio technologies.
- **Network performance:** Packet loss and delay on wireless networks may be caused by poor signal strength, collisions, overloaded channel capacity or node congestion. High packet loss may cause a node to change its point of attachment.
- **Sleep schedules:** Especially battery powered nodes in wireless embedded networks use aggressive sleep schedules in order to save battery power. If a node finds itself attached to a sleeping router without a suitable duty cycle for the application, this may cause the node to move to a better point of attachment.
- **Node failure:** Autonomous wireless nodes tend to be prone to failure, for example due to battery depletion. The failure of a router causes a topology change for nodes using it as their default router.

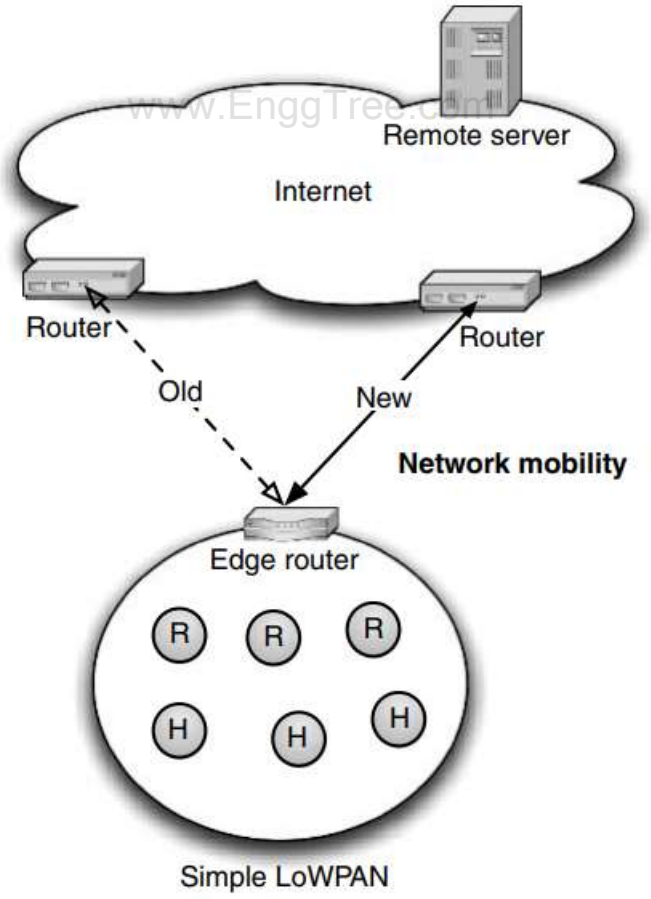


Fig 3.3.2 Network mobility example.

An example of network mobility is shown in Figure 3.3.2 When considering 6LoWPAN, network mobility occurs when an edge router changes its point of attachment while nodes in the LoWPAN remain attached to it. The kind of network mobility that affects 6LoWPAN is clearly macro-mobility, when the IPv6 address of the edge router changes, as this affects the addressing of all nodes in the LoWPAN.

Mobile IPv6

The mobility of nodes on the Internet can be dealt with at the network layer using a protocol called Mobile IP (MIP). The goal of MIP is to deal with the mobility roaming problem by allowing a host to be contacted using a well-known IP address, regardless of its location on the Internet. Mobile IP does this using the concept of a home address, which is associated with a host's home network. When a host is away from its home network, and attaches to another network domain (called the visited network), the new IP address it configured there is called its care-of address. A node communicating with a mobile node roaming in a visited network is called the correspondent node. Normally forwarding in IP networks is handled only by routers, whose route tables are maintained by routing protocols. Mobile IP works using a special kind of routing functionality, which is host controlled. This concept is called a binding, and is implemented.

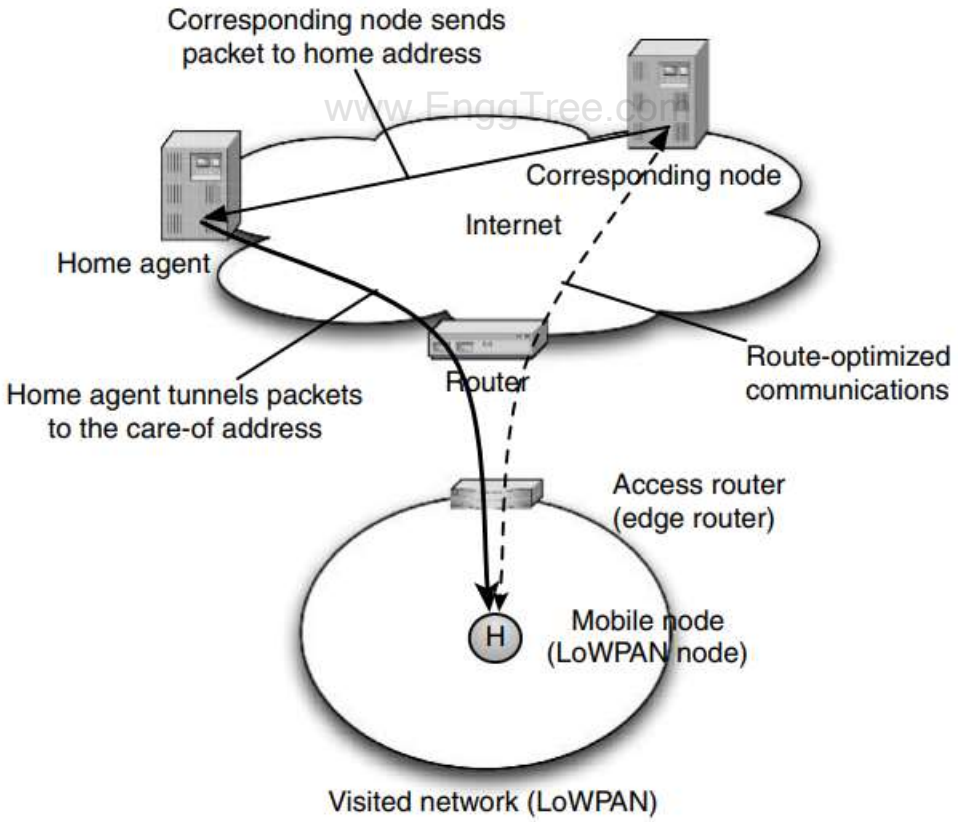


Figure 3.3.3 Example of Mobile IPv6 used with 6LoWPAN.

Figure 3.3.3 illustrates basic MIPv6 functionality. When a mobile node roams to a visited network, it uses MIPv6 in the following way to maintain global connectivity via its home address:

- After detecting that the subnet has changed, and that the node is no longer in its home network, it sends a MIPv6 binding update message to its HA. If the node doesn't know the location of its HA or its home prefix, there are methods to discover both. The binding update is acknowledged by the HA with a binding acknowledgment. These messages must be secured using e.g. IPsec methods.
- A bidirectional IPv6-in-IPv6 tunnel is then set up between the HA and mobile node for exchanging data packets. When an incoming packet arrives from a corresponding node to the home address of a mobile node at its home network, it is intercepted by the HA using an ND proxy technique. The packet is then encapsulated in another IPv6 header with the destination address set to the mobile node's care-of address.
- After receiving and decapsulating the packet, the mobile node can then respond through the IPv6-in-IPv6 tunnel through its HA back to the corresponding node. Alternatively, the mobile node can respond directly to the corresponding node using its care-of address.
- This is a case of triangular routing, where a more optimal path directly between the mobile node and corresponding node is possible. MIPv6 includes route optimization to avoid triangular routing. After executing a reverse routability test and possible security association between the correspondent node and mobile node, they can begin to communicate directly. First a binding update is sent to the corresponding node. Then data traffic is exchanged using IPv6 extension headers to properly indicate the actual home address of the mobile node.

In order for MIPv6 to be applied to 6LoWPAN Node mobility, it would have to be implemented on LoWPAN Nodes. The use of MIPv6 as defined in with 6LoWPAN has the following problems:

- IPv6-in-IPv6 tunneling between the HA and LoWPAN Node would incur large header overheads as the encapsulated IPv6 and transport headers cannot be compressed by the existing compression methods.
- The requirement of IPsec security associations between MIPv6 entities may be unreasonable for LoWPAN Nodes.
- The added complexity of implementing MIPv6 in terms of code size and RAM may be unjustifiable for LoWPAN Nodes.
- In domains with large LoWPANs and frequently mobile nodes, the traffic burden caused by MIPv6 may be too much for low-bandwidth wireless links.
- Route optimization adds an even greater burden on nodes, as state must be maintained for every active correspondent node.

Proxy Home Agent

A proxy Home Agent (PHA) is an entity which performs MIPv6 functions on behalf of a local mobile node, interacts with the actual Home Agent of the node, and handles route optimization on its behalf. This greatly simplifies the functions that a mobile node needs to perform to participate in MIPv6. In 6LoWPAN this is an especially critical

optimization as seen from the previous section. A global architecture for PHA is described in [ID-global-haha], where PHA functionality is described.

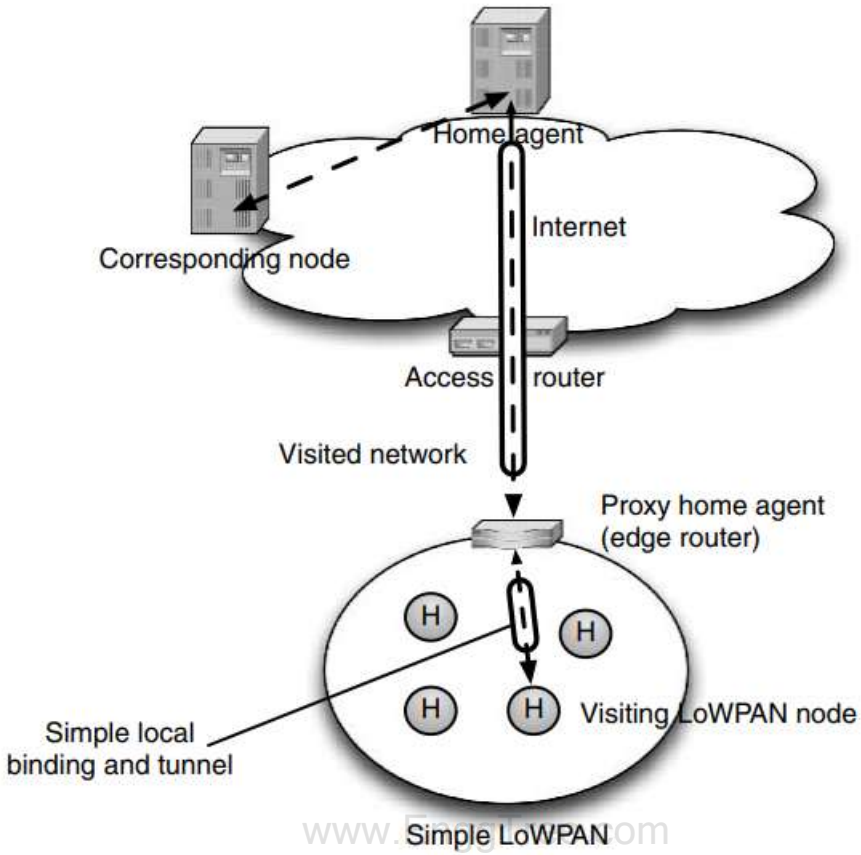


Fig 3.3.4 Example of a proxy Home Agent located on an edge router.

The PHA is located in the visited network where a mobile node is roaming, in 6LoWPAN this would logically be the LoWPAN Edge Router. A PHA acts like a normal MIPv6 host, but additionally performs binding updates, HA tunneling and route optimization on behalf of other nodes. In order for a mobile node to use a PHA, it simply needs to perform a local binding update with (possibly much simpler) credentials, and to create a single tunnel to the PHA. The PHA architecture is show in Figure 3.3.4. This provides huge improvements in efficiency with regards to security associations and for route optimization which requires tunnels and other state for every correspondent node. In order to use the PHA concept with 6LoWPAN, a mechanism for registering with the PHA would need to be defined for use inside the LoWPAN. The logical place to do this would be as an option for the 6LoWPAN-ND Node Registration message. Such an option would need to include the Home Agent's address or home prefix, the node's home address and some credentials (if L2 credentials are not sufficient). As the tunnel is local between the LoWPAN Edge Router and the LoWPAN Node, it could be realized as a simple IPv6 extension header option with very low overhead.

Proxy MIPv6

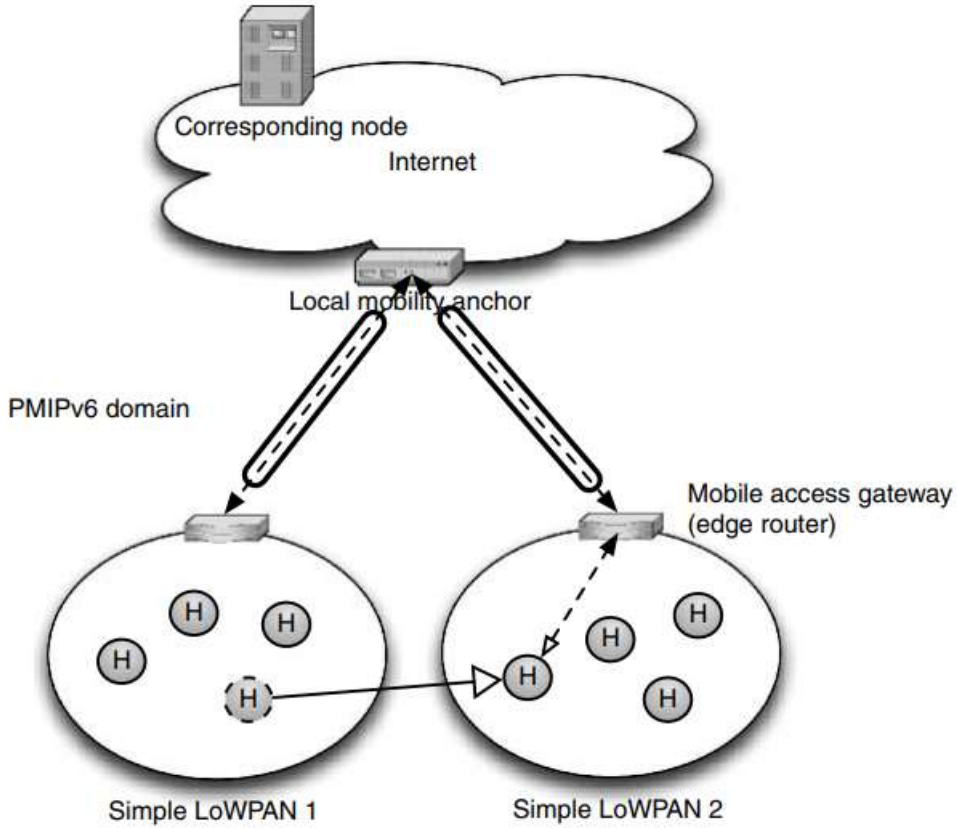


Fig 3.3.5 Example of PMIPv6 with 6LoWPAN.

