Technical Report

UCAM-CL-TR-646
ISSN 1476-2986

Number 646

A survey of Wireless Sensor Network technologies: research trends and middleware's role

Eiko Yoneki, Jean Bacon

September 2005

ISSN 1476-2986

A survey of Wireless Sensor Network technologies: research trends and middleware's role

Eiko Yoneki and Jean Bacon
University of Cambridge Computer Laboratory
Cambridge CB3 0FD, United Kingdom
{eiko.yoneki, jean.bacon}@cl.cam.ac.uk

ABSTRACT

Wireless Sensor Networks (WSNs) provide a new paradigm for sensing and disseminating information from various environments, with the potential to serve many and diverse applications. Current WSNs typically communicate directly with a centralized controller or satellite. On the other hand, a smart WSN consists of a number of sensors spread across a geographical area; each sensor has wireless communication capability and sufficient intelligence for signal processing and networking of the data. The structure of WSNs are tightly application-dependent, and many services are also dependent on application semantics (e.g. application-specific data processing combined with data routing). Thus, there is no single typical WSN application, and dependency on applications is higher than in traditional distributed applications. The application/middleware layer must provide functions that create effective new capabilities for efficient extraction, manipulation, transport, and representation of information derived from sensor data. In this paper, we report recent trends in wireless sensor network research including an overview of the various categories of WSN, a survey of WSN technologies and a discussion of existing research prototypes and industry applications. We focus on middleware technology, and describe details of some existing research prototypes, then address challenges and future perspectives on the middleware. This study highlights that middleware needs to provide a common interface for various functional components of WSN: detection and data collection, signal processing, data aggregation, and notification. By integrating sensing, signal processing, and communication functions, a WSN provides a natural platform for hierarchical information processing.

1. INTRODUCTION

The need to monitor and measure various physical phenomena (e.g. temperature, fluid levels, vibration, strain, humidity, acidity, pumps, generators to manufacturing lines, aviation, building maintenance and so forth) is common to many areas including structural engineering, agriculture and forestry, healthcare, logistics and transportation, and military applications. Wired sensor networks have long been used to support such environments and, until recently, wireless sensors have been used only when a wired infrastructure is infeasible, such as in remote and hostile locations. But the cost of installing, terminating, testing, maintaining, trouble-shooting, and upgrading a wired network makes wireless systems potentially attractive alternatives for general scenarios.

Recent advances in technology have made possible the production of intelligent, autonomous, and energy efficient sensors that can be deployed in large numbers to form self organizing and self healing WSNs in a geographical area. Moreover, the dramatic reduction in the cost of this wireless sensor technology has made its widespread deployment feasible, and the urgent need for research into all aspects of WSNs has become evident. The WSN has great, long-term potential for transforming our daily lives, if we can solve the associated research problems.

The sensors that, when distributed in the environment, comprise WSNs include cameras as vision sensors, microphones as audio sensors, and those capable of sensing ultrasound, infra-red, temperature, humidity, noise, pressure and vibration. Although the individual sensor's sensing range is limited, WSNs can cover a large space by integrating data from many sensors. Diverse and precise information on the environment may thus be obtained. Sensor networks are an emerging computing platform consisting of large numbers of small, low-powered, wireless *motes* each with limited computation, sensing, and communication abilities. It is still a challenge to realize a distributed WSN comprising: small and cost effective sensor modules; high speed, low latency and reliable network infrastructures; software platforms that support easy and efficient installation of the WSN; and sensor information processing technologies.

WSNs could potentially become a disruptive technology, for example because of social issues such as security and privacy, but the technological vision is for new and diverse types of applications for the social good. The environment can be monitored for fire-fighting, to detect marine ground floor erosion, and to study the effect of earthquake vibration patterns on bridges and buildings. Surveillance of many kinds can be supported, such as for intruder detection in premises. Wireless sensors can be embedded deeply within machinery, where wired sensors would not be feasible because: wiring would be too costly; could not reach the deeply embedded parts; would limit flexibility; would represent a maintenance problem; or would prevent mobility. Mobile items such as containers can be tagged, as can goods in a factory floor automation system. Smart price tags for foods could communicate with a refrigerator. Other classes of application include car-to-car or in-car communication.

But sensor network programming is difficult, and the human programming resource is costly. Complexities arise from the limited capabilities (processing, storage and transmission range) and energy resources of each node as well as the lack of reliability of the radio channel. As a result, application designers must make many complex, low-level choices, and build up software to perform routing, time synchronization, node localization and data aggregation within the sensor network. Unfortunately, little of this software carries over directly from one application to another, since it encapsulates application-specific tradeoffs in terms of complexity, resource usage, and communication patterns. No WSN application will therefore be seen as typical, and application-dependency will be higher than in traditional distributed applications.

Recent WSN research has focused increasingly on the application layer and an API at an appropriate abstraction level is needed urgently. Such an API would hide from programmers the complexities of sensor nodes and routing. The middleware technology must provide such an abstract API. In WSNs, the task of middleware is to collect large amounts of data from sensors, and/or on the movement of sensors. It must then perform aggregation and management of data in appropriate formats for the target distributed applications. WSNs have recently received a lot of attention in the research literature; recent survey and overview papers are: [56, 227, 8, 52, 214, 69, 6, 66, 10, 4, 180, 117]. These papers focus on operating systems, routing, or applications.

In this paper we report the latest trends in WSN research, focusing on middleware technology and related areas, and including application design principles. However, we do not cover every possible WSN technology; for example, mobile, ad hoc, communication technology may be important for WSNs, but it is out of scope of this paper because it already has an established base and hardware. First, we give an overview of WSNs and design aspects of applications, including existing research prototypes and industry applications. Secondly, we describe the technology supporting these sensor applications from the view of system architecture and network communication. We focus on the middleware technology, describing the current state of research that supports sensor network environments. We then highlight outstanding issues and conclude with future perspectives on middleware technology.

This paper continues as follows: section 2 describes characteristics of WSNs, section 3 describes applications design principles, section 4 describes technologies supporting WSNs, section 5 describes the recent hot topic programming paradigms, section 6 summarizes middleware technologies, section 7 describes future challenges, and we conclude in section 8.

2. WIRELESS SENSOR NETWORKS

A WSN is a collection of millimeter-scale, self-contained, micro-electro-mechanical devices. These tiny devices have sensors, computational processing ability (i.e.CPU power), wireless receiver and transmitter technology and a power supply. In a WSN a large number of sensor nodes usually span a physical geographic area. For example, the prototype of a future sensor node (mote) in the Smart Dust project [188], performs the wireless communication function, the sensor function, the power supply unit, and the information processing function on the MEMS (Micro Electro Mechanical System) chip, which has a scale only of several millimeters.

Typical WSNs communicate directly with a centralized controller or a satellite, thus communication between the sensor and controllers is based on a single hop. In future, a WSN could be a collection of autonomous nodes or terminals that communicate with each other by forming a multi-hop radio network and maintaining connectivity in a decentralized manner by forming an ad hoc network. Such WSNs could change their topology dynamically when connectivity among the nodes varies with time due to node mobility. But current, real-world deployment usually consists of stationary sensor nodes.

WSNs are intelligent compared with traditional sensors, and some WSNs are designed to use in-network processing, where sensed data can be gathered in situ and transformed to more abstract and aggregated high-level data before transmission. The combination of processing power, storage and wireless communication also means that data can be assimilated and disseminated using smart algorithms. The vast number of sensor nodes planned for many applications also implies a major portion of these networks would have to acquire self organization capability.

Intuitively, a denser infrastructure would create a more effective sensor network. It can provide higher accuracy and has more energy available for aggregation. If not properly handled, a denser network can also lead to collisions during transmission, and network congestion. This will no doubt increase latency and reduce efficiency in terms of energy consumption. One distinguishing characteristic of WSNs is their lack of strong boundaries between sensing, communication and computation. Unlike the Internet, where data generation is mostly the province of end points, in sensor networks every node is both a router and a data source.

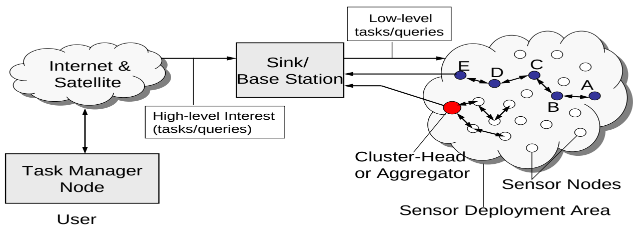


Figure 1: WSN Overview

2.1 WSNs vs. MANETs

There are two major types of wireless ad hoc networks: mobile ad hoc networks (MANETs) and wireless sensor networks (WSNs). A MANET is an autonomous collection of mobile routers (and associated hosts) connected by bandwidth-constrained wireless links. Each node is envisioned as a personal information appliance such as a personal digital assistant (PDA) fitted out with a fairly sophisticated radio transceiver. The nodes are fully mobile. The network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet. Factors, such as variable wireless link quality, propagation path loss, fading, multiuser interference, power expended, and topological changes, significantly increase the complexity of designing network protocols for MANETs. Security, latency, reliability, intentional jamming, and recovery from failure are also of significant concern.

A WSN consists of a number of sensors spread across a geographical area. Each sensor has wireless communication capability and sufficient intelligence for signal processing and networking of the data. A WSN can be deployed in remote geographical locations and requires minimal setup and administration costs. Moreover, the integration of a WSN with a bigger network such as the Internet or a wireless infrastructure network increases the coverage area and potential application domain of the ad hoc network. Sensed information is relayed to a sink node by using multi hop communication. The sink node is a sensor node with gateway functions to link to external networks such as the Internet and sensed information is normally distributed via the sink node (see Fig.1).

WSNs differ from MANETs in many fundamental ways. Viewing a WSN as a large-scale multi-hop ad hoc network may not be appropriate for many real-world applications. The communication overhead for configuring the network into an operational state is too large. The number of nodes in a WSN can be several orders of magnitude higher than the nodes in an ad hoc network and sensor nodes that are prone to failure are densely deployed. Sensor nodes mainly use broadcast, while most MANETs are based on the Peer-to-Peer (P2P) communication paradigm. Information exchange between end-to-end nodes will be rare in WSNs. They are limited in power, computational capacity and memory, and may not have global IDs. WSNs have a wide range of applications ranging from monitoring environments, sensitive installations, and remote data collection and analysis. In both MANETs and WSNs the nodes act both as hosts and as routers. They operate in a self organizing and adapting manner. Research and development in the areas of infrastructureless wireless networks have been advancing at a fast pace, and more effort needs to be dedicated in this direction for wide scale adoption and deployment. Current sensor hardware is resource and power constrained, but evolution of hardware and cost reduction will be improve rapidly. WSNs may eventually share the properties of MANETs.

