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.

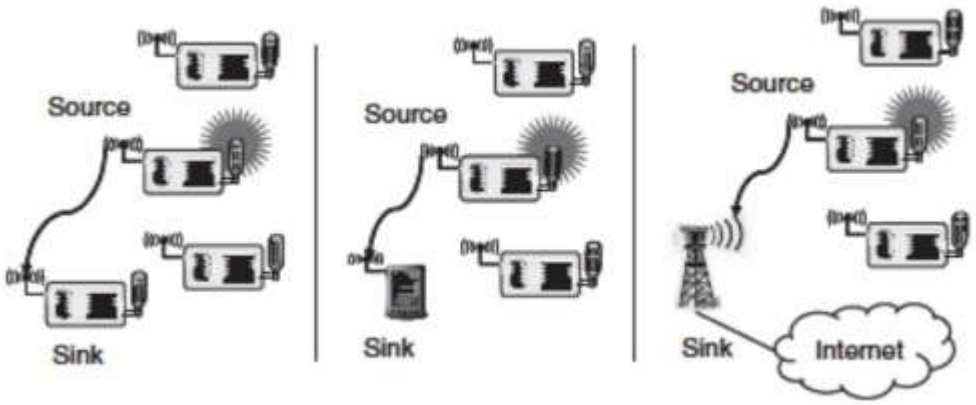

Fig: 1.3.1 Three types of sinks in a very simple, single-hop sensor network

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).

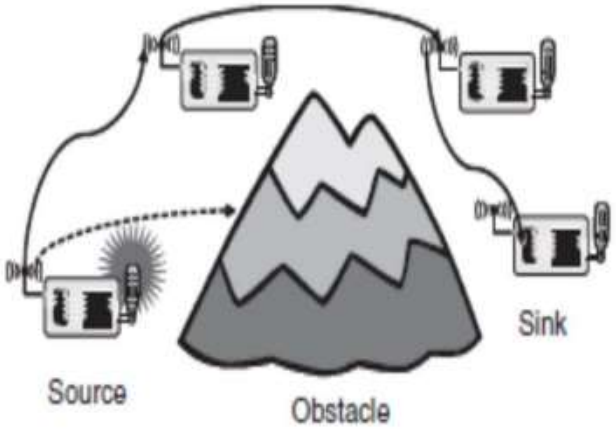

Fig: 1.3.2 Multihop networks: As direct communication is impossible because of distance and obstacles, multihop communication can circumvent the problem

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.

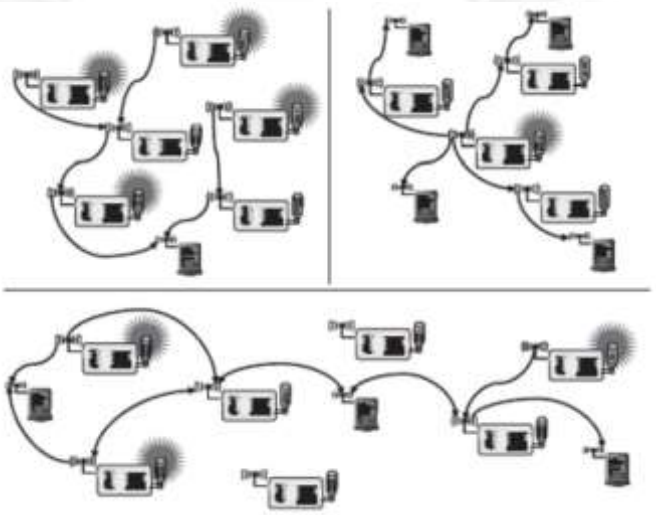

Fig 1.3.3 Multiple sources and multiple sinks

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.