The architecture of PMIPv6 is illustrated in Figure 3.3.4. The concept of a PMIPv6 domain is introduced, which is controlled by a local mobility anchor (LMA). The LMA function is usually combined with HA functionality. The LMA handles the local mobility of nodes with the help of mobile access gateways (MAGs), which are points of attachment supporting PMIPv6. MAGs send proxy binding updates to the LMA on behalf of mobile nodes attached to them. Using bidirectional tunnels built between each MAG and the LMA, the LMA is then able to forward traffic to the mobile node always using its static address (known as a mobile node home address). A binding in the LMA is made between this address and the temporary address from the visited MAG (the proxy care-of address). PMIPv6 makes use of RS/RA exchanges directly between the mobile node and MAG in order to detect when the mobile node has changed its point of attachment.

Although the PMIPv6 model would seem to fit well with 6LoWPAN, there are some problems that would still need to be solved:

- The RS/RA exchange defined in is not compatible with a multihop Route-Over LoWPAN, and would require each LoWPAN Router to act as a MAG.
- PMIPv6 is meant to provide a separate 64-bit prefix for each mobile node.
- PMIPv6 only enables a node to talk with its point of attachment (default router), and requires NS/NA exchanges which are not required by LoWPAN Nodes otherwise using 6LoWPAN-ND.

NEMO

Network mobility (NEMO) is a solution for dealing with network mobility problems, when a router and the nodes attached to it, move their point of attachment all together. The philosophy behind NEMO is to extend Mobile IP so that each node does not need to run Mobile IP, instead only the router they are attached to runs Mobile IP. This philosophy fits the 6LoWPAN model perfectly as LoWPAN Nodes are not capable of dealing with MIPv6. Edge routers or other router entities run full IPv6 stacks, and have the capability of dealing with MIPv6.

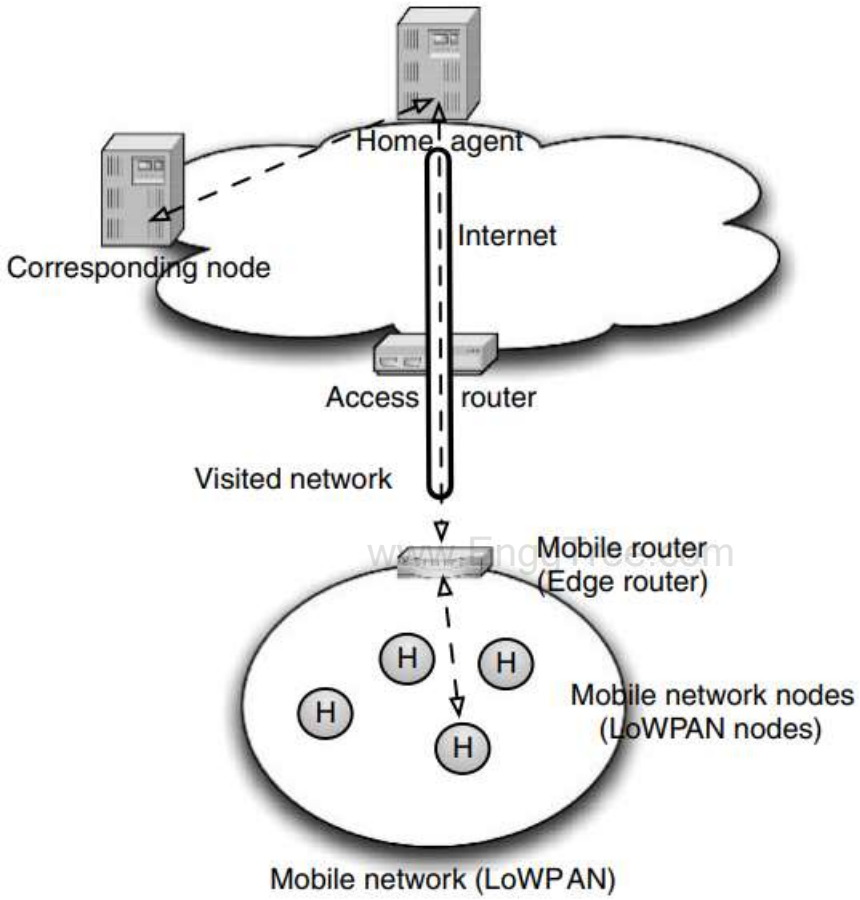


Figure 3.3.6 Example of the basic NEMO protocol working with 6LoWPAN.

The NEMO protocol works by introducing a new logical entity called the mobile router, which is responsible for handling MIPv6 functions for the entire mobile network. Mobile nodes which are part of the mobile network are called mobile network nodes (MNNs). These entities can be seen from Figure 4.7. MIPv6 normally only handles forwarding for the home addresses bound by mobile nodes. NEMO extends the functionality of the Home Agent to be able to deal with prefixes in addition to home addresses of mobile nodes. A mobile router functions like a normal MIPv6 host setting up a bidirectional tunnel with its Home Agent, but in addition it negotiates prefixes to be forwarded to it by the Home Agent. The Home Agent then forwards all packets matching the bound prefix (therefore packets for the MNNs) to the mobile router. A special flag in the binding update allows the mobile router to indicate it wants prefix forwarding,

and a prefix option lets it configure prefixes with the HA. Alternatively, prefix delegation can be done using e.g. DHCPv6

NEMO is clearly beneficial when applied to mobile LoWPANs, where the entire LoWPAN including the edge router and associated nodes move together to a new point of attachment. When this happens the edge router acts as a NEMO mobile router. Using MIPv6, it binds its new care-of address in the visited network, and in addition the home LoWPAN prefix. Thus inside the LoWPAN no change can be noticed due to network mobility, as the LoWPAN continues to use the same prefix as in its home network. The HA takes care of forwarding all traffic destined for the LoWPAN prefix through the tunnel to the edge router and vice versa.

The drawback of NEMO is that it can not deal with individual node mobility on behalf of the LoWPAN Nodes. Thus a mobile LoWPAN Node would still have to implement MIPv6, unless it would use a proxy Home Agent or be moving within a PMIPv6 domain as discussed in the previous sections. Furthermore, NEMO starts to become complicated when mixing different kinds of node mobility along with PMIPv6.

3.4 Routing

When discussing IP routing for 6LoWPAN, there are two types of routing which need to be considered: routing inside a LoWPAN, and routing between a LoWPAN and another IP network. Routing is challenging for 6LoWPAN, with low-power and lossy radio links, battery-powered nodes, multihop mesh topologies, and frequent topology change due to mobility. Successful solutions must take the specific application requirements into account, along with Internet topology and 6LoWPAN mechanisms.

As IP networks are *packet switched*, as opposed to circuit switched, forwarding decisions are made *hop-by-hop*, based on the destination address in a packet. Therefore reaching a destination node in a network from a source node requires building a path from the source to the destination node in route tables on nodes along the path. IP addresses are structured, and this structure is used to group addresses together under a single route entry. In IPv6 an address prefix is used for this purpose, which is why this is called prefix-based routing.

- LoWPAN Routers typically perform forwarding on a single wireless interface, i.e. they receive a packet on their wireless interface from one node, and then forward it to the next-hop destination using the same interface. This is an important difference to how forwarding on IP routers normally works, where packets generally are forwarded between interfaces (and thus between links). The reason for this model is that typically not all nodes in a LoWPAN are reachable in a single wireless transmission, thus IP forwarding is used to provide full connectivity over multiple hops within the same "link".
- A LoWPAN has a *flat* address space, as all nodes in a LoWPAN share the same IPv6 prefix. This is due to the way 6LoWPAN compression is achieved, using the fact that all nodes in the network know common information to elide or compress fields. Therefore 6LoWPAN routing tables only contain entries to destination addresses in the LoWPAN, along with default routes.
- LoWPANs are stub networks, and are not meant to be transit networks below

Two kinds of routing can be performed with 6LoWPAN, *intra-LoWPAN* routing between LoWPAN Routers, and *border routing* performed at the edge of the LoWPAN by the LoWPAN Edge Router or an IPv6 router on the backbone link for Extended LoWPANs. These routing domains and the associated network-layer forwarding are illustrated in Figure 4.8 and Figure 4.9.

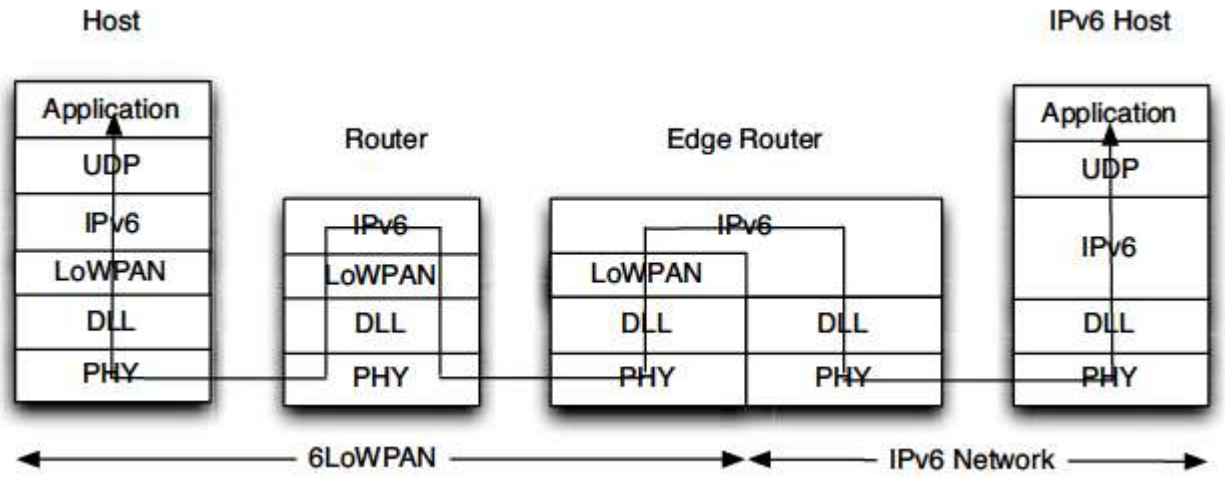


Fig: 3.4.1 Stack view of forwarding inside the LoWPAN and across the edge router.

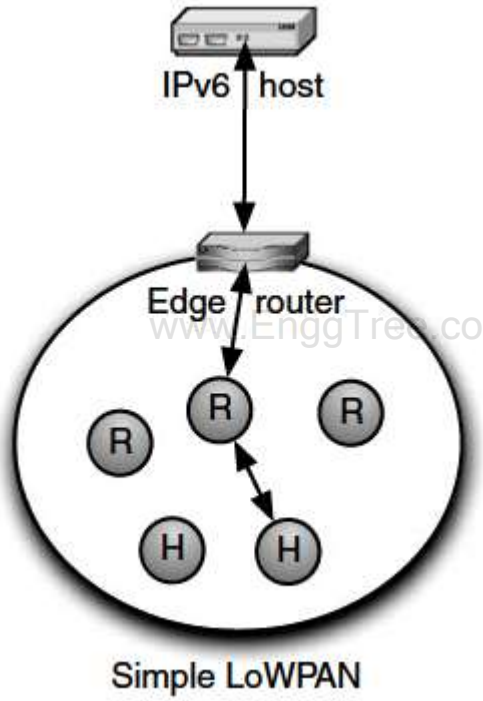


Fig 3.4.2 Topology view of forwarding inside the LoWPAN and across the edge router.

There are two main classes of routing protocols useful for 6LoWPAN: *distance-vector* routing and *link-state* routing.

Distance-vector routing:
These algorithms are based on variations of the Bellman–Ford algorithm. Using this approach, each link (and possibly node) is assigned a cost, using appropriate route metrics. When sending a packet from node A to node B, the path with the lowest cost is chosen. The routing table of each router keeps soft-state route entries for the destinations it knows about, with the associated path cost. Routing information is updated either *proactively* (a priori) or *reactively* (on-demand) depending on the routing algorithm. Owing to their simplicity, low signaling overhead and local adaptive nature, distance-vector algorithms are commonly applied to 6LoWPAN.

Link-state routing: In this approach, each node acquires complete information about the entire network, called a graph. To do this, each node floods the network with information about its link information to nearby destinations. After receiving link-state reports from sufficient nodes, each node then calculates a tree with the

shortest-path (least cost) from itself to each destination using e.g. Dijkstra's algorithm. This tree is used either to maintain the routing table in each node for hop-by-hop forwarding, or to include a source-route in the header of the IP packet. Link-state algorithms incur a large amount of overhead, especially in networks with frequent topology change. They require substantial memory resources for the amount of state needed by each node. Thus they are not suitable for distributed use among LoWPAN Nodes [ID-roll-survey].

Link-state algorithms may be usefully applied off-line on LoWPAN Edge Routers which have sufficient memory capacity if the signaling overhead for collecting the link-state information is reasonable. In this way only a single tree is constructed from the ER to nodes.

In order to update routing information throughout a network or along a path, routing protocols make use of either proactive or reactive signaling techniques. These terms can be defined as:

Proactive routing: Algorithms using a proactive approach build up routing information on each node before the routes are needed. Thus they proactively prepare for the data traffic by learning routes to all possible or likely destinations. Most protocols that are used in inter-domain or intra-domain IP routing use a proactive approach as topologies are stable. Examples of proactive algorithms can also be found from MANET, for example optimized link-state routing (OLSR) and topology dissemination based on reverse-path forwarding (TBRPF). The advantage of this approach is that routes are immediately available when needed, but this comes at the cost of increased signaling overhead especially with frequent topology changes and increased state for routers.