Processing sensor data: Processing the data gathered by sensors distinguishes sensor networks from MANETs. The end goal is the detection/estimation of some events of interest, and not just communication. To handle and improve the detection performance, it is useful to perform fusion on the data from a single or multiple sensors. Data fusion requires the transmission of data and control messages and can be part of the WSN architecture. Collaborative sensor-data processing is another factor that distinguishes WSNs from MANETs. To improve the detection rate of events of interest it is often useful to aggregate data from multiple sensors. This, again, requires the transmission of data and control messages, and may put constraints on the network architecture. See also section 3.2.9 and 3.4.

2.2 WSAN

Wireless sensor and actor networks (WSANs) consist of a group of sensors and actors (actuators) linked by a wireless medium to perform distributed sensing and actuation tasks. In such a network, sensors gather information about the physical world, while actors take decisions and then perform appropriate actions upon the environment, which allows a user to effectively sense and act at a distance. However, due to the presence of actors, WSANs have some differences from WSNs. [188] gives a good summary of research issues in WSANs. Unlike sensor nodes which are small, cheap devices with limited sensing, computation and wireless communication capabilities, actors must be more complex. Their function is more energy-consuming than sensing and actors are resource-rich nodes equipped with better processing capabilities, stronger transmission power and longer battery life than sensors. The number of sensor nodes deployed for collecting data on a phenomenon may be of the order of hundreds or thousands. However, such a dense deployment is not necessary for actor nodes due to the different coverage requirements and physical interaction methods of the acting task. Hence, there are far fewer actors than sensors in WSANs. To provide effective sensing and actuating, a distributed local coordination mechanism is necessary among sensors and actors.

In WSANs, depending on the application, there may be a need to respond rapidly to sensor input. Moreover, so as to provide correct actions, sensor data must still be valid at the time of actuating. Therefore, the issue of real-time communication is important in WSANs since actions are performed on the environment in response to the sensing.

WSANs are fast emerging as a new sensing paradigm based on the collaborative effort of large number of sensors deployed close to or inside the phenomenon to be observed and have the potential of providing diverse services to numerous applications.

3. APPLICATION DESIGN PRINCIPLES

A vision of future ubiquitous computing is that tiny processors and sensors will be integrated with everyday objects in order to make them smart. Smart objects can explore their environment, communicate with other smart objects, and interact with humans. Fig. 2 shows an application space and potential real applications of the distributed WSN that allow monitoring of a wide variety of environmental phenomena with adequate quality and scale.

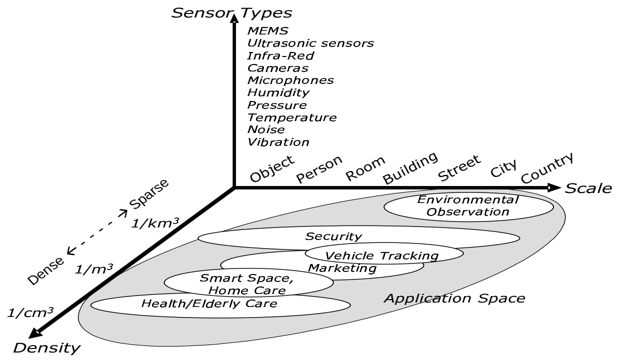


Figure 2: Application Space

3.1 Potential Applications

Current applications in research prototype environments or industry may be classified into the four categories below:

3.1.1 *Disaster/Crime Prevention and Military Applications*

Since 9/11, safety concerns about terrorism have increased dramatically and sensor network related research includes the detection and prevention of terrorist attacks. In [226], sensor networks are used for environmental monitoring: to detect distortion and structural problems in buildings in order to prevent disasters. Radiation sensors can be deployed in urban districts, for example at key road intersections, to detect terrorist attacks using radioactive substances [36]. Military WSNs can detect information about enemy movements, explosions, and other phenomena of interest and it is possible to calculate the position of a sniper using a sonic sensor [151, 199]. Many sensor network research projects inside the United States have received the support of DARPA (SensIT [153]); it has become the central topic in sensor network applications [237].

3.1.2 Environmental Applications

WSNs can be used to detect and monitor regional environmental changes in plains, forests, oceans, flood-levels, precision agriculture etc. [153, 40, 46, 197, 38, 213]. The common characteristic of these applications is that sensor data are aggregated in the servers and distributed with the location, and other environmental information, to the users. An example is GlacsWeb [153] which monitors glaciers in order to observe changes, possibly caused by global warming. Burrel [40] uses sensors to control vineyards, and also records and analyzes movements during the work of farmers. Zhang [235] has developed ZebraNet for observing and tracing wild animals using sensors.

3.1.3 Health Applications

Telemonitoring of human physiological data, tracking and monitoring patients and doctors inside a hospital, and assistance of the elderly are in this category. Wearable and implantable sensors can monitor a broad variety of conditions of the patients continuously at all times. This will reduce delays in obtaining test results, thus having a direct bearing on patient recovery rates. For trauma patients' survival it is particularly important that physicians can make a rapid and accurate diagnosis and recommend appropriate treatment. Telemonitoring may also enable testing and commencement of treatment in remote locations, as well as assisting in the precise location of accident or disaster sites.

3.1.4 Smart Spaces

Home automation and smart environments aim to create an intellectual space by means of a large number of networked sensors [190, 218, 68, 116]. Smart-Its [98] is a research project which uses sensors in the goods we use in daily life. Instead of bar-codes, where human interaction is necessary, creating autonomous sensor networks allows the position and quality of the goods (temperature humidity) to be detected in order to realize automated management systems. Managing inventory control, vehicle tracking and interactive museums could be in this category.

3.2 Design Aspects

Current applications for WSNs are mostly research prototypes or are custom made for a specific purpose. They are insufficiently mature to deploy widely in real world scenarios. At the current stage of research, there is still limited generic off-the-shelf smart sensor nodes, and there is not yet a generic application, which fulfills vastly diverse objectives. There is no typical WSN structure and architecture, and the basic goals of a WSN largely depend on the application. Single-hop networks, potentially using existing infrastructures (e.g. GSM) and devices (e.g. mobile phones with embedded sensors), are most attractive to industry at present. When WSNs are deployed, applications are not stand-alone but are integrated into a larger computing infrastructure.

The difficulty of sensor network research is constructing an open system that allows for the variety of real-world phenomena. Hardware constraints, such as the relation between capacity and power consumption, as well as other resources, must be taken into account. To deal with the complexity of sensor network research, Estrin et al. [69] suggest two design principles. First, a data centric design where the focus is managing sensor data instead of managing nodes. Secondly, an application-specific approach, where designing the physical and social characteristics of an application environment reduces the domain of discourse of the research. Much sensor network research tends to focus on specific hardware with efficient, ingenious communication control algorithms and system control architectures that address the specific resource constraints. A generic architecture does not emerge.

In this section, we identify different aspects of the characteristics of applications in order to help in designing WSN applications. The aspects we define are based on [182], extended with various data processing factors. Applying the right design approaches and technologies to the WSN applications is critical.

Table 2 in the Appendix classifies the sample applications according to the above aspects (except *Query ability*, *Reliability*, *Self configuration* and *Security*). The applications listed in the following subsections are research prototypes, commercial products and experimental projects, and the scale of the applications varies. All information was obtained through online information and available documents. The actual survey included over 50 projects. Due to space limitations, only 12 are listed below, which show various aspects of WSN applications.

3.2.1 Deployment

Sensor nodes may be installed at specific locations or be placed randomly. After the initial deployment, sensors may be added or replaced, which affects node location, density, and the overall topology.

Programs for each sensor node may be placed manually or automatically at runtime (see the programming paradigm in section 3.3).

3.2.2 Mobility

Sensor nodes may be attached to or carried by mobile entities. Mobility may be either an incidental side effect, or it may be a desired property of the system (e.g. to move nodes to interesting physical locations), in which case mobility may be either active (i.e. automotive) or passive (e.g. attached to a moving object not under the control of the sensor node). Mobility may apply to all nodes within a network or only to a subset of nodes. The degree of mobility may also vary from occasional movement with long periods of immobility in between, to constant travel. Mobility has a large impact on the expected degree of network dynamics and hence influences the design of networking protocols and distributed algorithms. The actual speed of movement may also have an impact.

3.2.3 Infrastructure

The various communication modalities can be used in different ways to construct a communication network. Two common forms are so-called infrastructure based networks and ad hoc networks. In infrastructure based networks, sensor nodes can only communicate directly with base stations. The number of base stations depends on the communication range and the area covered by the sensor nodes. In ad hoc networks, nodes can communicate directly with each other without an infrastructure. Nodes may act as routers, forwarding messages over multiple hops on behalf of other nodes.

Note that cost should not be forgotten. State of the art technology allows a Bluetooth radio system to cost less than 10 dollars. The cost of a sensor node should be much less than 1 dollar in order for the sensor network to be feasible.

3.2.4 Network Topology

One important property of a WSN is its diameter, that is, the maximum number of hops between any two nodes in the network. In its simplest form, a WSN forms a single-hop network, with every sensor node being able to communicate directly with every other node. An infrastructure based network with a single base station forms a star network with a diameter of two. A multi-hop network may form an arbitrary graph, but often an overlay network with a simpler structure is constructed such as a tree or a set of connected stars. The topology affects many network characteristics such as latency, robustness, and capacity. The complexity of data routing and processing also depends on the topology.

Given the large number of nodes and their potential placement in difficult locations, it is essential that the network is able to self-organize; manual configuration is not feasible. Moreover, nodes may fail (either from lack of energy or from physical destruction), and new nodes may join the network. Therefore, the network must be able to reconfigure itself periodically . Furthermore, while individual nodes may become disconnected from the rest of the network, a high degree of connectivity must be maintained.