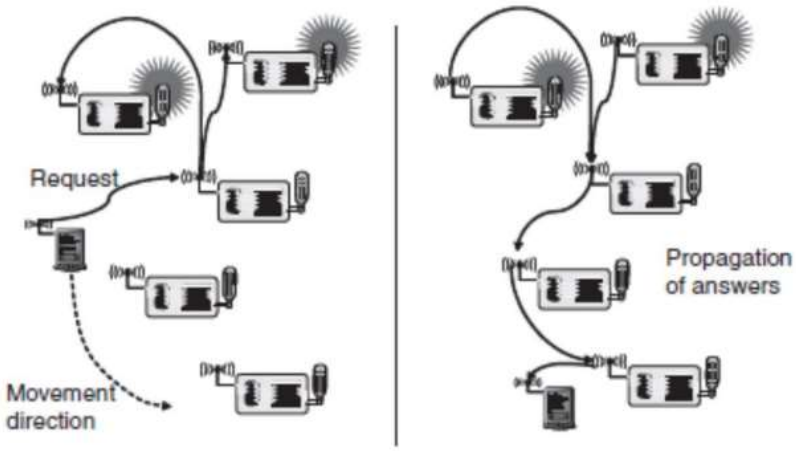

Fig 1.3.4 A mobile sink moves through a sensor network as information is being retrieved on its behalf

- 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.

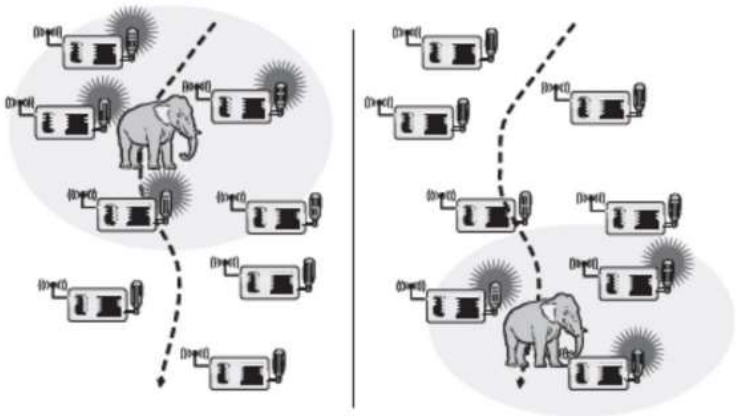

Fig 1.3.5 Area of sensor nodes detecting an event

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"

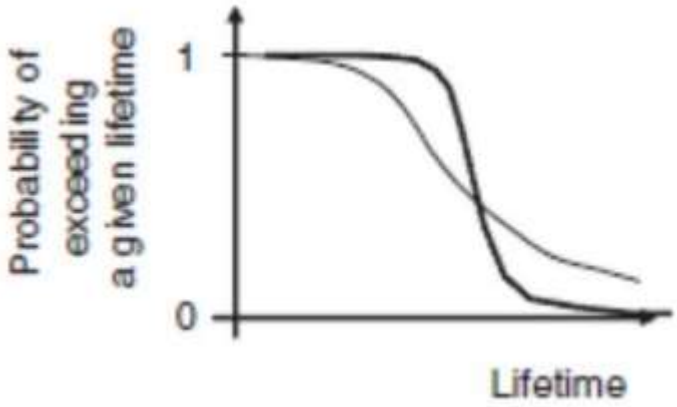

Fig: 1.3.6 Two probability curves of a node exceeding a given lifetime

- **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.

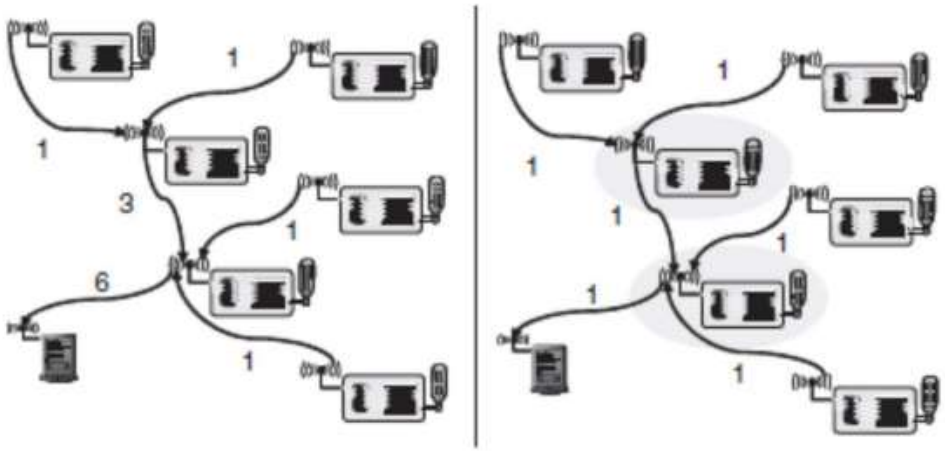

Fig : 1.3.7 Aggregation Example

- **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