Reactive routing: Reactive routing protocols store little or no routing information after autoconfiguration of the routing protocol. Instead, routes are discovered dynamically only at the time they are needed. Thus a process called *route discovery* is executed when a router receives a packet to an unknown destination. Examples of reactive algo-rithms include the MANET ad hoc on-demand distance vector (AODV) [RFC3561] and dynamic MANET on-demand (DYMO) [ID-manet-dymo] protocols, along with the ZigBee routing algorithm derived from AODV [ZigBee]. The advantage of this approach is that signaling and route state grows only as needed, and it is especially well suited to ad hoc networks with frequent topology change and mainly peer-to-peer communications.

Advanced techniques which may be applied in 6LoWPAN routing protocols include constrained routing using compound route metrics, local route recovery, flow labeling to achieve Multi Topology Routing (MTR), forwarding on multiple paths with multipath routing, and traffic engineering. Several of these techniques are being considered for the ROLL routing algorithm.

MANET routing protocols

MANET has also produced valuable work on basic mechanisms for supporting routing in these environments. A common packet format for use by all MANET protocols. Finally, a *two-hop*neighborhood discovery protocol is currently being developed for use in collecting route information in a standard way.

AODV

The ad hoc on-demand distance vector (AODV) protocol enables mobile ad hoc multihop networks by quickly establishing and maintaining routes between nodes, even with quickly changing (dynamic) topologies. AODV creates routes to destinations when needed for data communications (reactive), and only maintains actively used routes. It includes methods for local repair, and includes a destination sequence number to ensure loop-free operation.

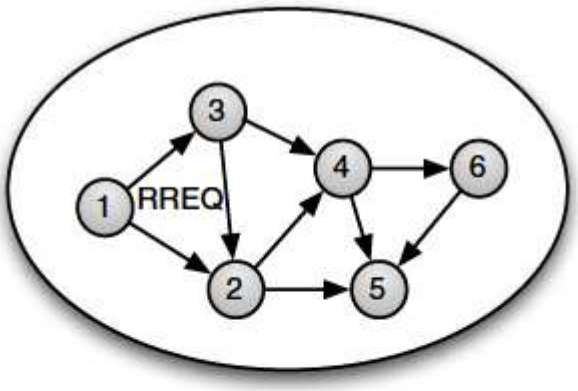
AODV is purely a route table management protocol; after routes have been established they are simply used by IP for forwarding.A small set of messages are used for discovering and maintaining routes by AODV and similar protocols. A *route request* (RREQ) is broadcast throughout the network in order to find paths to a destination. This is responded to with a *route reply* (RREP) by an intermediate router or by the destination. Figure 4.10 shows an example of reactive route discovery and forwarding in an ad hoc network. The *route error* (RERR) message is used to notify about broken links along a path. These messages are sent over UDP, one hop at a time, between the AODV processes running on ad hoc routers. AODV is specified in detail in [RFC3561].

As AODV was the first reactive distance-vector routing algorithm standardized in the IETF, it has been used as a model for many other routing algorithms. For example, the routing algorithm that ZigBee used in its network layer designs is modeled on the AODV algorithm with modifications to minimize overhead and to function on MAC addresses rather than IP addresses.

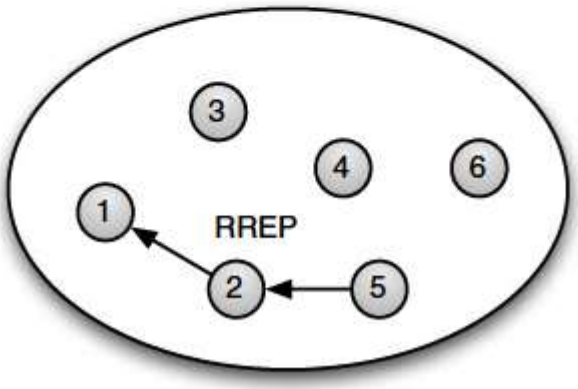
DYMO

A new reactive distance-vector routing protocol called the dynamic MANET on-demand (DYMO) protocol has been developed in the MANET WG [ID-manet-dymo], making improvements on previous protocols such as AODV and dynamic source routing (DSR). This protocol makes use of the same types of route discovery and maintenance messages as AODV. The main improvements compared to previous work include:

- improved convergence in dynamic topologies
- use of the common MANET packet format [RFC5444]
- support for a wide range of traffic flows



1. RREQ for node 5 broadcast over multiple hops.



2. RREP unicast back to node 1, creates route entries.

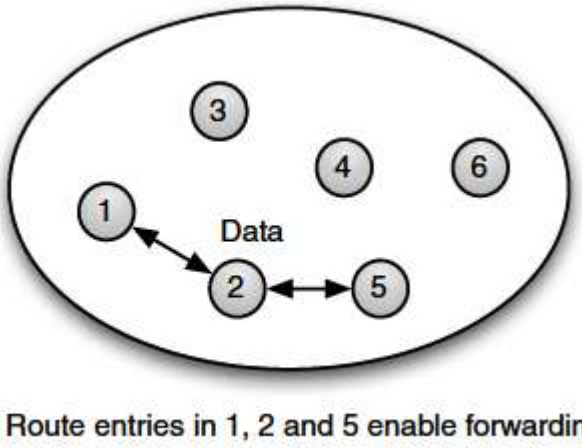


Figure 3.4 Example of reactive distance-vector routing.

OLSR

The MANET WG has also produced a proactive link-state routing protocol called the optimized link-state routing (OLSR) algorithm. Originally specified in [RFC3626], an improved OLSRv2 has been developed in the working group in [ID-manet-olsrv2]. This algorithm applies optimization to the classical link-state algorithm for use in mobile ad hoc networks. In order to build link-state tables, OLSR routers regularly exchange topology information with other routers. The flooding of this information is controlled by the use of selected multipoint relay (MPR) nodes. These MPR nodes are used as intermediate routers, and thus enable a kind of clustering technique. The OLSR algorithm makes use of the standard MANET packet format and two-hop ND techniques.

OLSR is best suited for relatively static ad hoc networks, thus minimizing the number of link-state updates throughout the network, which can cause a lot of overhead. OLSR is not very well suited to 6LoWPAN Routers because of the large amount of signaling and routing state. Link-state protocols are also not well suited to tree topologies, as often found in 6LoWPAN applications. Link-state approaches like OLSR may have possible uses for the partial optimization of larger route topologies or off-line use on border routers, for example with the ROLL routing algorithm.

The ROLL routing protocol

- Traffic patterns are not only peer-to-peer unicast flows, but more often point-to-multipoint or multipoint-to-point flows. Most applications of LLNs are Internet connected.
- Routers in LLNs have a very small, hard bound on state (limited memory).
- Most LLNs must be optimized for energy consumption.
- In most cases LLNs will be deployed over links with a limited frame size.
- Security and manageability are extremely important as LLNs are typically autonomous.
- The application spaces aimed at by ROLL are heterogeneous. Each may need a different set of features along with routing metrics to fulfill its requirements.

The following activities are going on in the ROLL working group:

- ✓ **Metrics:** The routing metrics useful for path calculation have initially been specified in [ID-roll-metrics]. In practice, work will still need to be done on algorithms for *evaluating* appropriate metrics for each specific application space.
- ✓ **Architecture:** The basic architectural requirements for ROLL have been captured in the requirement documents. Terminology for use in ROLL has been specified in [ID-roll-terminology].

- ✓ **Security:** A security framework is being developed in the working group. An overview of requirements and some techniques for ROLL security have been provided in [ID-roll-security], and some considerations for trust management are collected in [ID-roll-trust].
- ✓ **Protocol:** The goal is to design a routing protocol that can be successfully applied to fulfill the routing requirements of the four application areas identified for ROLL. Early contributions towards this goal have been made, and at the time of writing the initial ROLL protocol is being designed.

ROLL architecture

The architecture of LLNs is very different from that considered by MANET protocols or in research work done on wireless sensor networking. In fact it has more in common with traditional intra-domain IP routing methods. The ROLL protocol can be classified as a proactive distance-vector algorithm with advanced options for constraint-based routing, multi-topology routing and traffic engineering. The key requirements or assumptions for LLNs that affect the routing architecture are:

- ✓ LLNs are Internet-connected stub networks, with support for multiple points of attachment (multiple border routers to other IP networks).
- ✓ The majority of traffic flows are going to or from border routers using unicast, point-to-multipoint or multipoint-to-point flows. Node-to-node communication is less common, but may require specific constraints.
- ✓ Support for dynamic topologies and mobility is required.
- ✓ Support is required for multipath routing, and thus multiple forwarding options.
- ✓ Support is required for multiple node and route metrics, and their application in constraint-based and multi-topology routing. The evolution of metrics and support of multiple scenarios are important.
- ✓ A course-grained depth metric is assumed for general use, which is independent of the specific scenario. It is not assumed that this metric provides absolute loop avoidance.
- ✓ Routers in LLNs have limited memory resources.
- ✓ Most applications will require enterprise-class security.

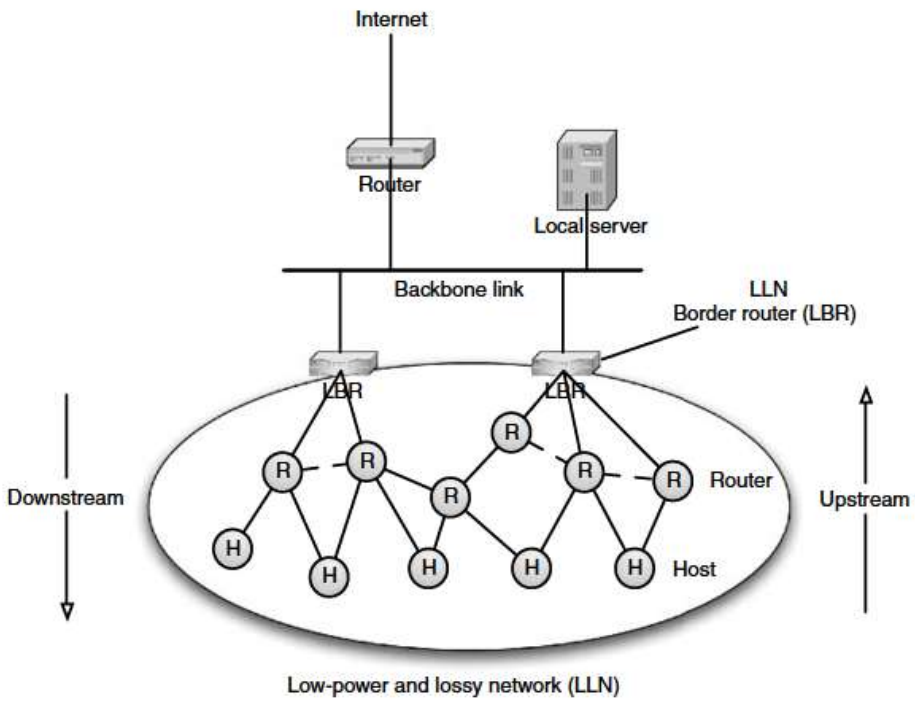


Figure 4.11 The ROLL architecture.

There are two concepts important to understanding ROLL protocol operation:
Metric granularity: ROLL uses the concept of a very granular (16–32 values) route metric called *depth*. This metric is used by the basic ROLL protocol mechanisms for building the graph, making use of siblings and for

loop avoidance. The evaluation of depth is simple for all routers and nodes and is independent of the application scenario. In addition, fine-grained sets of metrics (such as those presented in Section 4.2.4) along with evaluation algorithms are applied in an application-specific manner to achieve routing on the basic graph structure.

Routing time scale: ROLL makes routing decisions on two different time scales: route-setup time and packet-forwarding time. In *route-setup time* the routing protocol maintains the basic graph topology and routing tables using static or slowly moving metrics, which is a continuous process. In addition, ROLL enables *packet-forwarding time* decisions to be made using dynamic metrics on a packet-by-packet basis, for example the immediate use of alternative next-hop routers upon failure.

4.1 Introduction

Wireless Embedded Internet systems are usually designed for a specific purpose, for example a facility management network or for a simple home automation system. These two examples happen to have widely different application protocol requirements. Currently, large building automation systems are pre-configured to function in that environment, require management with e.g. SNMP, and often make use of industry-specific protocols such as BACnet. A home automation system on the other hand requires service discovery protocols such as SLP, and may make use of web-service style or proprietary protocols for data and management. What makes 6LoWPAN very different from vertical communication solutions is that the same network can be used by a large variety of devices running different applications thanks to the Internet model. All the protocols mentioned above can be run over the same IP network infrastructure, simultaneously. IP uses what is often called a *horizontal* networking approach.

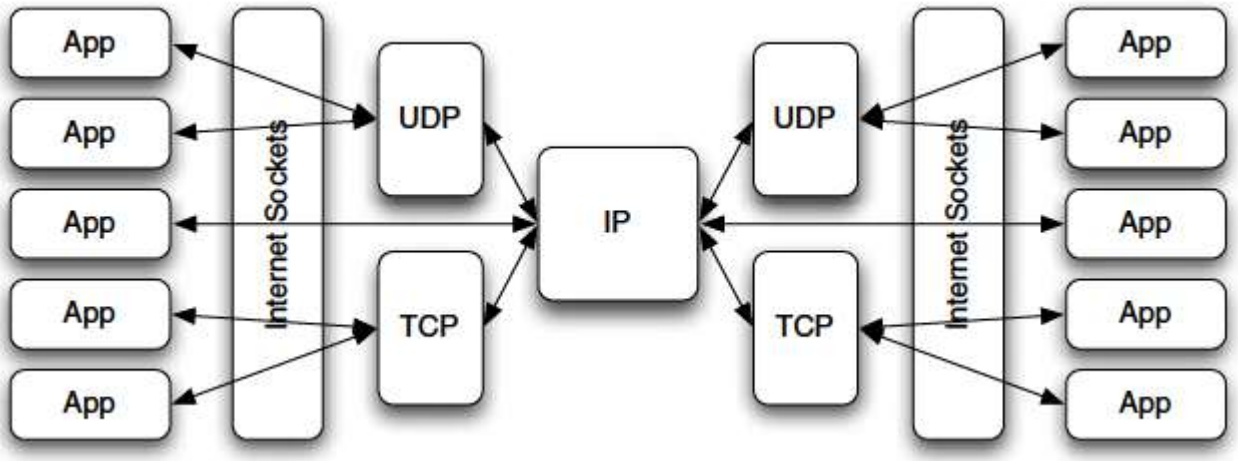


Figure 4.1.1 Applications process communication occurs through Internet sockets.