3.2.5 Density and Network Size

The effective range of the sensors defines the coverage area of a sensor node. The density of the nodes indicates the degree of coverage of an area of interest by sensor nodes. The network size affects reliability, accuracy, and data processing algorithms. The density can range from a few sensor nodes to a hundred in a region, which can be less than 10m in diameter. The density μ is calculated as in [39]:

$$\mu(R) = (N\pi R^2)/A$$

where N is the scattered sensor nodes in region A, and R is the radio transmission range. Generally, $\mu(R)$ gives the number of nodes within the transmission radius of each node in region A.

3.2.6 Connectivity

The communication ranges and physical locations of individual sensor nodes define the connectivity of a network. If there is always a network connection (possibly over multiple hops) between any two nodes, the network is said to be connected. Connectivity is intermittent if the network may be partitioned occasionally. If nodes are isolated most of the time and enter the communication range of other nodes only occasionally, we say that communication is sporadic. Connectivity influences the design of communication protocols and data dissemination mechanisms.

3.2.7 Lifetime

Depending on the application, the required lifetime of a sensor network may range from a few hours to several years. The necessary lifetime has a high impact on the required degree of energy efficiency and robustness of the nodes, thereby requiring the minimisation of energy expenditure.

3.2.8 Node Addressability

This indicates whether or not the nodes are individually addressable. For example, the sensor nodes in a parking lot network should be individually addressable, so that the locations of all the free spaces can be determined. Thus, it may be necessary to broadcast a message to all the nodes in the network. If one wants to determine the temperature in a corner of a room, then addressability may not be so important. Any node in the given region can respond.

3.2.9 Data Aggregation

Data aggregation is the task of data summarisation while data are travelling through the sensor network. An excessive number of sensor nodes can easily congest the network, flooding it with information. The prevailing solution to this problem is to aggregate or fuse data within the WSN then transmit an aggregate of the data to the controller.

There are three major ways of performing data aggregation: First, diffusion algorithms assume that data are transmitted from one node to the next, thus propagating through the network to the destination. Along the way data may be aggregated, mostly with simple aggregation functions and assuming homogeneous data. Second, streaming queries are based on SQL extensions for continuous querying. Here data are considered to be transient while the query is persistent. Third, event graphs work on streams of events and compose simple events into composite events based on an event algebra. The event algebras from reactive middleware have been extended with temporal constraints for event correlation and transmission probabilities for WSNs. Events are consumed according to event consumption modes.

The different approaches above influence when and where aggregation of data in WSNs should be performed. This involves how to deal with time in distributed and unreliable environments, with state, asynchrony and unstable communication, redundancy and so forth. Different aggregation mechanisms require different resources and will therefore influence the routing strategy such as adapting the routing to the application or vice versa. Decisions must be made as to whether query processing is to be performed at sensor nodes or only at designated, resource rich nodes, placing simple filters at peripheral sensing nodes. Typical aggregation operations include MAX, MIN, AVG, SUM and many other well-known database management techniques.

3.2.10 Query Ability and Propagation

There are two types of addressing in sensor network; data centric, and address centric. In a data centric paradigm, a query will be sent to specific region in the network, whereas in addressing centric, the query will be sent to an individual node. A user may want to query an individual node or a group of nodes for information collected in the region. Depending on the amount of data fusion performed, it may not be feasible to transmit a large amount of the data across the network. Instead, various local sink nodes will collect data from a given area and create summary messages. A query may be directed to the sink node nearest to the desired location.

3.2.11 Data Dissemination

The ultimate goal of a WSN is to detect specified events of interest in a sensor field and to deliver them to subscribers. Because of the overlap in the proximity ranges of sensors, the same phenomena might be recorded by multiple sensor nodes. Alternatively, systematic aggregation might lose all the data on the same phenomenon. End-to-end event transfer schemes that fit the characteristics of WSNs are needed, in the same way that delivery semantics of asynchronous communication, such as publish/subscribe, is needed for wired distributed systems.

The electric power consumed depends substantially on how the sensor data is handled and communicated. Because the capacity of the battery of the sensor node is very limited, it is necessary to consider the extent to which the demands of applications can be met. Adaptive communication protocols (Power aware Protocols, that consider power consumption, are actively being researched.

3.2.12 *Real-Time*

Object tracking applications may need to correlate events from different source nodes in real-time. Real-time support (e.g. a physical event must be reported within a certain period of time) may be critical in WSNs. This aspect affects time synchronization algorithms, which may be affected by the network topology and the communication mechanism deployed.

3.2.13 *Reliability*

The reliability $R_k(t)$ or fault tolerance of a sensor node is modelled in [97] using the Poisson distribution to capture the probability of not having a failure within the time interval (0; t):

$$R_k(t) = exp(-\lambda_k t)$$

where λ_k and t are the failure rate of sensor node k and the time period, respectively. The fault tolerance level depends on the application of the WSN.

3.2.14 *Self Configuration*

Given the large number of nodes and their scattered placement in hostile locations, it is essential that the network be able to self-organize. Moreover, nodes may fail from limitation of energy, from physical destruction or any other means, and new nodes may need to join the network. The nodes can also coordinate to exploit the redundancy provided by high density so as to extend the overall system lifetime. The large number of nodes deployed in these systems will preclude manual configuration, and the environmental dynamics will preclude design-time preconfiguration. Nodes will have to self-configure to establish a topology that provides communication under stringent energy constraints. The network must be able to continuously and periodically reconfigure itself so that it can continue to function properly and serve its purpose. A high degree of connectivity is essential.

3.2.15 *Security*

Threats to a WSN are described in [14] and classified into the following categories:

- Passive Information Gathering: If communications between sensors, or between sensors and intermediate nodes or collection points are in clear, then an intruder with an appropriately powerful receiver and well designed antenna can passively pick off the data stream.

- Subversion of a Node: If a sensor node is captured it may be tampered with, interrogated electronically and perhaps compromised. Once compromised, the sensor node may disclose its cryptographic keying material, and access to higher levels of communication and sensor functionality may be available to the attacker. Secure sensor nodes, therefore, must be designed to be tamper proof and should react to tampering in a fail-complete manner where cryptographic keys and program memory are erased. Moreover, a secure sensor needs to be designed for its emanations not causing sensitive information to leak from it.

- False Node: An intruder might *add* a node to a system and feed false data or block the passage of true data. Typically, a false node is a computationally robust device that impersonates a sensor node. While such problems with malicious hosts have been studied in distributed systems, as well as ad-hoc networking, the solutions proposed there (group key agreements, quorums and per-hop authentication) are in general too computationally demanding for sensors.

- Node Malfunction: A node in a WSN may malfunction and generate inaccurate or false data. Moreover, if the node serves as an intermediary, forwarding data on behalf of other nodes, it may drop or garble packets in transit. Detecting and culling these nodes from the WSN becomes an issue.

- Node Outage: If a node serves as an intermediary or collection and aggregation point, what happens if the node stops functioning? The protocols employed by the WSN need to be robust enough to mitigate the effects of outages by providing alternate routes.

- Message Corruption: Attacks against the integrity of a message occur when an intruder inserts itself between the source and destination and modifies the contents of a message.

- Denial of Service: A denial of service attack on a WSN may take several forms. Such an attack may consist of jamming the radio link, exhausting resources or misrouting data. Karlof and Wagner [118] identify several DoS attacks including: *Black Hole*, *Resource Exhaustion*, *Sinkholes*, *Induced Routing Loops*, *Wormholes*, and *Flooding* that are directed against the routing protocol employed by the WSN.

- Traffic Analysis: Although communications might be encrypted, an analysis of cause and effect, communications patterns and sensor activity might reveal enough information to enable an adversary to defeat or subvert the mission of the WSN. Addressing and routing information transmitted in clear often contributes to traffic analysis.

Security aspects of wireless sensor networks have received little attention compared with other aspects. The main focus of WSN security has been on centralized communication approaches; there is a need to develop distributed approaches.

3.3 Operational Paradigms

An operational scenario for sensor networks, including data processing, requires consideration of complex combination of the aspects described in the previous section. Operational paradigms of sensor nodes vary and depend on the deployment of the sensor nodes and applications. Operational paradigms may be classified into the following categories [14], where it is assumed that a base station or sink node exists as part of WSNs.

3.3.1 Single-hop to Sink

Sensor nodes sense and transmit the data to the sink nodes (e.g. base station or cluster head nodes), which is within range of transmission, thus routing or coordination among nodes is not required. This paradigm, therefore, uses a centralized, point-to-point, single-hop communication model, as though the infrastructure supported wireless networks. The sensors take periodic measurements and transmit this data directly either immediately following data collection or when scheduled, at some periodic interval. A problem of this paradigm is that sensor nodes are out of communication when they are out of radio range from the sink nodes.

3.3.2 Multi-hop to Sink

This paradigm allows sensor nodes far away from the sink nodes to transmit data to neighbouring sensor nodes, which in turn forward the data toward the sink nodes. The forwarding process may involve multiple sensor nodes on the path between the source node and the collection point. Thus, this paradigm uses a centralized, multi-hop communication model. Regardless of the length of the path, the data eventually reaches the collection point. Coordination among nodes in *routing* the data to the base station is part of this paradigm.

3.3.3 On-Demand Operation

In this paradigm, sensors receive commands from a controller, either directly or via forwarding, and configure themselves based on the requests. Both the *Single-hop to Sink* and *Multi-hop to Sink* paradigms are *many-to-one* communication models, designed exclusively for non on-demand data transmission. On the other hand, in *On-demand Operation*, commands may be broadcast from the controller (e.g. base station) to the entire WSN, or may be unicast to several sensor nodes in (*one-to-many* communication). Data transmission is expensive in a WSN, and non on-demand data transmissions may reduce the lifetime of the WSN significantly. Also, transmissions may be unnecessary (e.g. 100 sensor nodes reporting directly to a sink that the temperature in the region is 35C). If transmissions could be regulated appropriately, then the lifetime of the WSN could be increased without affecting the quality of information reaching the sink.

Consider, for example, a group of sensor nodes that are deployed to monitor temperature. Upon deployment, all nodes begin operating in the idle mode, which is a low power mode. The controller broadcasts a wakeup to the group, which react to this command by making the transition to the active state. Subsequently, the controller broadcasts a get data command, which solicits data from the sensor nodes. Finally, the controller instructs the sensor nodes to idle and the cycle repeats periodically. Thus, the combination of the one-to-many and many-to-one communication models is more energy efficient than simply using the latter for non on-demand data transmission. If unicast messaging is employed, then some form of addressing of each individual node needs to be employed. However, no guarantees on the unicast message actually reaching the intended recipient can be given, because none of the nodes in the WSN may be aware of either route(s) to the recipient or the topology of the WSN.