Although the Internet Protocol provides basic packet networking over heterogeneous links, it is UDP and TCP that allow for the large range of application protocols by providing *best-effort* (UDP) [RFC0768] and *reliable connection-oriented* (TCP) [RFC0793] multiplexed communications between application processes. IP protocols use a socket-based approach, where process *end-points* are identified by 16-bit source and destination port identifiers [RFC1122]. These are commonly called Internet sockets or network sockets.

The concept is illustrated in Figure 4.1.1 The communication between any two end-points is uniquely identified for each transport by a four-tuple consisting of the local and remote socket addresses:

{src IP address, src port, dst IP address, dst port}

Application protocols use a socket API to access *datagram socket* (UDP) and *stream socket* (TCP) transport services along with *raw socket* (IP) services within a protocol stack. The different types of sockets are completely independent of each other (e.g. UDP port 80 and TCP port 80 can be used simultaneously). 6LoWPAN supports the compression of UDP ports down to a range of 16 [RFC4944], which is useful because a LoWPAN usually has a limited number of applications.

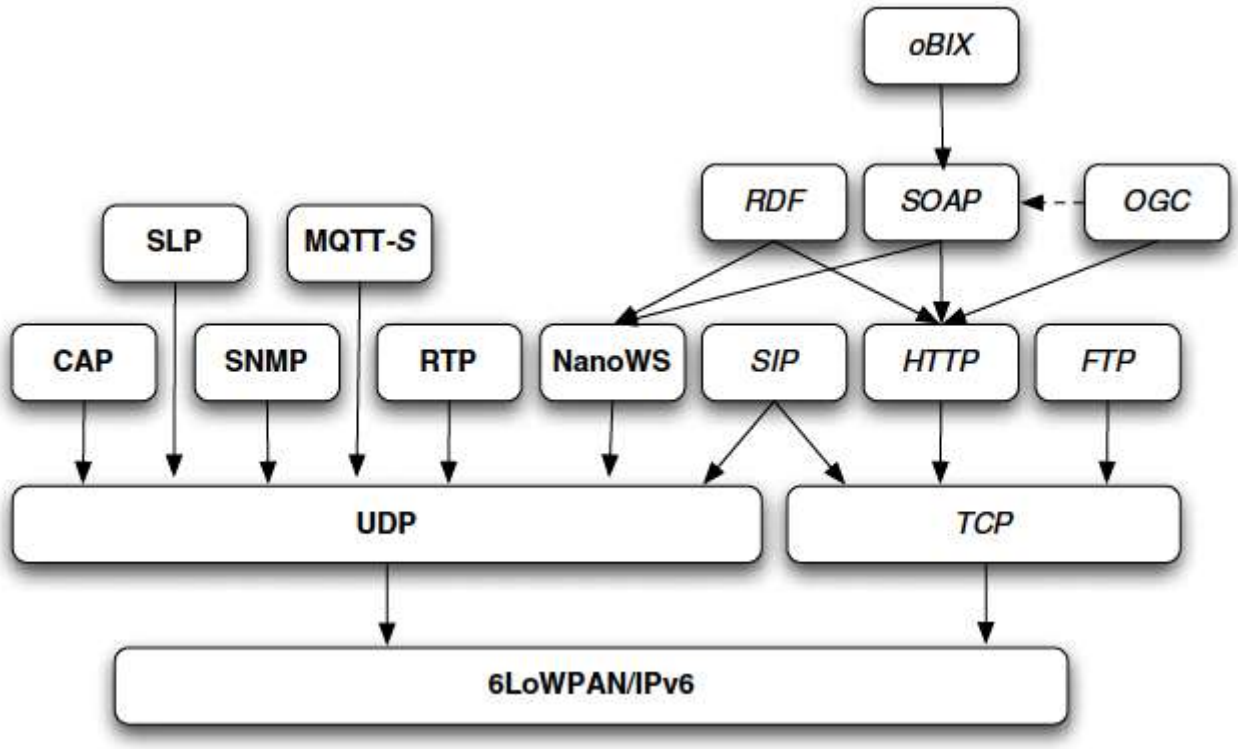


Figure 4.1.2 The relationship of common IP protocols.

Design Issues

Application protocols used over 6LoWPAN need to take a number of requirements into account which are typically not an issue over general IP networks. These issues include:

- Link layer: Link-layer issues include lossy asymmetrical links, typical payload sizes of 70–100 bytes, limited bandwidth, and no native multicast support.
- Networking: Networking related issues include the use of UDP, limited compressed UDP port space and performance issues regarding the use of fragmentation.
- Host issues: Unlike typical Internet hosts, 6LoWPAN hosts and networks are often mobile in nature during operation. Furthermore, battery-powered nodes use sleep periods with duty cycles often between 1–5 percent. A node may be identified in many ways, e.g. using its EUI-64, its IPv6 address or by a domain name, which should be taken into account.
- Compression: The small payload sizes available often require compression to be used on existing protocols. Issues to consider include header and payload compression, and whether it is performed end-to-end or by an intermediate proxy.
- Security: 6LoWPAN makes use of link-layer encryption which protects a single hop. Intermediate nodes are susceptible to attack, requiring sensitive application to employ end-to-end application-level security. Edge routers need to implement firewalls in order to control the flow of application protocols in and out of LoWPANs.

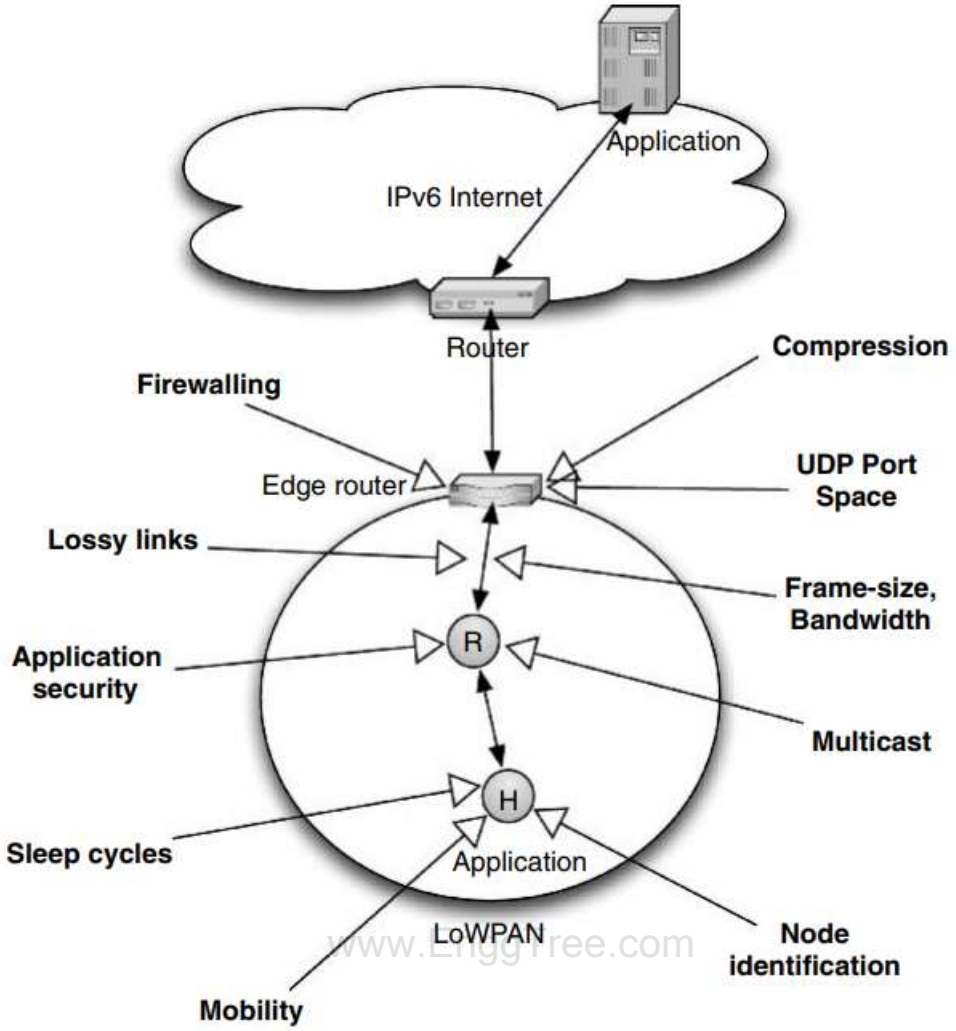


Fig 4.1.1 Application design issues to consider and where they occur in a LoWPAN.

Figure 4.1.1 illustrates where these issues typically occur in a LoWPAN. Mobility, node identification and sleep cycles are caused by node design and network properties. Intermediate 6LoWPAN Routers are a security risk, motivating end-to-end application security. The wireless link layer introduces bandwidth and frame size limitations. Finally, at the edge router we need to deal with compression, firewalls and UDP port space.

4.2 Protocol Paradigms

There is a basic set of paradigms by which most Internet application protocols function. These include the end-to-end paradigm, streaming, sessions, publish/subscribe and finally web services.

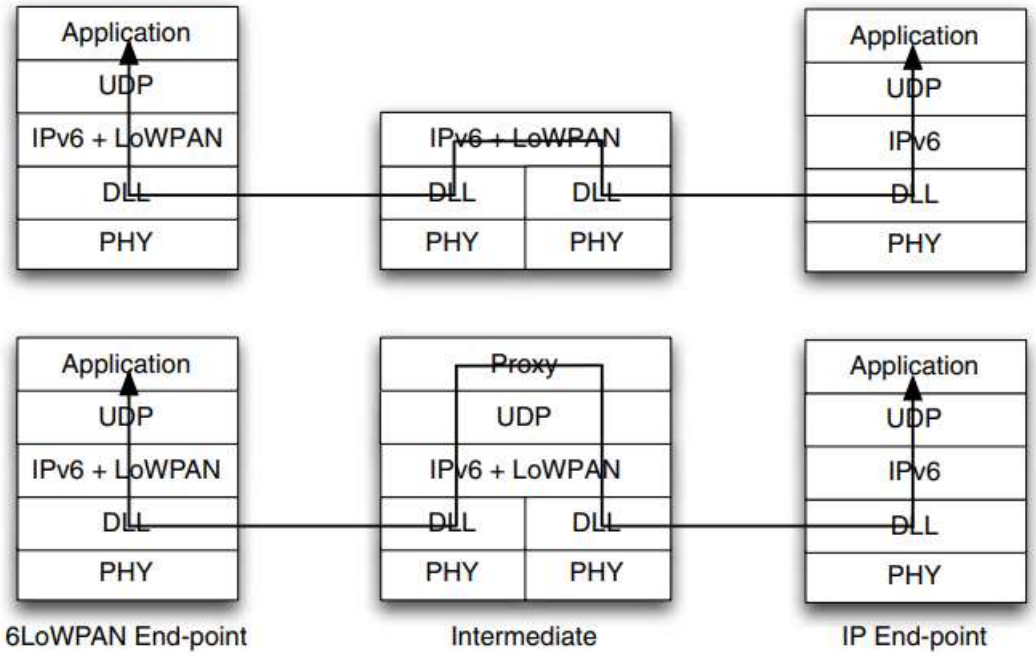


Fig 4.2.1 End-to-end (above) and proxied (below) application protocol paradigms.

- **End-to-end**: The Internet socket model is based on the use of the underlying transport layer to provide a transparent datagram or byte stream service between application processes, or so-called application end-points. When considering the application layer this can be called an end-toend paradigm where only the end-points participate in the application protocol exchanges. Some application protocols also include the possibility for intermediate nodes to inspect, cache or modify application protocols. Here we refer to this as proxying. An example would be an HTTP proxy that performs web-page caching.
- **Real-time streaming and sessions**: Internet protocols already provide a good framework for working with real-time streams, which can be employed by 6LoWPAN applications as well. The real-time transport protocol (RTP) encapsulates streams with appropriate timestamp and sequence information, while the companion RTP control protocol (RTCP) is used to control the stream. If a relationship between the sender(s) and receiver(s) of a stream needs to be automatically setup and configured, the session initiation protocol (SIP) can be employed.
- **Publish/subscribe**: Publish/subscribe (also known as pub/sub) is an asynchronous messaging paradigm in which publishers send data without knowing who the receiver is, and receivers subscribe to data based on the topic or content of the data. Pub/sub can be implemented using centralized brokers that match publishers and subscribers in a store-and-forward fashion, or in a distributed manner where subscribers filter messages directly from publishers. This decoupling of the application end-points allows for scalability and flexibility. For the Internet of Things, pub/sub plays an important role as most applications are data-centric, i.e. it is not so important who sends data, but rather what the data is. One good example of a pub/sub protocol is the MQ telemetric transport (MQTT), which is a broker-based enterprise pub/sub protocol for telemetry, used widely by IBM
- **Web service paradigms**: Web services are defined by the W3C as a software system designed to support interoperable machine-to-machine communications over a network [WS]. Web services as a whole commonly work between clients and servers over HTTP. There are two different forms of web services: service-based

(SOAP) web services and resource-based (REST) web services. Both forms of web services will play an important roll in 6LoWPAN applications.

Service-based web services use XML following the SOAP format to provide remote procedure-calls (RPCs) between clients and servers [SOAP]. These SOAP messages and sequences can be described using the web services description language (WSDL) [WSDL]. This paradigm is widely used in enterprise machine-to-machine systems. A SOAP interface is typically designed with a single URL that implements several RPCs called methods(a good analogy is a verb) as in the following example:

The representational state transfer (REST) paradigm instead models objects as HTTP resources (a good analogy is a noun), each with a URL accessible using standard HTTP methods [REST]. These interfaces can be described using the web application description language (WADL). With the release of WSDL 2.0, REST-based interfaces can alternatively be defined in a similar way to SOAP interfaces. This REST paradigm is widely used on the Internet between web sites. The content of REST HTTP messages can be of any MIME content, although XML is common in machine-to-machine applications. An example of a REST design follows, where objects are accessible using standard HTTP GET, POST, PUT and DELETE methods. In this example GET would be used on all resources to request the value, and POST would be used to set a new value for a parameter:

http://sensor10.example.com/sensors/temp http://sensor10.example.com/sensors/light
http://sensor10.example.com/sensors/acc-x http://sensor10.example.com/sensors/acc-y
http://sensor10.example.com/sensors/acc-z http://sensor10.example.com/config/sleeptime
http://sensor10.example.com/config/waketime http://sensor10.example.com/config/enabled
http://sensor10.example.com/config/samplerate

4.3 Common Protocols

This section introduces protocols that are commonly used or have good potential for use over 6LoWPAN. These include web service protocols, MQTT-S, ZigBee CAP, service discovery protocols, SNMP, RTP/RTCP, SIP and industry-specific protocols.

Web service protocols

The web service concept is hugely successful on the Internet, especially in enterprise machine-to-machine Internet systems. As many back-end systems incorporating information from LoWPAN devices will already be using existing web service principles and protocols, it is expected that 6LoWPAN will be integrated into the web service architecture. The use of XML, HTTP and TCP makes the adaptation of web services challenging for LoWPAN Nodes and networks. In this section we look inside web service protocols, and the technologies that can adapt them for use with 6LoWPAN.

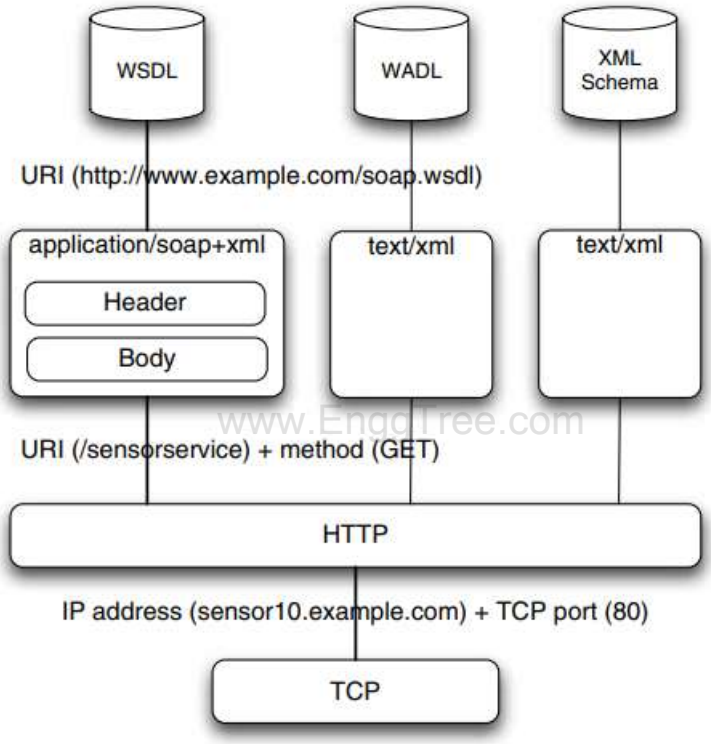


Fig 4.3.1 Typical structure of web service content over HTTP/TCP

Figure 4.3.1 shows the typical structure of web service content, which is always built upon HTTP and TCP as used today on the Internet. Web services are simply URLs available on an HTTP server with services or resources accessible behind them. In the SOAP model a URL identifies a service, e.g. /sensorservice in the figure. This service may support any number of methods (e.g. GetSensor) with corresponding responses that are described by a WSDL document. SOAP (application/soap+xml) is an XML format consisting of a header and a body, in which the body carries any number of messages.

Resource-based web services can also be realized using a REST design. Figure 5.5 also shows text/xml content used directly over HTTP. In this model formal message sequences are not used, instead each resource is identified by a URL. By using different HTTP methods on that URL, the resource can be accessed. For example sending an HTTP GET for /sensors/temp might return a text/xml body with the temperature of the sensor. REST designs make use of well-known XML or other formats to give meaning to the content that can be understood by all parties. All web services have the same basic problems for 6LoWPAN use. XML is typically too large for marking up content in the payload space available, HTTP headers have high overhead and are difficult to parse, and finally TCP has limitations of its own. A simplified HTTP header with application/soap+xml content for the example above may look like:

```
POST /sensorservice HTTP/1.1

Host: sensor10.example.com

Content-Type: application/soap+xml; charset=utf-8

Content-Length: nnn 0x1a
```

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.com/soap.wsdl">
  <m:GetSensor>
    <m:SensorID>0x1a</m:SensorID>
  </m:GetSensor>
</soap:Body>