3.3.4 Self Organization

Upon deployment, the WSN self organizes, and a central controller(s) learns the network topology. Knowledge of the topology may remain at the controller (e.g. base station) or it may be shared, in whole or in part, with the nodes of the WSN. This paradigm may include the use of more powerful sensors that serve as cluster heads for a small coalition within the WSN, and it requires a strong notion of routing. This paradigm extends the bidirectional communication model by introducing the concept of self organization. It consists of three primary tasks: node discovery, route establishment and topology maintenance. The accomplishment of these three tasks leads to the formation of a true WSN. While the previous paradigms used a centralized communication model, this and the following paradigms seek to employ a combination of centralized and distributed communication in order to allow the WSN to perform as efficiently as possible. Topology maintenance in a WSN is unlike that in any other wireless network. In WSNs, nodes may be stationary or mobile. When there is no mobility, the topology, once established, usually does not change. However, as nodes perform their assigned tasks they deplete and eventually exhaust their energy store, causing them to die. The WSN may be *refreshed* by the periodic addition of new nodes to the WSN.

3.3.5 Data Aggregation

This paradigm is that data aggregation is performed in order to reduce the traffic and conserve the power (see Data Aggregation in section 3.2.9). All sensor nodes transmit the data towards the base station (or sink nodes) in either an on-demand or non on-demand manner.

3.3.6 Reacting Process

All the above paradigms essentially consider gathering data and delivering them to sink nodes. Nodes in the WSN are not concerned with the semantics of the data obtained through the sensing task. Predicated upon their own measurements and upon the values of incoming data, requires that sensor nodes act based on those values. The action may be a calculation, or actions triggered by incoming data. If multi routing protocols are operated in the WSN, an action may be a decision on the choice of protocol. Alternatively, the routing protocol may be performed based only on the geographical location, in which case the node processes accordingly. A sensor actuator node can analyze the data and react to change the state of the real world such as sending commands to actuators. WSAN belongs to this paradigm.

3.4 Aggregation, Filtering and Correlation

Sensor fusion is a combination of sensed data or data derived from sensed data such that the resulting information gives more information than individual sensed data. Sometimes, the information may be acquired from multiple sources (sensor, database, information gathered by human, etc.). Since sensor fusion algorithms often afford information about the measurement instant, or even coordinated distributed measurements, at a particular point in time, the employment of a time-triggered architecture for distributed sensor fusion systems is advantageous. In-network data aggregation in WSN summarizes sensor values in some or all of a sensor network, and it highlights an importance of data aggregation. Many aggregation operations come from database operations such as taking an average or maximum value in existing middleware (e.g., TinyDB [146]). However, the aggregation process fundamentally depends on an application's requirement, and to fulfil the request, a simple aggregation may not be enough. It may also involve event (data) filtering and correlation during processing sensor data. It is important to use approaches taken from global computing in the processing of sensor data.

Event Correlation is essential when the data is produced in a WSN and multi-step operation carries the data from event sources to the end-applications. These may reside in the Internet and it must be possible to combine information collected by wireless devices with higher level information or knowledge in distributed environments.

In [230], we introduced a generic composite event semantics, which combines traditional event composition with the generic concept of data aggregation in wireless sensor networks. The main focus is on supporting time and space related issues such as temporal ordering, duplication handling, and interval-based semantics, especially for wireless network environments. We presented event correlation semantics defining precise and complex temporal relationships among correlated events using interval-based semantics.

Event correlation is deployed sometimes as a part of applications, event notification services, or middleware services. Definition and detection of composite events vary, especially over distributed environments. Event composition operators do not necessarily have the same semantics in different network environments, while similar semantics might be expressed using different operators.

Many event-based middlewares offer content-based filtering. Here, subscribers use flexible querying languages to declare their interests with respect to event contents. Event filtering allows specific attribute values on an event type to be selected, while event correlation addresses the temporal and spatial relation among instances of event types. Filtering and correlation share many properties. WSNs led to new issues to be addressed in event correlation. Traditional networking research approached data aggregation as an application-specific technique that can be used to reduce the network traffic. Fig.3 highlights the relation among aggregation, filtering and correlation.

TinyDB is an inquiry processing system for sensor networks and takes a data centric approach, where each node keeps its data, and nodes execute retrieval and aggregation (in-network aggregation), with on-demand based operation to deliver the data to external applications. TinyLIME [57] is enhancing LIME (Linda In Mobile Environments) to operate on TinyOS. In TinyLIME, a partitioned tuple space, as well as LIME is maintained on each sensor node, and a coordinated tuple space is formed when connecting with the base station within one hop. This works as middleware by offering an abstract interface to the application. The current form of TinyLIME does not provide any data aggregation function. Only a data filtering function, based on the Linda/LIME operation, is provided at the base station node. On the other hand, TinyDB supports a data aggregation function via SQL query, but redundancy/duplication handling is not clear thus far.

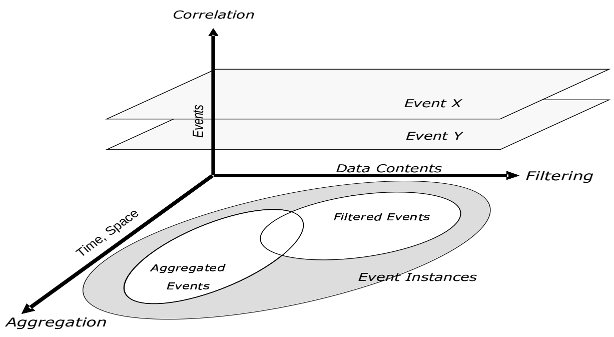


Figure 3: Filtering, Aggregation and Correlation

The coordination of nodes within WSN differs from the other wireless ad hoc networks, where the group of nodes act as a single unit of processors in many cases. For a single phenomenon, multiple events may be produced in order to avoid the loss of event instances, which is a totally different concept from traditional duplicate events.

Middleware research for WSN has recently increased, but most research focuses on in-network operation for specific applications. A global view of event correlation, filtering and aggregation over whole distributed systems must be addressed. Aggregation should have three stages: local, neighbour, and global.

4. TECHNOLOGIES SUPPORTING WSN

The advance of WSN depends on a wide range of technologies, such as hardware, system software and network communication. One of the difficult issues for WSNs is that the application requirements on WSNs differ from one application to another. In [135], 20 different properties are measured by commercial sensor systems using electrical, photonic, seismic, chemical and other transduction principles. Desirable, quantified properties are ease of installation, self identification, self diagnosis, reliability, time awareness, and locality awareness. The results show that it is not easy to define generic requirements on accuracy and sampling rate, for sensor nodes and networks. Fig. 4 outlines the technologies surrounding WSNs.

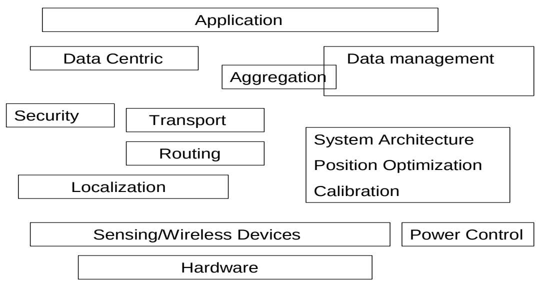


Figure 4: Technologies for WSNs

This section describes various technologies supporting sensor networks. The issues related to the sensor node hardware is out of scope in this paper. We focus on the system and network communication technologies in this section and discuss the current state of these technologies.

4.1 System Architecture

This section describes a research trend in system architecture including operating systems, positioning and tracking targets, localization, time synchronization, security, and programming paradigms. We locate the programming models topic in section 5.

4.1.1 System Software and Platform

Many research groups in both academia and industry are currently devoted to the development of sensor platforms. One of the first and probably most popular platforms is Berkeley motes. TinyOS [96] is an operating system to control the hardware of motes. TinyOS is an open source project to provide microkernels. These include stripped-down versions of functional units in operating systems that can be selected to provide system support only when their functionality is needed. The bottom line is that due to severe restriction on various resources, the system should consist of only those functionalities needed, whether implemented in hardware or system software.

TinyOS provides lightweight thread support and efficient network interfaces. Two issues in particular are addressed: (1) to support concurrency, different flows of data must be kept moving simultaneously, and (2) the system must provide efficient modularity (i.e., hardware and application-specific components must combine with little processing and storage overhead). Users can select only the minimum component that the application needs from among these component groups. Components can be executed in parallel, and while waiting events, resource consumption (Neither Blocking nor Polling are done) is minimised.

NesC [72], which is a C language extension can be used on TinyOS. NesC has the feature such as event driven, parallel execution, component orientation etc. and corresponds to the components and event processing of TinyOS. Greenstein et al. [59] offer a component library SNACK (Sensor Network Application Construction Kit) to facilitate the development on TinyOS. Dai et al. [76] propose the file system ELF (Efficient Log structured Flash file system) for storage using flush memory.

4.1.2 Localization

When sensor information is obtained from a certain sensor node, its physical position is that at which the sensor information is obtained. Therefore, the position of the sensor node is essential information for analyzing sensor information. This information can be used for various algorithms such as location based routing [165].

Various research has focussed on detecting the positions of sensor nodes. For example, Moore [158] proposes an algorithm that successfully locates nodes in a sensor network with noisy distance measurements, using no beacons or anchors. Shang [193] and Ji [111] use multi dimensional scaling (MDS) to identify positions. This uses connectivity information such as who is within communications range of whom to derive the locations of the nodes in the network, and can take advantage of additional data, such as estimated distances between neighbours or known positions for certain anchor nodes, Moreover, Cheng [49] proposes the technique of a time based positioning scheme (TPS). TPS relies on TDoA (Time-Difference-of-Arrival) of RF signals measured locally at a sensor to detect range differences from the sensor to three base stations. These range differences are averaged over multiple beacon intervals before they are combined to estimate the sensor location through trilateration.

Accurate positioning is a key for many sensor applications including surveillance networks, robotic sensors, location based routing in WSNs, smart spaces and environmental monitoring by mobile sensors. Although GPS can potentially provide accurate positioning, the complexity of the required receivers may be too costly for inexpensive sensor nodes. Furthermore, the GPS signal is extremely weak and positioning can be unreliable inside buildings or under dense foliage. These drawbacks to GPS positioning have led to increasing interest in GPS-less distributed radio location methods for wireless sensor networks.