</soap:Envelope>
```

This simple example has a length of 424 bytes, which may require up to six fragments to transmit it over a 6LoWPAN network. Next we look at different technologies to allow web services to be used with 6LoWPAN. There are two fundamental ways to integrate 6LoWPAN into a web service architecture: using a gateway approach or a compression approach.

Gateway approach: In the gateway approach, a web service gateway is implemented at the edge of the LoWPAN, often on a local server or the edge router. Inside the LoWPAN a proprietary protocol is used to request data, perform configuration etc. The gateway then makes the content and control of the devices available through a web service interface. In this approach web services actually end at the gateway. This technique is widely used with non-IP wireless embedded networks such as ZigBee and vendor-specific solutions. One downside of this approach is that a proprietary or 6LoWPAN-specific protocol is needed between nodes and the gateway. Furthermore, the gateway is dependent on the content of the application protocols. This creates scalability and evolvability problems where each time a new use of the LoWPAN is added or the application format is modified, all gateways need to be upgraded.

Compression approach: In the compression approach, the web service format and protocols are compressed to a size suitable for use over 6LoWPAN. This can be achieved using standards, and has two forms: end-to-end and proxy. In the end-to-end approach the compressed format is supported by both application end-points. In the proxy approach an intermediate node performs transparent compression so that the Internet end-point can use standard web services.

Several technologies exist for performing XML compression. The WAP Binary XML (WBXML) format was developed for mobile phone browsers [WBXML]. Binary XML (BXML) from the Open Geospatial Consortium (OGC) was designed to compress large sets of geospatial data and is currently a draft proposal [BXML]. General compression schemes like Fast Infoset (ISO/IEC 24824-1) work like zip for XML [FI]. Finally, the W3C is currently completing standardization of the efficient XML interchange (EXI) format, which performs compact binary encoding of XML [EXI]. One suitable technology for use with 6LoWPAN is the proposed EXI standard, as it supports out-of-band schema knowledge with a sufficiently compact representation. LoWPAN Nodes do not actually perform compression; instead they directly make use of the binary encoding for content, which keeps node complexity low.

XML compression alone only solves part of the problem. HTTP and TCP are still not suitable for use over 6LoWPAN. One commercial protocol solution called Nano Web Services (NanoWS) from Sensinode applies XML compression in an efficient binary transfer protocol over UDP, which has been specifically designed for 6LoWPAN use [Sensinode].

The SENSEI project is also researching the use of embedded web services inside Internetbased sensor networks [SENSEI]. The ideal long-term solution will be the standardization of a combination of XML binary encoding bound to a suitable UDP-based protocol.

The namespace and schema used with 6LoWPAN devices must also be carefully designed. Standard schemas such as SensorML from the OGC [SensorML] are often much too large and complicated for efficient use over 6LoWPAN even with compression. The most efficient result is achieved using a simple schema or one designed specifically for use with embedded devices.

1. **MQ telemetry transport for sensor networks (MQTT-S)**

The MQ telemetry transport (MQTT) is a lightweight publish/subscribe protocol designed for use in enterprise applications over low-bandwidth wide area network (WAN) links such as ISDN or GSM [MQTT]. The protocol was designed by IBM and is used in commercial products such as Websphere and Lotus, enjoying widespread use in M2M applications. MQTT uses a broker-based pub/sub architecture, to which clients publish data based on matching topic names. Subscribers then request data from the broker based on topic names. Although MQTT was designed to be lightweight, it requires the use of TCP, and the format is inefficient over 6LoWPAN networks.

In order to allow for MQTT to be used also in sensor networks, MQ telemetry transport for sensor networks (MQTT-S) was developed [MQTT-S]. This optimized protocol can be used over ZigBee, UDP/6LoWPAN or any other simple network providing a bi-directional datagram service. MQTT-S is optimized for low-bandwidth wireless networks with small frame sizes and simple devices. It is still compatible with MQTT and can be seamlessly integrated with MQTT brokers using what is called an MQTT-S gateway.

The MQTT-S architecture is shown in Figure 5.6. It is made up of four different elements: MQTT brokers, MQTT-S gateways, MQTT-S forwarders and MQTT-S clients. Clients connect themselves to a broker through a gateway using the MQTT-S protocol. The gateway may be located e.g. on the LoWPAN Edge Router, or it may be integrated in the broker itself, in which case MQTT-S messages are simply carried end-to-end over UDP. Gateways translate between MQTT-S and MQTT. In case a gateway is not directly available, forwarders are used to forward (unmodified) messages between clients and brokers. Forwarders may not be needed with 6LoWPAN as UDP datagrams can be sent directly to a gateway.

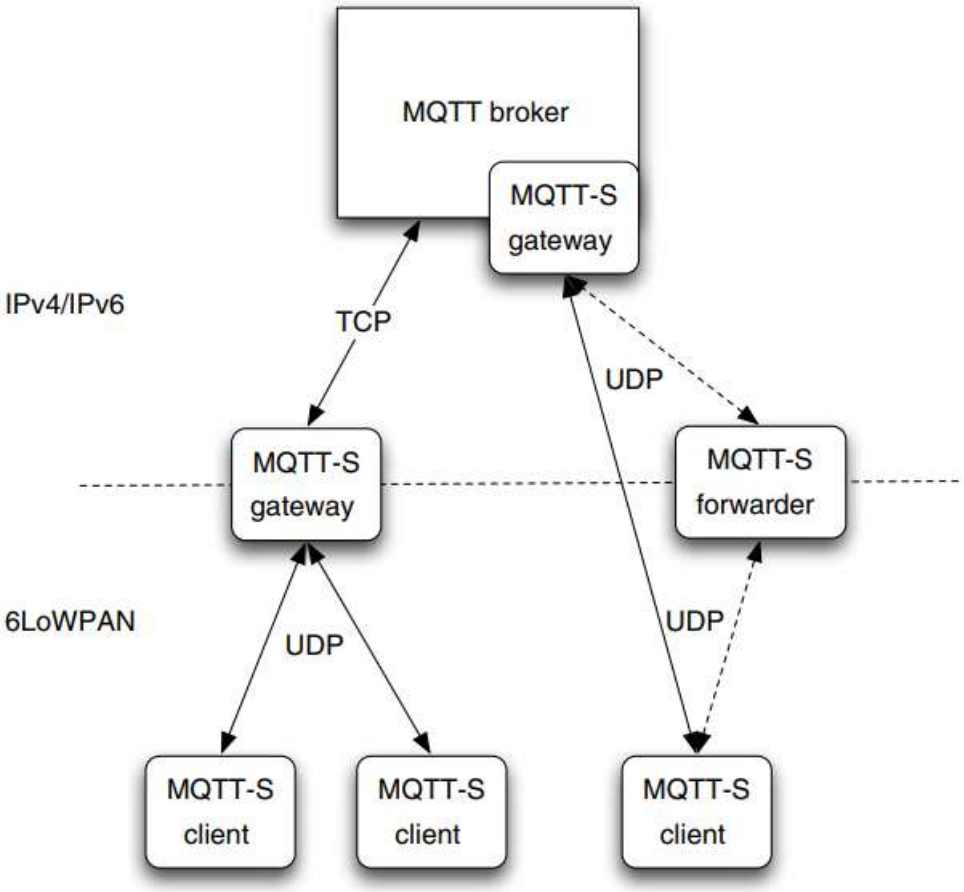


Fig 4.3.2 The MQTT-S architecture used over 6LoWPAN.

Figure 4.3.2 shows the MQTT-S message structure, which consists of a length field, a message type field and then a variable-length message part. Next the basic functionality of MQTT-S is described shortly. Readers should consult the MQTT-S specification for full protocol details [MQTT-S].

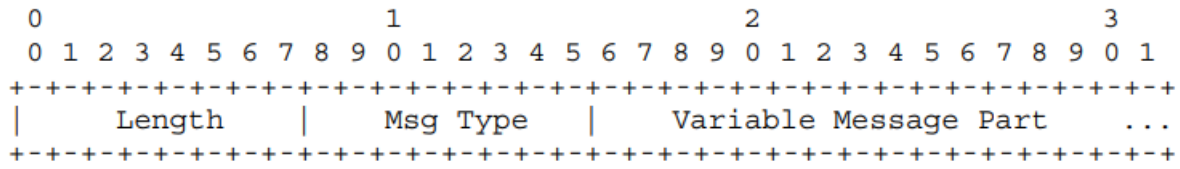


Fig 4.3.3 The MQTT-S message structure

Protocol operation: MQTT-S includes a gateway discovery procedure, which does not exist in MQTT. Alternatively clients can be pre-configured with the gateway location to avoid discovery overhead. Gateways send periodic ADVERTISE messages, and clients may send SEARCHGW messages. A GWINFO message is sent to a client in response to SEARCHGW with basic information about the gateway.

Clients CONNECT to a gateway which responds with an ACK. DISCONNECT is used to end a connection or to indicate a sleep period. Clients can connect with multiple gateways (and thus brokers) which are able to perform load balancing. Gateways can function in either transparent mode, where a connection to the broker is maintained for each client, or in aggregation mode, where the gateway aggregates messages from all clients into a single broker connection. Aggregation mode can considerably improve scalability.

MQTT-S makes use of two-byte topic IDs and short topic namesto optimize the long topic name strings normally used in MQTT. MQTT-S includes a registration message REGISTER, which explicitly indicates the topics that a client is

publishing. This reduces the bandwidth compared to MQTT where the topics and data are published in the same message. A client can send a REGISTER message with the topic name, which is acknowledged with a REGACK indicating the assigned topic ID. Topic names and IDs can also be pre-configured to avoid the need for registrations in certain cases. The client then publishes its data with PUBLISH messages including the topic ID and possible QoS information.

Clients subscribe by sending SUBSCRIBE to the gateway including the topic name of interest, which is acknowledged with SUBACK including an assigned topic ID. UNSUBSCRIBE is used to remove a subscription from the gateway.

2. ZigBee compact application protocol (CAP)

The ZigBee application layer (ZAL) [ZigBee], and the ZigBee cluster library (ZCL) [ZigBeeCL] specify an application protocol enabling interoperability between ZigBee devices at the application layer. The ZigBee Alliance maintains a series of specifications for ad hoc networking between embedded devices using a single radio, IEEE 802.15.4. Typical applications for ZigBee include home automation, energy applications and similar localarea wireless control applications. ZigBee makes use of a vertical profile approach over the ZAL and ZCL, with profiles for different industry applications such as the ZigBee home automation profile [ZigBeeHA] or the ZigBee smart energy profile [ZigBeeSE]. The ZAL and ZCL provide the key application protocol functionality in ZigBee, enabling the exchange of commands and data, service discovery, binding and security along with profile support. These protocols use compact binary formats with the goal of fitting in small IEEE 802.15.4 frames.

The ZigBee application protocol solution would have benefits used over standard UDP/IP communications as well, especially over 6LoWPAN, as the ZigBee application protocol has been designed with similar requirements. This would allow much wider use of ZigBee profiles, also end-to-end over the Internet eliminating the need for gateways. However the ZAL had been originally designed with only the ZigBee network layer primitives and IEEE 802.15.4 in mind.

A solution for using ZigBee application protocols and profiles over UDP/IP has been proposed in [ID-tolle-cap], which is an IETF Internet draft. This specification defines how the ZAL is mapped to standard UDP/IP primitives, enabling the use of any ZigBee profile over 6LoWPAN or standard IP stacks. This adaptation of the ZAL for use with UDP/IP is called the compact application protocol (CAP). The CAP protocol stack is shown in Figure 5.8. The functions of the ZAL and ZCL are implemented by the CAP. The data protocol corresponds to the ZigBee cluster library. The management protocol corresponds to the ZigBee device profile handling binding and discovery. Finally the security protocol implements ZigBee application sublayer (APS) security. Any ZigBee public or private application profile can be implemented over CAP in the same way that it would use the native ZigBee ZAL/ZCL. This allows for ZigBee application profiles to directly be applied to IP networks.

The main modification to the ZAL has to do with using IP hosts and IP addresses instead of IEEE 802.15.4 hosts and IEEE 802.15.4 addresses. ZigBee application layer messages are placed inside UDP datagrams using the CAP data protocol instead of ZigBee network layer frames. To receive unsolicited notifications CAP listens to a well-known UDP port. The ZAL identifies nodes by their 64-bit or 16-bit IEEE 802.15.4 MAC address. In CAP this is replaced by a CAP address record which can contain an IPv4 address plus UDP port, IPv6 address plus UDP port, or a fully qualified domain name plus UDP port.

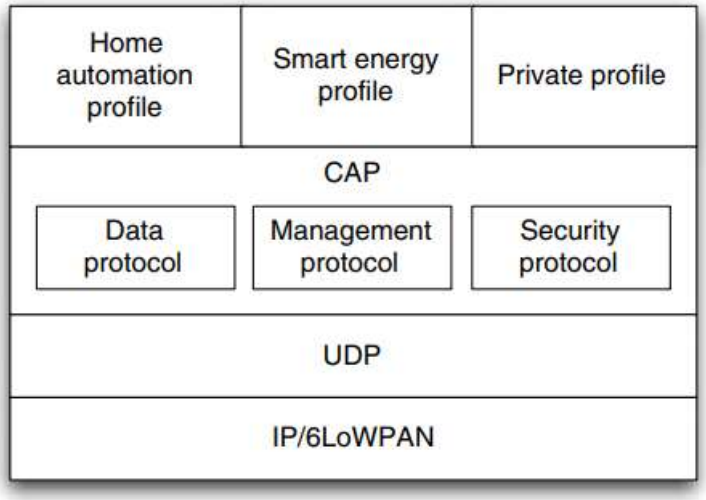


Fig 4.3.4 The CAP protocol stack.

It is assumed that the CAP is configured with the IP address and port of the discovery cache, trust center, binding coordinator and binding cache during bootstrapping. These can be configured manually, using DHCP, a special DNS entry or using a CAP server discovery message.

The CAP protocol is simply ZigBee application layer APS frames placed in UDP, with all the standard options and extensions. The APS delivery modes are mapped to IP unicast and broadcast delivery, and groupcast is reduced down to broadcast. CAP supports secure transmission and the use of APS acknowledgments, which provide limited application protocol reliability. The CAP data protocol is contained within the APS payload and it contains the ZCL command frame, with support for all ZCL command types. None of the ZCL commands require modification for use with CAP. The CAP management protocol modifies the ZigBee device profile command frame to remove IEEE 802.15.4 specific frames or to modify the address where possible. The ZigBee security and key management features are implemented by the CAP security protocol.

3. Service discovery

Service discovery is an important issue in Wireless Embedded Internet applications, where devices are autonomic – also requiring the autoconfiguration of applications. Service discovery is used to find which services are offered, what application protocol settings they use, and at what IP address they are located. Typical protocols used for service discovery on embedded devices includes the service location protocol (SLP), universal plug-n-play (UPnP) and devices profile for web services (DPWS). Some application protocols such as ZigBee CAP or MQTT-S have their own built-in discovery features. Frameworks such as OGC or SENSEI also have built-in service discovery and description mechanisms.

The service location protocol is used for general service discovery over IP networks. SLP needs optimizations in order to be effectively used with 6LoWPAN because of the size of typical messages. There has been a proposal for a simple service location protocol (SSLP) [ID-6lowpan-sslp], which provides a simple, lightweight protocol for service discovery in 6LoWPAN networks. Such a protocol could be easily interconnected with SLP running on IP networks by an SSLP translation agent located on an edge router – thus allowing 6LoWPAN services to be discovered from outside the LoWPAN and vice versa. SSLP supports most of the features of SLP, including the optional use of directory agents. The SSLP header format consists of a four-byte base header followed by specific message fields. As in SLP, service types, scopes and URLs are carried as strings. Strings used with a scheme like SSLP should be kept as short as possible.

UPnP is a protocol aimed at making home devices automatically recognizable and controllable as specified in [UPnP]. UPnP makes use of three protocols: the simple service discovery protocol (SSDP) for discovering devices, the generic event notification architecture (GENA) for event notification and SOAP for controlling devices. Devices descriptions are stored as XML and are retrieved using HTTP after initial discovery using SSDP. UPnP is not directly applicable to

6LoWPAN devices because of its dependence on broadcast along with XML- and HTTP-based descriptions and protocols. SSDP may be applicable directly over 6LoWPAN as it is similar to SSLP. It may be possible to use UPnP, or a subset of it, over 6LoWPAN with web service compression and binding applied to UPnP descriptions and protocols. However, this would require a special version of UPnP to be specified for use over 6LoWPAN and similar networks.

The devices profile for web service (DPWS) describes a basic set of functionality to enable embedded IP devices with web-service-based discovery, device descriptions, messaging and events [DPWS]. The objectives of DPWS are similar to those of UPnP, but DPWS uses a pure web-service approach. DPWS has recently been standardized under OASIS. As DPWS descriptions are XML and all messaging is based on XML/HTTP/TCP it would require web service compression and binding, along with simplification, in order to be used over 6LoWPAN. DPWS has been gaining in popularity for use in enterprise and industrial systems as devices using DPWS can be automatically integrated into back-end systems based on web services.

4. Simple network management protocol (SNMP)

Network management is an important feature of any network deployment, and a certain amount of management is necessary even for autonomous wireless embedded devices. There are several ways of performing management in IP networks, e.g. the simple network management protocol (SNMP), web services or proprietary protocols. SNMP is a standard for the management of the network infrastructure and devices in IP networks. It includes an application protocol, a database schema and data objects. The current version is SNMPv3, specified in [RFC3411]–[RFC3418]. SNMP exposes variables to a management system which can be GET or in some cases SET in order to configure or control a device. The variables exposed by SNMP are organized in hierarchies called management information bases (MIBs).

The polling approach used by SNMP (GET messages) is the biggest drawback of the approach. Polling approaches don't work for battery-powered LoWPAN Nodes which use sleep schedules, and blindly polling for statistics (which may not have changed) creates unnecessary overhead. An event-based approach would need to be added to SNMP for applicability to 6LoWPAN management.