4.1.3 Spatial Temporal Correlation

WSNs are characterized by the dense deployment of sensor nodes that continuously observe physical phenomena. Because of high density in network topology and the nature of the physical phenomena, sensor observations are highly correlated in space and time domains.

- Spatial Correlation: Typical WSN applications require spatially dense sensor deployment for reliable coverage.

- Temporal Correlation: Event tracking may require sensor nodes to periodically perform observation and transmission of sensed data.

The spatial and temporal correlations along with the collaborative nature of the WSN bring significant potential advantages for efficient communication protocols. Akyildiz et al. [9] proposes a theoretical framework to model the spatial and temporal correlations in sensor networks. The objective of this framework is to enable the development of efficient communication protocols which exploit these advantageous intrinsic features of the WSN paradigm.

An example of a communication abstraction is *Spatial Programming* [30] which uses Smart Messages to provide content-based spatial references to embedded resources.

4.1.4 Position Optimisation

Coverage is the spatial sensing range of a sensor node. This has to be coordinated among sensor nodes to avoid redundancy, and to take account of communication distance and other characteristics of sensing tasks. Research on positioning sensor nodes has taken different approaches. For example, sensor nodes can move to optimise their own positions [41, 198, 175, 219]. Batalin [22] proposes a sensor architecture that combines both fixed and mobile sensor nodes to achieve a spatio temporal environment coverage. Mobility allows the networked sensor system to always seek the most efficient spatio temporal sampling distribution to achieve a specified accuracy of environmental variable reconstruction. Further, mobility also permits the system to respond to initially unpredictable and variable environmental evolution. Recently Ravelomanana et al. [177] analysed various critical transmitting/sensing ranges for connectivity and coverage in three-dimensional sensor networks. Kumar et al. [127] formulate the coverage problem as a decision problem, and analyses to determine whether every point in the service area of the sensor network is covered by at least k sensors, where k is a predefined value. The sensing ranges of sensors can be unit disks or non unit disks. They present polynomial time algorithms, in terms of the number of sensors that can easily be translated to distributed protocols.

4.1.5 Time Synchronization

The management of time and time related operations in WSNs is essential to timestamp events, control an operation cycle, synchronize network operations and so forth. Network Time Protocol (NTP) [157] is an Internet standard. NTP allows for assignment of real-time timestamps with given maximal errors. Global and partial order of events can be obtained with a certain accuracy based on timestamps. However, NTP requires frequent message exchange to synchronize a clock in each node, and existence of NTP server cannot be expected in WSN environments. This results in NTP not being suitable for WSNs and a new type of time synchronization is needed [201]. Dai et al. [58] propose an architecture TSync, where push and pull time synchronization mechanisms are combined. Maróti et al. [158] use flooding in multi hop sensor networks. Li et al. [137] discuss three methods to achieve global synchronization in a sensor network: a node based approach, a hierarchical cluster based method, and a fully locad diffusion based method. Their *Diffusion Method* is a synchronous and asynchronous implementation of diffusion based protocols.

Prakash et al. [173] present a GPS based virtual global clock that is used for timestamping events, and deploys a similar concept 2g-precedence. Without the existence of GPS, there is no means to synchronize the clocks of all the nodes in a deterministic fashion with an upper bound independent of the message propagation delay and system size. Logical time cannot be used to determine temporal ordering, because causal ordering of events in the real world must be obtained. Thus, physical time has to be used requiring clock synchronization. However, most of the synchronization algorithms rely on partitioned networks. Post-facto synchronization [65] is based on unsynchronized local clocks but limits synchronization to the transmit range of the mobile nodes. In [184], Römer takes a similar approach using unsynchronized clocks. The idea of the algorithm is not to synchronize the local computer clocks of the devices but instead to generate timestamps from a local clock. When such locally generated timestamps are passed between devices, they are transformed to the local time of the receiving device. Lucarelli et al. [143] builds protocols on the recent work of Hong, Cheow, and Scaglione [99] in which the synchronization update rules are mode led by a system of pulse coupled oscillators. They define a class of models that converge to a synchronized state based on the local communication topology of the sensor network only, thereby lifting the all-to-all communication requirement implicit in [99]. Under some rather mild assumptions of the connectivity of the network over time, these protocols still converge to a synchronized state when the communication topology is time varying.

4.1.6 Object Tracing

A standard problem of WSNs includes the problem of localization of one observation object (target) perceived by two or more sensors. For instance, to measure the position of an intruder vehicle according to sensor information on each sensor node located in an area. Bergamo [24] proposes to identify the position using a sensor array of iPAQs. Cheng et al. [48] propose dynamic construction of clustering. Instead of assuming the same role for all the sensors, their vision is a hierarchical sensor network that is composed of (1) a static backbone of sparsely placed high capability sensors which will assume the role of a cluster head (CH) upon triggering by certain signal events; and (2) moderately to densely populated low end sensors whose function is to provide sensor information to CHs upon request. A cluster is formed and a CH becomes active, when the acoustic signal strength detected by the CH exceeds a predetermined threshold. The active CH then broadcasts an information solicitation packet, asking sensors in its vicinity to join the cluster and provide their sensing information. The proposed dynamic clustering algorithm effectively eliminates contention among sensors and yields more accurate estimates of target locations as a result of better quality data collected and fewer collisions. Zhang et al. [235] propose a tree based approach to facilitate sensor nodes collaborating in detecting and tracking a mobile target. As the target moves, many nodes in the tree may become distant from the root of the tree, and hence a large amount of energy may be wasted for them to send their sensing data to the root. Hu et al. [102] introduce the sequential Monte Carlo Localization method and argue that it can exploit mobility to improve the accuracy and precision of localization. Gui et al. [78] propose a collaborative messaging scheme that wakes up and shuts down the sensor nodes with spatial and temporal preciseness. This approach quantifies the trade-off between power conservation and quality of surveillance while presenting guidelines for efficient deployment of sensor nodes for target tracking application. EnviroTrack [1] is the first programming support for sensor networks that explicitly supports tracking mobile objects (see details in 6.2.3).

4.1.7 Security

Sensor network security is a critical issue but minimal research has been done compared to other aspects of WSNs. Sensor nodes are resource-constrained and embedded in physical environments, where unlimited resource for the calculation can not be expected. A different technology from existing network security is required for WSNs. In [14], Avancha et al. provide a good summary of the direction of security research in WSNs. Wood [224] provides a survey of many kinds of denial of service attacks in sensor networks and discusses defence technologies. Perrig describes necessary technologies for the security for WSNs [171] and sumarises as follows.

For network communication, desired technologies are sharing methods of encryption keys and encrypting methods [62], privacy, DoS (denial of service) attack, secure routing, discovery of malicious nodes and their restoration. [170] addresses secure communication in resource constrained sensor networks, introducing two low level secure building blocks. For system support, group management, invasion discovery and secure data aggregation are needed. Karlof implemented TinySec [119] as an encryption module for the link layer. TinySec has achieved operation on the resources of TinyOS that are very limited. Karlof [118] analyzed security flaws of various routing protocols on WSNs, and proposed countermeasures to enhance sensor network routing. To defend against the rushing attack, this paper proposed that every node only process beacon messages through bidirectional links as well as verified neighbour nodes. While the issue of intrusion tolerance has been known for quite some time, recent increase in the need for safety critical systems has significantly raised research activity in this area. Recent projects addressing intrusion tolerance include [174, 187]. All these projects are aimed at providing intrusion tolerance capabilities in a traditional, resource rich computing environment.

The use of hybrid models for communication security will enable easier integration of data aggregation and key management algorithms. This will also ensure flexibility and adaptability of the WSN; self healing mechanisms and the ability to react to changing conditions can also be easily integrated. Both centrad and distributed key management techniques for WSNs have been discussed in the literature. Research efforts directed toward this problem have shown that key distribution in WSNs can be energy efficient and secure under certain conditions.

Distributed key management techniques are completely independent of any security architecture. For example, the work on secure pebblenets [21] is mainly concerned with choosing a key manager in every round, but does not address the issues involved in using the key for encryption, authentication or other security functions. On the other hand, the SPINS project [170] assumes pre-deployed keying in the entire security architecture. The ideal security model will consist of a combination of robust, energy efficient, secure key distribution mechanisms with well defined, comprehensive security architectures. A similar situation is observed when security architectures and data aggregation algorithms are considered. Current security architectures do not really consider the issue of integrating data aggregation algorithms; rather they assume that designated nodes, such as the controller or cluster heads, will perform the required aggregation functions. On the other hand, data aggregation algorithms assume complete and unhindered cooperation among all sensor nodes in the WSN as far as performing the aggregation functions is concerned. This assumption is non-trivial; a security protocol that supports such a cooperation model will not be scalable because pairwise communication security is required across the WSN. Thus, integration of energy efficient data aggregation algorithms with robust security architectures is essential in designing the ideal security model.

Flexibility, adaptability and self healing mechanisms are essential to the functioning of a WSN and optimal resource use during its lifetime. None of the existing security architectures use the data associated with sensed environmental conditions to help detect the beginnings of attacks or of aberrant behaviour by nodes. This reduces the ability of the WSN to protect itself from attacks mounted within and outside the network. In fact, detecting and preventing attacks from within, i.e. attacks mounted by compromised nodes, is a harder problem than preventing external attacks such as jamming. Thus, the move toward the ideal security model calls for the design and development of compact, lightweight mechanisms to capture and reason over data describing environmental and security conditions.

While there is little need for security in environmental monitoring, intelligent agriculture or radiation detection and other natural or wide range phenomenon, when a similar setting is used to monitoring human activities, the concern about privacy, and even safety, becomes a major factor. WSNs in assisted living or intelligent offices, are valuable in terms of automation, remembering and remote assistance. But the same technology that allows doctors and relatives to monitor the condition of an elderly person can lead to breaches of privacy if data is processed in an improper manner or accessed by unauthorized persons. The capabilities of automated analysis and remote access make this new generation of sensing technology an even worse threat to individuals' privacy. This issue can be addressed with a combination of technical measures and analytic frameworks from the perspective of law and psychology. Techniques such as tighter access control to the collected data, secure channel communications, options for users to voluntary opt out, or control of data granularity can all mitigate the privacy concern. Regarding the privacy issue from the perspective of law, Jacobs has suggested an analytic framework based on the Fourth Amendment and Supreme Court ruling, with an audience of concern, and the motivation of the reasoning process as the two axes of the paradigm [110].

MANET Security: MANETs employ a distributed, multi-hop, node-centric communication model. In a MANET, users control their wireless devices, however the device itself has some degree of autonomy. The autonomy is best illustrated by a device's choosing the most appropriate set of neighbouring devices to contact based on user requirements. Thus, communication in MANETs is node centric, rather than user centric. We may immediately observe that device authentication and data confidentiality are much more important than access control, which may not be relevant in certain situations. Unlike infrastructure supported wireless networks, the security problem confronted by a MANET is to mitigate the actions of malicious users who may attempt to disrupt communication in the network. Thus, security protocols in MANETs employ mechanisms such as certificates to authenticate users and encrypt data using symmetric or asymmetric algorithms. Due to the fact that MANETs allow multi-hop communications, most attacks are directed against the protocols that route data between intermediate nodes on the path from source to destination. Thus, network level security is the focus of attention in security research related to MANETs. Protocols and methods designed to address this issue include SEAD [100], Ariadne [101], security enhancements in AODV [26], secure position aided routing [43], secure ad hoc routing [168] and an on-demand secure routing protocol resilient to Byzantine failures [15]. End-to-end security is a non-trivial problem in MANETs because security protocols must rely on intermediate nodes which, depending upon their individual capabilities, may not contain all of the mechanisms required by the protocols.

4.2 Network and Communication Control

This section describes research trends in network and communication control in WSNs. A more intensive survey of network communication can be found in [4, 6]. The research on network control includes that on communication methods, saving power consumption, congestion control, topology management, routing, and modelling. Current architectures for the Internet and ad hoc wireless networks may not be used for WSNs for the following reasons.

- Number of sensor nodes in a WSN is much higher than an ad hoc network.
- Sensor nodes failure rate is high.
- Sensor nodes are more resource constrained.
- Sensor nodes are limited in power.
- The use of acknowledgement packets should be used sparingly.

Thus, an architecture for WSNs will be:

- Combine power and routing awareness.
- Integrate data with network protocols.
- Communicate power efficiently.
- Share tasks among neighbour nodes.

Several research projects have reported on new network control algorithms to solve sensor network specific problems such as constrained resources, localization and power consumption.

4.2.1 Power Saving Communication Protocol

Sensors are meant to be left in a real environment for the long term and saving power consumption is an important issue. The function that consumes most power in WSNs is communication. Reducing power consumption requires optimisation across all layers, from the physical layer, channel coding, and media access control layer up through the routing, transport, and application layers. The MAC layer plays the most crucial role in the communication protocols for energy efficiency, especially for networks with low duty cycling radios. In [67, 172], a low power operation through a careful codesign approach combining a dedicated duty cycled radio with a low power MAC protocol is proposed. Reason et al. [178] introduce a technique of application aware radio duty cycling called on-demand spatial TDMA for a moderatesize, multi-hop, sensor network,.

4.2.2 Congestion Control

Congestion control in wired networks is usually done using end-to-end and network layer mechanisms acting in concert. However, this approach does not solve the problem in wireless networks because concurrent radio transmissions on different links interact with and affect each other, and because radio channel quality shows high variability over multiple timescales. WSNs are based on broadcasts as the main communication mechanism. Hull et al. [106] examine three techniques that span different layers of the traditional protocol stack: hop by hop flow control, rate limiting source traffic when transit traffic is present, and a prioritised medium access control (MAC) protocol. The combination of these techniques can improve network efficiency. Ee et al. [64] propose a distributed and scalable algorithm that eliminates congestion within a sensor network, and that ensures the fair delivery of packets to a central node, or base station. Kang et al. [115] show a method to estimate the communication capacity of end-to-end in WSNs.

4.2.3 Topology Management

Heinzelman et al. [92] propose and analyze a low energy adaptive clustering hierarchy (LEACH). This is a protocol architecture for micro sensor networks that combines the ideas of energy efficient cluster based routing and media access together with application specific data aggregation to achieve good performance in terms of system lifetime, latency, and application perceived quality. LEACH includes a new, distributed cluster formation technique that enables self organization of large numbers of nodes, algorithms for adapting clusters and rotating cluster head positions to distribute the energy load evenly among all the nodes, and techniques to enable distributed signal processing to save communication resources. Sohrai et al. [206] propose a formation mechanism for wireless unattended ground sensor applications using a multi-cluster hierarchical topology (Rendezvous Clustering Algorithm), where a novel dual radio architecture is used. Smaragdakis et al. [203] propose SEP, a heterogeneity-aware protocol to prolong the time interval before the death of the first node, which is crucial for many applications where the feedback from the sensor network must be reliable. SEP is based on weighted election probabilities of each node becoming the cluster head according to the remaining energy in each node. Younis et al. [232] propose a similar approach, HEED (Hybrid Energy Efficient Distributed clustering), that periodically selects cluster heads according to a hybrid of their residual energy and a secondary parameter, such as a node's proximity to its neighbours or node degree. Cerpa et al. [44] propose an adaptive self configuration topology mechanism, where the ASCENT algorithm is deployed. ASCENT only builds a subset of the nodes necessary to establish a routing forwarding backbone. In ASCENT, each node assesses its connectivity and adapts its participation in the multi-hop network topology based on the measured operating region. Initially a small number of active nodes participate in routing, the rest being in passive mode. Nodes in passive mode regularly turn to test mode and change to active mode when there are not many neighbour nodes and loss of the packets is high. Ma et al. [144] propose a hub spoke network topology that is adaptively formed according to the resources of its members. A protocol named Resource Oriented Protocol (ROP) was developed to build the network topology. This protocol principally divides the network operation into two phases. In the topology formation phase, nodes report their available resource characteristics, based on which a network architecture is built optimally.

4.2.4 Routing

Many routing protocols have been proposed [103, 95, 215, 56], where the network topology changes frequently because of node failure and communication conditions. Helmy et al. [95] propose an architecture that is geared towards one shot frequent queries in sensor networks. Their approach aims at reducing the total energy cost of query resolution as opposed to searching for high quality routes. The architecture uses a hybrid approach, where each node collects information about nodes in its proximity, up to R hops away, using a link state protocol. Beyond this proximity, the novel notion of contacts that act as short cuts to reduce the degrees of separation between the request source and the target is introduced. Trakadas et al. [215] classify a selection of algorithms proposed for ad hoc networks according to their relevance and efficiency. A spatial position node from GPS or other coordination mechanisms can be used for geographic routing and there have been many proposals using this, such as [155, 191, 71]. Survey papers [4, 6, 215] contain useful details of routing protocols in WSNs. Data routing approaches in WSNs fall into into three main categories, namely data centric, hierarchical and location based. A few other protocols follow the traditional network flow and QoS modelling methodology.

There are some hybrid protocols that fit under more than one category as shown in Table 1. The table summarizes the classification of the hybrid protocols (see more detailed explanation in [6]. Protocols, which name the data and query the nodes based on some attributes of the data are categorized as data centric. Many researchers follow this paradigm in order to avoid the overhead of forming clus-

Routing protocol	Reference	Data Centric	Hierarchical	Location Based	QOS	Network Flow	Aggregation
1. SPIN	[90]	✓					✓
2. Directed Diffusion	[108]	✓					✓
3. Rumor Routing	[37]	✓					✓
4. Shah et al.	[192]	✓		✓			
5. GBR	[189]	✓					✓
6. CADR	[53]	✓					
7. COUGAR	[61]	✓					✓
8. ACQUIRE	[186]	✓					
9. LEACH	[91]		✓				✓
10. TEEN&APTEEN	[149]	✓	✓				✓
11. PEGASIS	[138]		✓				✓
12. Younis et al.	[231]		✓	✓			
13. Subramanian et al.	[211]		✓				✓
14. MECN&SMECN	[179]			✓			✓
15. GAF	[228]		✓	✓			
16. GEAR	[233]			✓			
17. Chang et al.	[45]		✓			✓	
18. Kalpakis et al.	[113]			✓		✓	
19. Akkaya et al.	[5]		✓		✓		
20. SAR	[205]				✓		
21. SPEED	[86]			✓	✓		

Table 1: Classification of routing protocols in sensor networks

ters, the use of specialized nodes etc. However, the naming schemes such as attribute value pairs might not be sufficient for complex queries and they are usually dependent on the application. Efficient standard naming schemes are one of the most interesting future research direction relating to this category. On the other hand, cluster based routing protocols group sensor nodes to relay the sensed data efficiently to the sink. The cluster heads are sometimes chosen as specialized nodes that are less energy constrained. A cluster head performs aggregation of data and sends it to the sink on behalf of the nodes within its cluster. The most interesting research issue regarding such protocols is how to form the clusters so that the energy consumption and contemporary communication metrics such as latency are optimised. The factors affecting cluster formation and cluster head communication are open issues for future research. Moreover, the process of data aggregation and fusion among clusters is also an interesting problem to explore.

Protocols that utilize the location information and topological deployment of sensor nodes are classified as location based. The number of energy aware location based approaches found in the literature is rather small. The problem of intelligent utilization of the location information in order to aid energy efficient routing is the main research issue. Spatial queries and databases using distributed sensor nodes and interacting with the location based routing protocol are open issues for further research.

Although the performance of these protocols is promising in terms of energy efficiency, further research is needed to address issues such as Quality of Service (QoS) posed by video and imaging sensors and real-time applications. Energy aware QoS routing in sensor networks will ensure guaranteed bandwidth (or delay) through the duration of connection as well as providing the use of the most energy efficient path. QoS routing in sensor networks has several applications including real-time target tracking in battle environments, emergent event triggering in monitoring applications etc.

Currently, there is little research that looks at handling QoS requirements in an energy constrained environment like sensor networks. Another interesting issue for routing protocols is the consideration of node mobility. Most of the current protocols assume that the sensor nodes and the sink are stationary. However, there might be situations where the sink, and possibly the sensors, need to be mobile. In such cases, the frequent update of the position of the command node and the sensor nodes and the propagation of that information through the network may excessively drain the energy of nodes. New routing algorithms are needed in order to handle the overhead of mobility and topology changes in such an energy-constrained environment. Other possible future research for routing protocols includes the integration of sensor networks with wired networks (i.e. the Internet).