The suitability of using SNMPv3 with 6LoWPAN has also been analyzed in [ID-snmp-optimizations], which found that optimizations are needed to reduce the packet size and memory cost. Furthermore a MIB has been specified for 6LoWPAN in [ID-6lowpan-mib]. The following optimizations for SNMPv3 have been identified in these drafts:

- Currently SNMPv3 requires the handling of payload sizes up to 484 bytes, which creates too much overhead for managing large 6LoWPANs.
- The SNMPv3 header is variable in size, and needs to be optimized for 6LoWPAN. Only a minimal subset of functionality should be supported and the header size should be limited.
- The binary encoding rules (BER) of the payload use variable length fields. For 6LoWPAN, fixed length fields or more compact encoding may be necessary.
- Payload compression and aggregation may be needed for 6LoWPAN.
- To reduce memory requirements the maximum size of SNMP messages should be limited. Furthermore MIB support should be carefully chosen.

5. Real-time transport and sessions

The real-time transport protocol (RTP) [RFC3550] is used for the end-to-end delivery of real-time data. RTP is designed to be IP-version- and transport-independent, and can be used over both UDP and TCP. The base RTP header provides basic features for end-toend delivery: payload-type identification, a sequence number and a timestamp. The RTP header format is shown in Figure 5.9. RTP does not by itself provide any kind of QoS, but it is able to help deal with out-of-order packets and jitter with the sequence number and timestamp fields, respectively. The accompanying real-time control protocol (RTCP) is used during an RTP session to provide feedback on the QoS of RTP data delivery, to identify the RTP source, adjust the RTCP report interval and to carry session control information. RTP uses the concept of

profiles, which define possible additional headers, features and payload formats for a particular class of application. The RTP audio video profile (AVP) specifies the profile for common audio and video applications.

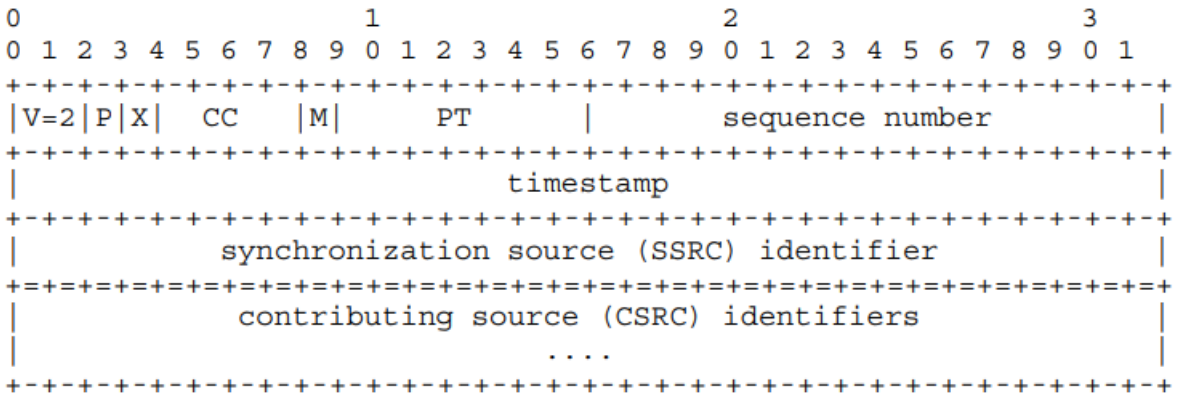


Fig 4.3.5 The RTP base header.

RTP is applicable also over 6LoWPAN for the transport of real-time data. As RTP makes use of UDP, is IP-version-independent and has a fairly compact header format, it is directly usable without modification. Existing profiles such as AVP are useful for the streaming of low-rate audio and video using the most efficient codecs, and custom profiles can be created for the transmission of e.g. sensor data streams.

Although RTP can be used to delivery and monitor real-time data streams, it requires that the sender and receiver somehow know about and find each other. Although this may be the case in specialized embedded applications of RTP over 6LoWPAN, the automatic negotiations of real-time sessions or other messaging may be very useful. The session initiation protocol (SIP) [RFC3261] was designed for establishing, modifying and tearing down multimedia sessions over IP. SIP is widely used for Voice-over-IP (VoIP) applications, and forms the backbone for the IP Multimedia System (IMS) upon which future cellular.

services will be built. The SIP design is similar to that of HTTP, in that it uses a humanreadable header format. SIP can be used over either UDP or TCP, can be handled by intermediate proxies, and provides identifiers for dealing with mobility. SIP exchanges are typically performed between SIP user agents (e.g. embedded devices) and servers. Typical methods include REGISTER, INVITE, ACK and BYE. SIP uses a separate session description protocol (SDP) to negotiate media types. An example SIP header for an INVITE from is shown below:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```

The SIP header and body format is typically too large for efficient use over 6LoWPAN. But as SIP can be applied to session setup, alarms, events and IMS integration, it has valuable use in low-power embedded networks. One solution for using SIP with sensor networks is TinySIP, which defined alternative messages for use with TinyOS networking that were then mapped to SIP by a gateway application [TinySIP].

6. Industry-specific protocols

This section gives an overview of industry-specific application protocols that can be used over IP, and are relevant for Wireless Embedded Internet applications using 6LoWPAN. Building automation and energy are good examples of industries that have traditionally specified their own application protocols and formats. These are enterprise applications where system integrators make use of equipment from multiple vendors together with backend computer systems to achieve large deployments. The need for common application protocols and formats is obvious in such an environment. As communication technology has evolved, industry-specific protocols have steadily evolved to enable use over IP. Many industry-specific protocols may be used over 6LoWPAN, whereas others may require the addition of compression, IPv6 support or UDP support for example. Next we examine some common building automation and energy industry application protocol standards.

BACnet: BACnet is a network and application protocol format with support for a wide range of communication technologies including Ethernet, RS232, RS-485 and LonTalk. BACnet includes support for use over UDP/IP known as BACnet/IP. The standard BACnet network and application protocol frames are carried over UDP by encapsulating them in a BACnet virtual link layer (BVLL). This adaptation binds BACnet to an underlying communication technology. Currently there is only a BVLL defined for use with IPv4, but an extension of that for IPv6 is straightforward. IPv6 support is current under design in the BACnet IP working group. BACnet makes use of unicast, broadcast and optionally multicast IP communications, and is based on an object-oriented design. BACnet objects have properties that are acted upon using protocol services. These protocol services include Who-Is, I-Am, Who-Has and I-Have used for device and object discovery along with read-property and write-property used for data access.

As BACnet was designed for a whole range of low-bandwidth links, and has native support for IPv4, it will be a useful protocol over 6LoWPAN for building automation applications. The adaptation of BACnet/IP for use with 6LoWPAN will require IPv6 support, and should make careful use of multicast – keeping in mind its overhead on wireless multihop mesh networks. The performance and possible optimization of BACnet service protocol traffic over low-power wireless mesh networks should also be studied as BACnet currently assumes wired links.

KNX: The KNX protocol supports several different communication media: twisted pair, powerline, radio frequency (RF) and IP. Twisted pair is the most common KNX medium and is typically installed when a building or house is constructed. Powerline and RF are often used when retrofitting existing buildings. The KNX RF specification uses its own framing over an 868 MHz radio at 16 kbit/s. KNX networks can theoretically support up to 64k devices using twisted pair, power-line or RF communications.

KNX has some support for IP, also known as KNXnet/IP. KNX IP support is part of a framework called ANubis (advanced network for unified building integration and services) and among many other things provides a way to encapsulate KNX frames over IP. The purpose of this is currently to interconnect KNX networks using IP networks, to enable remote monitoring and to interconnect with other systems such as BACnet. This IP encapsulation technique could also be usefully applied over low-power IP and 6LoWPAN networks to KNX devices themselves. This would need a considerable amount of development and standardization.

oBIX: oBIX provides a web service interface, which can be used to interact with any building automation network including BACnet, KNX, Modbus, Lontalk or proprietary networks using the oBIX XML format. The format provides normalized representation of constructs common to building automation protocols: points (scalar value, status), alarms and histories. It has an extensible meta-format which can be used to describe any system. oBIX provides a low-level object model for working with these constructs. Usually these are accessed by using generic oBIX constructs, for example by an enterprise developer.

oBIX is web service binding agnostic. It can be used over both SOAP and directly over HTTP in a REST style. It represents objects with URLs and object state with XML. oBIX may be applicable also for use directly in building

automation networks thanks to 6LoWPAN. Instead of running a control network specific building automation protocol such as BACnet/IP or KNX over 6LoWPAN, oBIX together with compression and UDP/IP binding may be a solution. Careful design of the oBIX objects and elements used would be important to keep packet sizes reasonable.

ANSI C12.19

The ANSI C12.18 standard defines a point-to-point optical interface along with the protocol specification for electric metering (PSEM). The ANSI C12.19 standard defines utility industry end device data tables, which are accessible using PSEM. ANSI C12.21 specifies a communication protocol over modem lines. Finally a new standard ANSI C12.22 specified the interfacing of devices to any data communications network including Internet protocols. This furthermore specifies the extended protocol specification for advanced metering (EPSEM).

The ANSI C12 PSEM protocol and device data formats (tables) were designed with simple embedded electric meters and low-bandwidth point-to-point links in mind. Therefore PSEM uses a compact binary encoding for all its protocol and data fields. The typical frame size expected by point-to-point links is 64 bytes in the ANSI C12.22 specification. The ANSI C12.22 specification is therefore well suited for use over 6LoWPAN. The specification includes a simple example of using the application layers with a TCP/IPv4 communication stack. The standard may also be used with a UDP/IPv6 stack, as the protocol includes acknowledgment features for reliability along with application-layer segmentation and reassembly. Application layer security features are built in elements of the protocol.

APPLICATION PROTOCOLS

DLMS/COSEM

The device language message specification (DLMS) is a European model of communication exchange used to interact with utility end devices for meter reading, tariff and load control. It uses the companion specification for energy metering (COSEM) as a data exchange format and protocol. Together, they are standardized by the IEC under the 62056 series [IEC62056]. The DLMS Association [DLMS] actively promotes the maintenance and use of the standards. IEC 62056 is similar in function to its American counterpart, ANSI C12. It makes use of local optical or current loops to meters, point-to-point serial modems, or IP networks for communication. The COSEM protocol defines the application layer in IEC 62056-53. As with ANSI C12, an object model is used to access information as defined in IEC 62056- 61. The use of COSEM over IPv4 is described in detail in IEC 62056-47. This specification describes transport using both UDP and TCP, which enables use over 6LoWPAN. Although the specification describes the use of IPv4, the changes needed for IPv6 are minor. It is unclear if any changes would be needed to the standard to allow the use of IPv6. Even though the data representation format is not as compact as in ANSI C12, it is still suitable for the frame size and bandwidth limitations of 6LoWPAN.

5.1 Operating System: Tiny OS

- Tiny OS aims at supporting sensor network applications on resource-constrained hardware platforms
- To ensure that an application code has an extremely small footprint.
- Tiny OS chooses to have no file system, supports only static memory allocation, implements a simple task model, and provides minimal device and networking abstractions.
- To a certain extent, each Tiny OS application is built into the operating system.
- Like many operating systems, Tiny OS organizes components into layers.
- Intuitively, the lower a layer is, the "closer" it is to the hardware; the higher a layer is, the "closer" it is to the application.
- In addition to the layers, Tiny OS has a unique component architecture and provides as a library a set of system software components.
- A component specification is independent of the component implementation.
- Although most components encapsulate software functionalities, some are just thin wrappers around hardware.

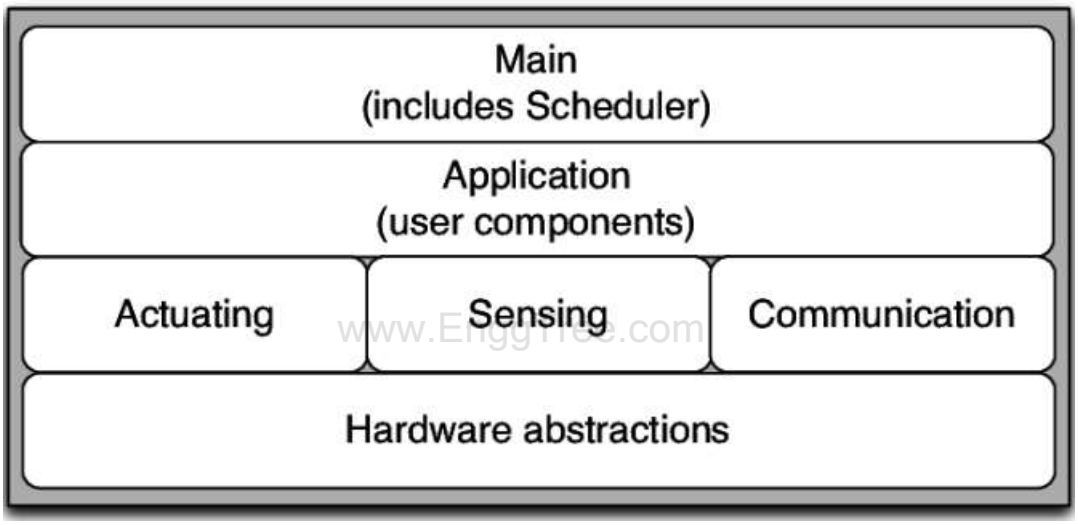


Fig: 5.1.1 Tiny OS Architecture

- An application, typically developed in the nesC language covered in the next section, wires these components together with other application-specific components.
- Let us consider a Tiny OS application example—Field Monitor, where all nodes in a sensor field periodically send their temperature and photosensor readings to a base station via an ad hoc routing mechanism.
- A diagram of the Field Monitor application is shown in above Figure, where blocks represent Tiny OS components and arrows represent function calls among them.
- The directions of the arrows are from callers to callees
- Hardware abstraction components are the lowest-level components.
- They are actually the mapping of physical hardware such as I/O devices, a radio transceiver, and sensors. Each component is mapped to a certain hardware abstraction.
- Synthetic hardware components are used to map the behaviour of advanced hardware and often sit on the hardware abstraction components.
- Tinyos designs a hardware abstract component called the Radio-Frequency Module(RFM) for the radio transceiver and a synthetic hardware component called radio byte, which handles data into or out of the underlying RFM.
- Higher-level components encapsulate software functionality, but with a similar abstraction.

- They provide commands, signal events and have internal handlers task threads and state variables.

Advantages of TinyOS

1. It requires very little code and a small amount of data.
2. Events are propagated quickly and the rate of posting a task and switching the corresponding context is very high
3. It enjoys efficient modularity.

NesC language.

- nesC is an extension of C to support and reflect the design of TinyOS v1.0 and above.
- It provides a set of language constructs and restrictions to implement TinyOS components and applications.

Component Interface

- A component in nesC has an interface specification and an implementation.
- To reflect the layered structure of TinyOS, interfaces of a nesC component are classified as provides or uses interfaces.
- A provides interface is a set of method calls exposed to the upper layers, while a uses interface is a set of method calls hiding the lower layer components.
- Methods in the interfaces can be grouped and named.
- For example, the interface specification of the Timer component in above Figure is listed in below Figure.
- The interface, again, independent of the implementation, is called Timer Module.
- Although they have the same method call semantics, nesC distinguishes the directions of the interface calls between layers as event calls and command calls.
- An event call is a method call from a lower layer component to a higher layer component, while a command is the opposite.
- Note that one needs to know both the type of the interface (provides or uses) and the direction of the method call (event or command) to know exactly whether an interface method is implemented by the component or is required by the component.
- The separation of interface type definitions from how they are used in the components promotes the reusability of standard interfaces.
- A component can provide and use the same interface type, so that it can act as a filter interposed between a client and a service.
- A component may even use or provide the same interface multiple times.
- In these cases, the component must give each interface instance a separate name, as shown in the Clock interface in Figure.

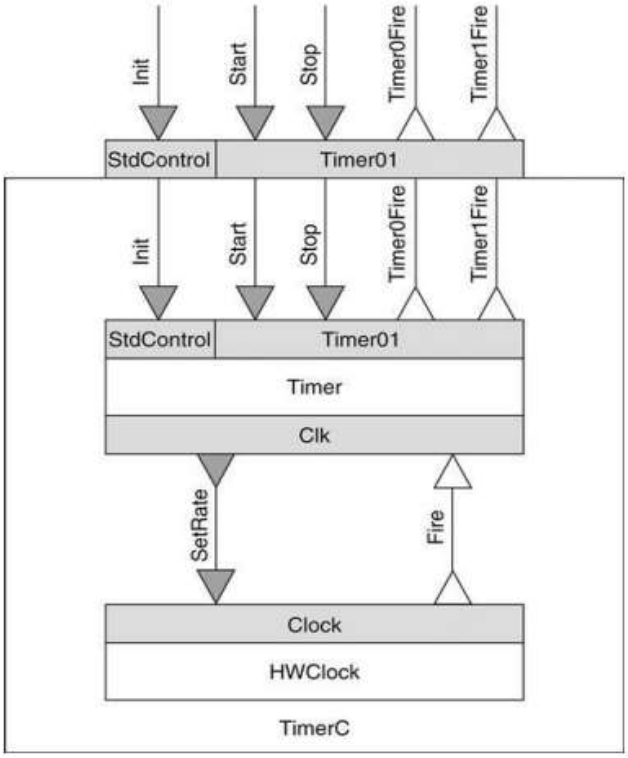


Fig: The TimerC configuration implemented by connecting Timer with HWClock.

Component Implementation

- There are two types of components in nesC depending on how they are implemented: modules and configurations.
- Modules are implemented by application code (written in a C-like syntax). Configurations are implemented by connecting interfaces of existing components.
- The implementation part of a module is written in C-like code.
- A command or an event bar in an interface foo is referred as foo.bar.