4.2.5 Modelling

The technology that estimates the performance of the sensor network beforehand is requested by modelling, and analyzing the operation of the sensor network of a real environment. That helps to investigate a theoretical performance of the network control in a real environment, where networks are more dynamic. For example, under ideal settings, drastic performance improvement over strictly address centric routing schemes is proven. While geographic routing has been shown to be correct and efficient when location information is accurate, its performance in the face of location errors is not well understood. Helmy et al. [207, 122] analyse the impact of inaccurate location information in geographic routing, which is caused by mobility of nodes. Zhou et al. [238] investigate an effect of radio irregularity on the communication performance in WSNs. They analyze the impact of radio irregularity on some of the well-known MAC and routing protocols. A widely employed energy saving technique is to place nodes in sleep mode, corresponding to low power consumption and reduced operational capabilities. Chiasserini et al. [51] use a Markov model of a sensor network whose nodes may enter a sleep mode and investigate the system performance in terms of energy consumption, network capacity, and data delivery delay.

4.2.6 Group Management

A collaborative group is a useful concept to form groups by geographic proximity, roles, or resource availability. However managing groups in a distributed manner requires reliable communication and consensus. Kumer et al. [128] describe a data aggregation and consensus algorithm for object location and tracking applications in WSNs. This consensus algorithm permits ad hoc, in-network, group formation in response to a detected event. By reaching a consensus in the network, only a single message needs to be sent leading to significant savings in communication costs and prolonging the life of the network. The event is forwarded to the tracking application at a base station.

5. PROGRAMMING PARADIGMS

The area of high level languages for programming diverse, distributed networks of sensors has begun to receive attention during the last few years. This section describes research trends in programming models for WSNs.

Currently, the majority of sensor network applications are implemented as complex, low level programs that specify the behaviour of individual sensor nodes. The most popular platform currently available for WSN programming is TinyOS. TinyOS provides a component based architecture that targets resource constrained nodes by offering only a limited set of services, disallowing dynamic allocation, and providing a simple concurrency model. Typically the same application is deployed on every node.

The embedded nature of WSNs requires the propagation of new program code over the network. For example, [105] addresses network programming (the programming of nodes by disseminating code over the network). Each node maintains the most recent data by periodic broadcast. Distributed virtual machines is introduced to provide convenient high-level abstractions to application programmers, while implementing low level distributed protocols transparently in an efficient manner. This approach is taken in MagnetOS [142], which exports the illusion of a single Java virtual machine on top of a distributed sensor network. The application programmer writes a single Java program. The runtime system is responsible for code partitioning, placement, and automatic migration such that total energy consumption is minimized.

The Maté virtual machine takes the same approach, being built on top of TinyOS, but allows more flexibility in terms of code mobility than TinyOS. This is achieved by allowing code to move and be updated through the network. However, Maté's ability to allow code mobility is highly limited such as to version updating.

While TinyOS targets small, highly resource constrained nodes, other programming frameworks instead address the needs of systems with larger, more capable nodes. SensorWare, a sensor network programming framework [33], addresses some of the challenges of mobile code by expressing its sensor network tasks as Tcl scripts written in a *tasking language*. Scripts can move from node to node through the network acting as mobile code based on an agent based model.

Thus far there is no agreed solution. A high-level, global programming model, where the application can be specified in terms of system wide behaviour, which is then compiled down to the per-device program is desirable. The challenge in programming for WSNs is to coordinate the sensing, coolaborative processing, and data flow in the network, so that the required functionality is achieved. When an end user manually translates the global application behaviour in terms of local actions at each node, application logic will be tightly linked with the part of the program that coordinates lower level services such as resource management, routing, localization and so forth. This lack of separation between system level code and application level code results in high complexity of programming for WSNs. The limited computation, communication, and energy available on individual sensor nodes makes for difficult programming: that of decomposing complex, systemwide behaviour into the local actions to be taken at each node. This approach is top down and has the potential to significantly ease the burden of programming these large, complex systems.

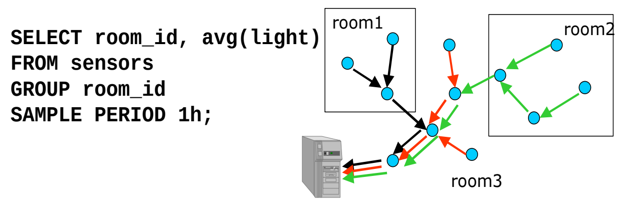
Programming for sensor networks brings two main issues; programming abstractions and programming support. The former is focused on providing programmers with abstractions of sensors and sensor data. The latter is providing additional runtime mechanisms that simplify program execution. Examples of such mechanisms include safe code execution, or reliable code distribution.

Furthermore, in program development for sensor networks, it is impossible to debug on the real sensor network, where devices are resource poor and crash prone, and it is highly distributed with a large amount of information sharing and cooperative processing. Efforts for the simulation environment to confirm how the sensor network operates beforehand are TOSSIM simulation Environments [133] and Power TOSSIM [196]).

5.1 Programming Abstraction

Programming abstractions fall into two categories; application level and system level. The former defines and manipulates events at the desired level of semantic abstraction (e.g., *latest position of target*, *location of all faulty sensors*). The latter precisely specifies distributed computation and communication (e.g., *apply f(x) to all x within 10m*, *send data item d to 10 nearest nodes*) The tradeoffs between these two models are expressiveness, efficiency, reusability and automation. Existing programming abstractions can be classified into the following categories.

Data Based Model: The model follows a traditional query based approach such as TinyDB [146]. The code is split into two parts. The server side code deals with query parsing, query planning and optimisation, while the sensor side code is responsible for routing, query aggregation, partial data aggregation, and lifetime specification, and so forth. The approach is a straightforward extension of traditional queries, with sensor network specific modification to the query language, and to incorporate message routing and data aggregation. This model is applied to collect streaming numeric data.



```
SELECT room_id, avg(light)
FROM sensors
GROUP room_id
SAMPLE PERIOD 1h;
```

Figure 5: Database based Model

One of the more intriguing suggestions is cross layer modification to achieve better efficiency. This approach is not popular in traditional layered network protocol models, but may merit considerations in this case due to the limitation in resources and the pursuit of lightweight implementation. Fig.5 shows WSN as a distributed database.

Agent Based Model: When writing an agent, the programmer can read or write the agents local variables both on the heap and in the nodes local variables. The agent can move to another node. An agent could read sensor values, actuate devices, and send arbitrary radio packets. Under this model, arbitrary algorithms can be implemented. Agent propagation (in a sense, routing) is a decision made by the programmer and coded into the agents self forwarding logic. The actuators in sensor networks need cooperative control to achieve the common goal. How sensing and actuating units should specify the conditions and reactions to follow can be defined in a formal language. The use of such a language eases the programming of a distributed system, and provides simplicity by formal analysis and automated reasoning. Fig.6 shows mobile agent (active sensor) model.

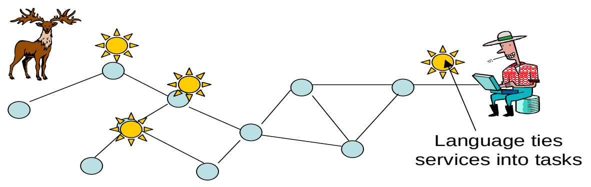


Figure 6: Mobile Agent Model

Macroprogramming Model: Another approach to programming large, complex sensor networks is *macro-programming*, which considers a WSN's global behaviour, rather than the low level actions of individual nodes. End users specify tasks at a higher level, where the embedded system's details of node communication protocols are hidden. PARCs PIECES framework [139] presents a *state centric* abstraction for programming a sensor network. PIECES emphasizes a state as the core concept in the model. It has the notion of collaboration groups that abstracts out common patterns in application specific communication and resource allocation. The number of nodes is typically quite large in sensor networks, and the complexity resulting from the large number of participants makes some form of higher level abstraction a necessity. The sensors are divided into groups based on their locations or functionalities, which allows programmers to deal with nodes as a group. The concept of principal is used to keep and maintain the state associated with physical phenomena, and each state has only one principal that stores, updates, and responds to queries as to the value of the state. The task of programming the system becomes the task of how to define interaction between principals without worrying about the low level, per node operation and hurdles. An application developer specifies a computation as the creation, combination, and transformation of states, which naturally map to the vocabulary used by signal processing and control engineers. More specifically, an application program is written as algorithms for state update and observation, with input supplied by dynamically created collaboration groups. This higher level of abstraction allows domain experts not familiar with programming to define the system behaviour using the terms they are familiar with, namely the various states. This approach bridges the gap between node centric programming and high-level processing. In contrast, Hood [222] proposes Neighbourhood oriented algorithms, where a node can identify a subset of nodes around it by a variety of criteria and share state with those nodes. This abstraction allows developers to design distributed algorithms in terms of the neighbourhood abstraction itself, instead of decomposing them into component parts such as messaging protocols, data caches, and neighbour lists. In applications that are already neighbourhood based, this abstraction is shown to facilitate good application design and to reduce algorithmic complexity, inter component coupling, and total lines of code.

The above classification is one view, at an early stage of research in this area. Fig.7 shows an example of a holistic view of the various functionalities in WSNs.