- A keyword call indicates the invocation of a command.
- A keyword signal indicates the triggering by an event. For example, the above Figure shows part of the implementation of the Timer component, whose interface is defined in Figure.
- In a sense, this implementation is very much like an object in object-oriented programming without any constructors.
- Configuration is another kind of implementation of components, obtained by connecting existing components.
- Suppose we want to connect the Timer component and a hardware clock wrapper, called HWClock, to provide a timer service, called Timer C.
- A conceptual diagram of how the components are connected and Figure a. shows the corresponding nesC code.
- In the implementation section of the configuration, the code first includes the two components and then specifies that the interface StdControl of the TimerC component is the StdControl interface of the TimerModule; similarly for the Timer01 interface.
- The connection between the Clock interfaces is specified using the -> operator.
- Essentially, this interface is hidden from upper layers.
- nesC also supports the creation of several instances of a component by declaring abstract components with optional parameters.
- Abstract components are created at compile time in configurations.

- TinyOS does not support dynamic memory allocation, so all components are statically constructed at compile time.
- A complete application is always a configuration rather than a module.
- An application must contain the Main module, which links the code to the scheduler at run time.
- The Main has a single StdControl interface, which is the ultimate source of initialization of all components.

Concurrency and Atomicity

- The language nesC directly reflects the TinyOS execution model through the notion of command and event contexts.
- The SenseAndSend component is intended to be built on top of the Timer , an analog-to-digital conversion (ADC) component, which can provide sensor readings, and a communication component, which can send a packet.
- When responding to a timer0Fire event, the SenseAndSend component invokes the ADC to poll a sensor reading. Since polling a sensor reading can take a long time, a split-phase operation is implemented for getting sensor readings.
- The call to ADC.getData() returns immediately, and the completion of the operation is signaled by an ADC.dataReady() event.
- A busy flag is used to explicitly reject new requests while the ADC is fulfilling an existing request. The ADC.getData() method sets the flag to true, while the ADC.dataReady() method sets it back to false.
- Sending the sensor reading to the next-hop neighbor via wireless communication is also a long operation.
- To make sure that it does not block the processing of the ADC.dataReady() event, a separate task is posted to the scheduler.
- A task is a method defined using the task keyword. In order to simplify the data structures inside the scheduler, a task cannot have arguments.
- Thus the sensor reading to be sent is put into a sensor Reading variable.
- There is one source of race condition in the Sense And Send, which is the updating of the busy flag.
- To prevent some state from being updated by both scheduled tasks and event-triggered interrupt handlers, nesC provides language facilities to limit the race conditions among these operations.
- In nesC, code can be classified into two types:
    - Asynchronous code (AC): Code that is reachable from at least one interrupt handler.
    - Synchronous code (SC): Code that is only reachable from tasks.
- Because the execution of TinyOS tasks are non pre-emptive and interrupt handlers pre-empt tasks, an SC is always atomic with respect to other SCs.
- However, any update to shared state from AC, or from SC that is also updated from AC, is a potential race condition.
- To reinstate atomicity of updating shared state, nesC provides a keyword atomic to indicate that the execution of a block of statements should not be preempted.
- This construction can be efficiently implemented by turning OFF hardware interrupts.
- To prevent blocking the interrupts for too long and affecting the responsiveness of the node, nesC does not allow method calls in atomic blocks.
- In fact, nesC has a compiler rule to enforce the accessing of shared variables to maintain the racefree condition
- If a variable x is accessed by an AC, then any access of x outside of an atomic statement is a compile-time error.
- This rule may be too rigid in reality.
- When a programmer knows for sure that a data race is not going to occur, or does not care if it occurs, then a no race declaration of the variable can prevent the compiler from checking the race condition on that variable.
- Thus, to correctly handle concurrency, nesC programmers need to have a clear idea of what is synchronous code and what is asynchronous code.
- However, since the semantics is hidden away in the layered structure of TinyOS, it is sometimes not obvious to the programmers where to add atomic blocks.

5.2 SENSOR NETWORK SIMULATORS

- Node-level design methodologies are usually associated with simulators that simulate the behavior of a sensor network on a per-node basis.
- Using simulation, designers can quickly study the performance (in terms of timing, power, bandwidth, and scalability) of potential algorithms without implementing them on actual hardware and dealing with the vagaries of actual physical phenomena.

A node-level simulator typically has the following components:

Sensor node model:

- A node in a simulator acts as a software execution platform, a sensor host, as well as a communication terminal.
- In order for designers to focus on the application-level code, a node model typically provides or simulates a communication protocol stack, sensor behaviours (e.g., sensing noise), and operating system services.
- If the nodes are mobile then the positions and motion properties of the nodes need to be modelled.
- If energy characteristics are part of the design considerations, then the power consumption of the nodes needs to be modelled.

Communication model:

- Depending on the details of modelling, communication may be captured at different layers.
- The most elaborate simulators model the communication media at the physical layer, simulating the RF propagation delay and collision of simultaneous transmissions.
- Alternately, the communication may be simulated at the MAC layer or network layer, using, e.g., stochastic processes to represent low-level behaviours.

Physical environment model:

- A key element of the environment within which a sensor network operates is the physical phenomenon of interest.
- The environment can also be simulated at various levels of detail.
- For example, a moving object in the physical world may be abstracted into a point signal source.
- The motion of the point signal source may be modeled by differential equations or interpolated from a trajectory profile.
- If the sensor network is passive—i.e., it does not impact the behavior of the environment—then the environment can be simulated separately or can even be stored in data files for sensor nodes to read in.
- If, in addition to sensing, the network also performs actions that influence the behavior of the environment, then a more tightly integrated simulation mechanism is required.

Statistics and visualization:

- The simulation results need to be collected for analysis.
- Since the goal of a simulation is typically to derive global properties from the execution of individual nodes, visualizing global behaviors is extremely important.
- An ideal visualization tool should allow users to easily observe on demand the spatial distribution and mobility of the nodes, the connectivity among nodes, link qualities, end-to-end communication routes and delays, phenomena and their spatio-temporal dynamics, sensor readings on each node, sensor node states, and node lifetime parameters (e.g., battery power).
- A sensor network simulator simulates the behavior of a subset of the sensor nodes with respect to time.
- Depending on how the time is advanced in the simulation, there are two types of execution models: cycle-driven (CD) simulation and discrete-event (DE) simulation.

Cycle-driven (CD) simulation and discrete-event (DE) simulation.

- A CD simulation discretizes the continuous notion of real time into (typically regularly spaced) ticks and simulates the system behavior at these ticks.
- At each tick, the physical phenomena are first simulated, and then all nodes are checked to see if they have anything to sense, process, or communicate. Sensing and computation are assumed to be finished before the next tick.
- Sending a packet is also assumed to be completed by then. However, the packet will not be available for the destination node until the next tick.
- This split-phase communication is a key mechanism to reduce cyclic dependencies that may occur in CD simulations.
- That is, there should be no two components such that one of them computes yk = f(xk) and the other computes xk = g(yk) for the same tick index k.
- In fact, one of the most subtle issues in designing a CD simulator is how to detect and deal with cyclic dependencies among nodes or algorithm components.
- Most CD simulators do not allow interdependencies within a single tick.
  Synchronous languages, which are typically used in control system designs rather than sensor network designs, do allow cyclic dependencies.

- They use a fixed-point semantics to define the behavior of a system at each tick.
- Unlike CD simulators, DE simulators assume that the time is continuous and an event may occur at any time.
- An event is a 2-tuple with a value and a time stamp indicating when the event is supposed to be handled. Components in a DE simulation react to input events and produce output events.
- In node-level simulators, a component can be a sensor node and the events can be communication packets; or a component can be a software module within a node and the events can be message passings among these modules.
- Typically, components are causal, in the sense that if an output event is computed from an input event, then the time stamp of the output event should not be earlier than that of the input event.
- Noncausal components require the simulators to be able to roll back in time, and, worse, they may not define a deterministic behavior of a system .
- A DE simulator typically requires a global event queue. All events passing between nodes or modules are put in the event queue and sorted according to their chronological order.
- At each iteration of the simulation, the simulator removes the first event (the one with the earliest time stamp) from the queue and triggers the component that reacts to that event. ϖ In terms of timing behavior, a DE simulator is more accurate than a CD simulator, and, as a consequence, DE simulators run slower.
- The overhead of ordering all events and computation, in addition to the values and time stamps of events, usually dominates the computation time.
- At an early stage of a design when only the asymptotic behaviors rather than timing properties are of concern, CD simulations usually require less complex components and give faster simulations.
- Partly because of the approximate timing behaviors, which make simulation results less comparable from application to application, there is no general CD simulator that fits all sensor network simulation tasks.
- We have come across a number of homegrown simulators written in Matlab, Java, and C++. Many of them are developed for particular applications and exploit application-specific assumptions to gain efficiency.

- DE simulations are sometimes considered as good as actual implementations, because of their continuous notion of time and discrete notion of events.
- There are several open-source or commercial simulators available. One class of these simulators comprises extensions of classical network simulators, such as ns-2, J-Sim (previously known as JavaSim), and GloMoSim/QualNet.
- The focus of these simulators is on network modeling, protocols stacks, and simulation performance.
- Another class of simulators, sometimes called software-in- the-loop simulators, incorporate the actual node software into the simulation.
- For this reason, they are typically attached to particular hardware platforms and are less portable. Examples include TOSSIM for Berkeley motes and Em* (pronounced em star) for Linux-based nodes such as Sensoria WINS NG platforms.

5.3 THE SIMULATOR TOSSIM

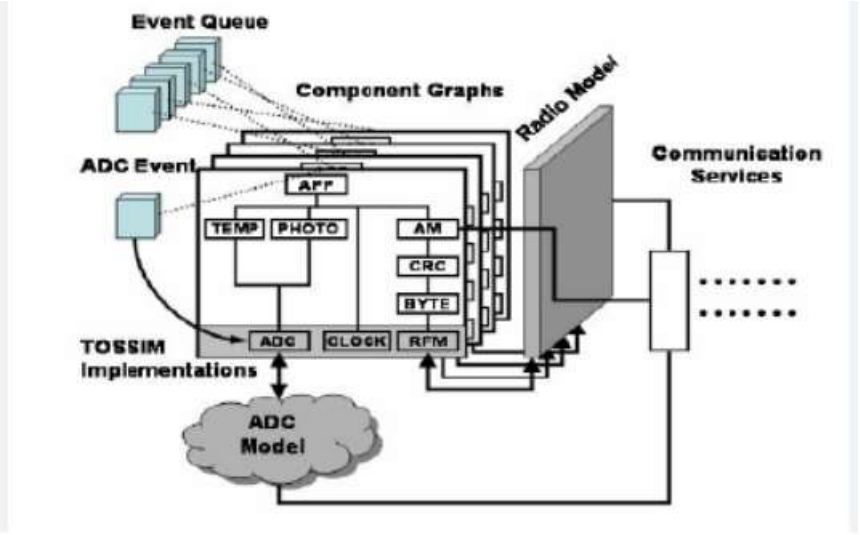


Fig: TOSSIM Architecture

- TOSSIM is a dedicated simulator for TinyOS applications running on one or more Berkeley motes.
- The key design decisions on building TOSSIM were to make it scalable to a network of potentially thousands of nodes, and to be able to use the actual software code in the simulation.
- To achieve these goals, TOSSIM takes a cross-compilation approach that compiles the nesC source code into components in the simulation.
- The event-driven execution model of TinyOS greatly simplifies the design of TOSSIM. By replacing a few low-level components, such as the ADC, the system clock, and the radio front end, TOSSIM translates hardware interrupts into discrete-event simulator events.
- The simulator event queue delivers the interrupts that drive the execution of a node.
- The upperLayer TinyOS code runs unchanged.
- TOSSIM uses a simple but powerful abstraction to model a wireless network.
- A network is a directed graph, where each vertex is a sensor node and each directed edge has a biterror rate.
- Each node has a private piece of state representing what it hears on the radio channel.
- By setting connections among the vertices in the graph and a bit-error rate on each connection, wireless channel characteristics, such as imperfect channels, hidden terminal problems, and asymmetric links, can be easily modeled. Wireless transmissions are simulated at the bit level.
- If a bit error occurs, the simulator flips the bit.
- TOSSIM has a visualization package called TinyViz, which is a Java application that can connect to TOSSIM simulations.
- TinyViz also provides mechanisms to control a running simulation by, e.g., modifying ADC readings, changing channel properties, and injecting packets.
- TinyViz is designed as a communication service that interacts with the TOSSIM event queue.
- The exact visual interface takes the form of plug-ins that can interpret TOSSIM events.
- Beside the default visual interfaces, users can add application-specific ones easily.

COOJA SIMULATOR

- Cooja is an emulator

- According to different sources, an emulator is:
  - ✓ a hardware or software system that enables one computer system (called the host) to behave like another computer system (called the guest): e.g. Cooja enabling your laptop to behave like a Z1 mote.
  - ✓ a system that typically enables the host system to run software or use peripheral devices designed for the guest system: e.g. Cooja enabling your laptop to run the RPL protocol, LIBP and/or other IoT protocols of interest.

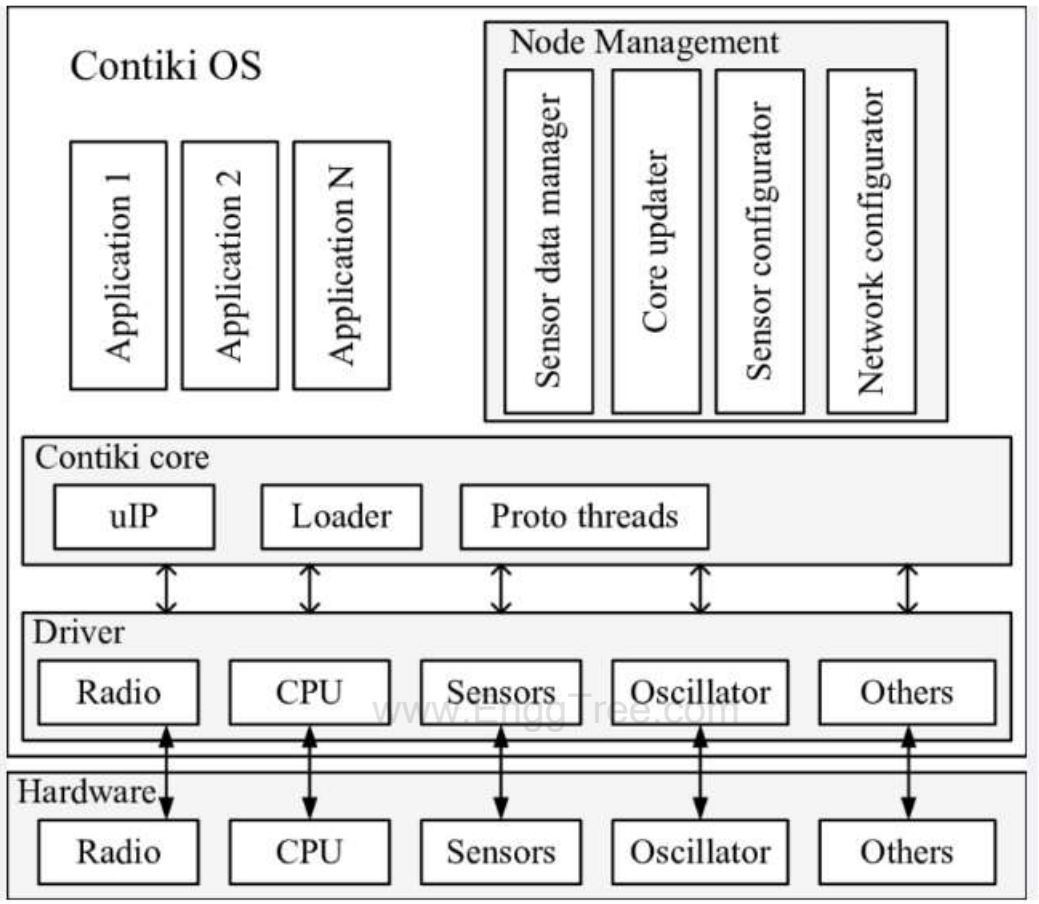


Fig: Architecture of Contiki

- Cooja is not a simulator
- According to different sources, a simulator is:
  - ✓ a hardware or software that that enables one computer system (called the host) to behave like another computer system (called the guest), but is implemented in an entirely different way :
  - ✓ e.g. A flight simulator gives you the feeling of flying an airplane, but you are completely disconnected from the reality of flying the plane, and you can bend or break those rules as you see fit.
  - ✓ e.g. Fly an Airbus A380 upside down between London and Sydney without breaking it.
- Cooja is a Contiki network emulator
  - ✓ An extensible Java-based simulator capable of emulating Tmote Sky (and other) nodes
- The code to be executed by the node is the exact same firmware you may upload to physical nodes
- Allows large and small networks of motes to be simulated ϒ Motes can be emulated at the hardware level
  - ✓ Slower but allows for precise inspection of system behavior ϒ Motes can also be emulated at a less detailed level
  - ✓ Faster and allows simulation of larger networks
- Cooja is a highly useful tool for Contiki development

- It allows developers to test their code and systems long before running it on the target hardware
- Developers regularly set up new simulations, to debug their software , to verify the behavior of their systems