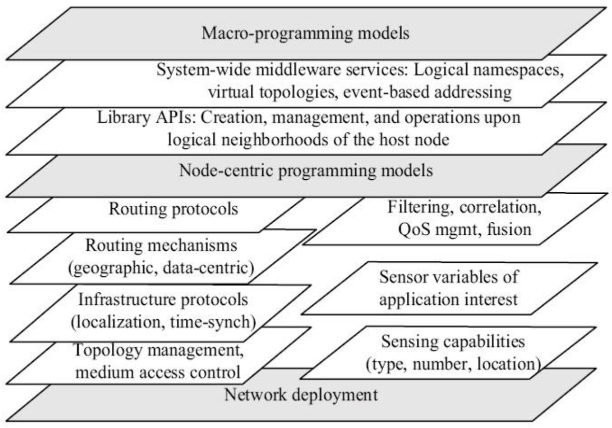

Macro-programming models
System-wide middleware services: Logical namespaces, virtual topologies, event-based addressing
Library APIs: Creation, management, and operations upon logical neighborhoods of the host node
Node-centric programming models
Routing protocols
Filtering, correlation, QoS mgmt, fusion
Routing mechanisms (geographic, data-centric)
Sensor variables of application interest
Infrastructure protocols (localization, time-synch)
Topology management, medium access control
Sensing capabilities (type, number, location)
Network deployment

Figure 7: Layers of abstraction for application development on WSNs
(from [18])

This wide range of programming abstractions has introduced ideas such as in-network query processing [146, 145, 33], event-based processing [120], native threaded virtual machines [220, 147, 183], and on-node script interpreting [33]. examples of macroprogramming methodologies for WSNs are TinyDB [146], Regiment [163], Kairos [79], ATaG [17], SensorWare [33], market based macroprogramming's pricing [147], diffusion's aggregation [108], and Semantic Streams [221]. TinyDB provides a declarative SQL-like query interface for sensor data, and ATaG offers a mixed imperative declarative programming style and data driven flow. Regiment is a demand driven functional language with support for region based aggregation. Karios is an imperative, control driven programming paradigm providing a distributed shared memory abstraction to node level programming. Semantic Streams' markup and declarative query language, based on Prolog, is used to specify queries over semantic information. The selection, writing and optimisation of low level modules to implement the querying is performed by a service composition framework, as described later. Complementary to the work on programming abstractions is network programming. Such systems enable high-level composition of sensor network applications (Sensorware [33] and SNACK [76]), efficient distribution of code (Deluge [105]), support for sandboxed application execution (Maté [132]), and techniques for automatic performance adaptation (Impala [140]). Rather than define a programming model, Application Specific Virtual Machines ASVMs [134] provide a way to implement and build the runtime underlying whichever model a user needs.

For self configuration, various approaches are devised. Examples include coverage [202]; aggregator placement [31]; clustering, routing and addressing [125, 205, 212]. [125] uses a fixed set of roles to build a network wide backbone infrastructure. However, none of these approaches are generic frameworks that support the assignment of user defined roles in an application specific manner. Only recently, neighbourhood programming abstractions [220, 222] have been proposed, where network neighbours can easily share variables.

A goal of macroprogramming is to simplify sensor network application design by providing high-level programming abstractions and primitives that automatically compile down to the complex, low level operations implemented by each sensor node.

The current research on programming models focusses on the configuration, propagation, and aggregation of WSNs. Within the proposed models, function units of sensor nodes are defined as roles, tasks, or services. The high-level representation of problems can be queries, service requests, composite events, or regular expressions. Expressiveness of high-level languages is an important aspect, and a distributed data processing framework is desirable. One of important issues is that end users' semantic requests must be translated correctly into the operations in WSNs through programming. This requires expressiveness of languages and accurate mapping between requests and operations, and it may require ontology involvement. Semantic Web Services (SWS) address similar problems in their domain. Knowledge may be produced by the pattern of sensing behaviours, too. In SWS, semantically described modular programs are created so that they can automatically compose new services from the modular components. In WSNs thus far, in many research prototypes, queries have been simple enough to be decomposed. When real world applications use macroprogramming, this will be an issue to be solved.

5.2 Amorphous Computing

Sensors can detect atomic pieces of information, and the information gathered from different devices will be analyzed and provide data that was impossible to obtain without these technologies. Combining regional sensed data from the different locations may spawn further information. Localized algorithms, in which simple local node behaviour achieves a desired global object, may be necessary for sensor network coordination. Modelling such systems is attempted by studying biological systems, distributed robotics, and amorphous computing. Amorphous computing, proposed by Abelson et al. [2], aims at discovering new approaches for programming and controlling a vast number of unreliable parts to achieve emergent global behaviours, such as some desired patterns. These are called computational entities, which are usually irregularly placed and locally interacting. This technology is especially suitable for such environments where the desired global behaviours can only be achieved based on local information and interactions among entities. [160] demonstrates how to form coordinate systems, arbitrary two and three dimensional shapes, arbitrary graphs of *wires*, and origami like folding patterns. Yet the Amorphous Computing effort has not to date provided a model for programming rather than pattern formation.

5.3 Existing Programming Models

In this section we describe existing programming models and paradigms (see also *Middleware Technology* in section 6).

Recent macroprogramming may fall into two classes: one focuses on providing the programmer abstractions that simplify the task of specifying nodes' local behaviour within a distributed computation, while the second enables programmers to express the global behaviour of the distributed computation. In the former class, three different types of programming abstractions have been explored. For example, Liu et al. [140] and Cheong et al. [50] have considered node group abstractions that permit programmers to express communication within groups sharing some common group state. Data centric mechanisms are used to efficiently implement these abstractions. By contrast, Mainland et al. [220] and Whitehouse et al. [222] show that topologically defined group abstractions (*neighbourhoods* and *regions* respectively) are capable of expressing a number of local behaviours powerfully. Finally, the work on EnviroTrack [1] provides abstractions for physical objects in the environment, enabling programmers to express tracking applications.

5.3.1 *Kairos*

Kairos [79] offers a network programming model that allows the programmer to express, in a centralized fashion, the desired global behaviour of a distributed computation on the entire sensor network. Kairos compile time and runtime subsystems expose a small set of programming primitives, while hiding from the programmer the details of distributed code generation and instantiation, remote data access and management, and inter node program flow coordination.

Kairos provides abstractions for expressing the global behaviour of distributed computations. Kairos does not contain explicit abstractions for nodes, but rather expresses a distributed computation in a network independent way. Thus, it is similar to the work on SQL-like expressive but Turing incomplete query systems (e.g., TinyDB [146, 145] and Cougar [61]).

DFuse [126] provides support for expressing computations over logical topologies or task graphs, which are then dynamically mapped to a network instances. Exporting the network topology as an abstraction can impose some rigidity in the programming model. It can also add complexity to maintaining the mapping between the logical and the physical topology when nodes fail. Complementary to these approaches, node dependent abstractions allow a programmer to express the global behaviour of a distributed computation in terms of nodes and node state. This is an approach that Kairos is using.

Regiment [163] takes a similar approach to Kairos. While Kairos focuses on a narrow set of flexible, language-agnostic abstractions, Regiment focuses on exploring how functional programming paradigms might be applied to programming sensor networks in the large, while Split-C [239] provides *split* local global address spaces to ease parallel programming. Kairos provides these through the remote variable access facility, but confines itself to the *C*

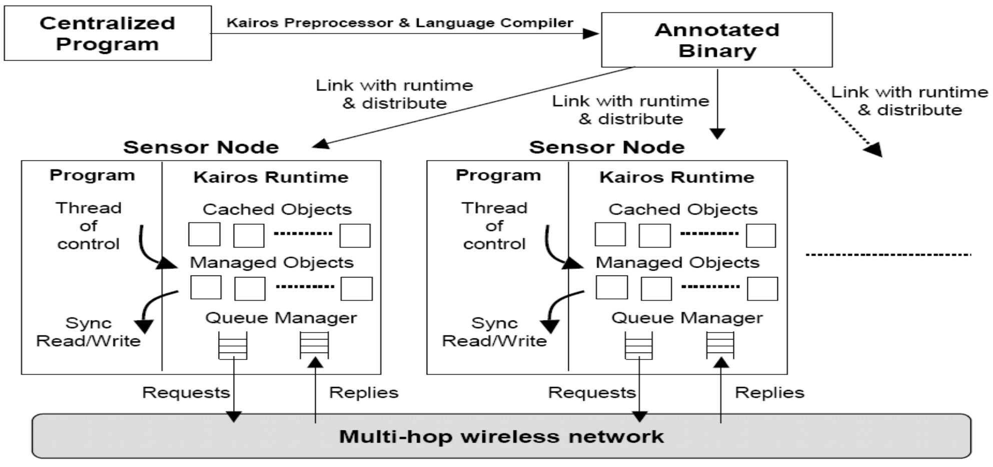


Figure 8: Programming Architecture in Kairos
(from [79])

language that lacks a rich object oriented data model and a language level concurrency model. Therefore, the fundamental concepts in these two works are language specific. Fig.8 shows an overview of Kairos Programming Architecture.

5.3.2 Abstract Task Graph

The Abstract Task Graph (ATaG) [17] is a data driven programming model for end-to-end application development on networked sensor systems. An ATaG program is a system level, architecture independent specification of the application functionality.

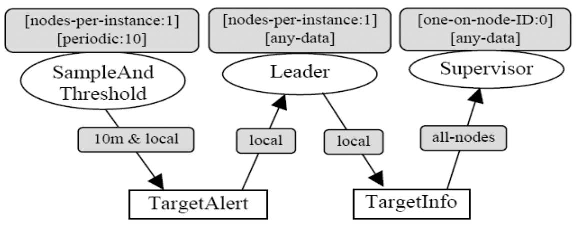

[nodes-per-instance:1] [periodic:10]
[nodes-per-instance:1] [any-data]
[one-on-node-ID:0] [any-data]
SampleAnd Threshold
Leader
Supervisor
10m & local
local
local
all-nodes
TargetAlert
TargetInfo

Figure 9: Object Tracking in ATaG
(from [17])

The application is modelled as a set of abstract tasks that represent types of information processing functions in the system, and a set of abstract data items that represent types of information exchanged between abstract tasks. Input and output relationships between abstract tasks and data items are explicitly indicated as channels. Each abstract task is associated with user-provided code that implements the information processing functions in the system. Appropriate numbers and types of tasks can then be instantiated at compile time or runtime to match the hardware and network configuration, with each node incorporating the user-provided code, automatically generated glue code, and a runtime engine that manages all coordination and communication in the network. Fig.9 shows a program for an object tracking. The ATaG programmer first models each behaviour in terms of a pattern of node level interaction. The next step is to identify the types of processing and the types of data in the system. *SampleAndThreshold*, *Leader*, and *Supervisor* are defined as abstract tasks, while *TargetAlert* and *TargetInfo* are abstract data. The input/output interfaces of the abstract tasks are shown in Fig.9. The final step is to associate annotations (shaded rounded rectangles) to indicate task placement and information flow patterns. In this example, *TargetAlert* is produced only if *SampleAndThreshold* detects an object, then *TargetAlert* is sent to all other nodes that might have detected the object. *Leader* determines if its own reading is the maximum of all readings received and *TargetInfo* is produced only if a node decides its own reading is the highest.

5.3.3 Active Sensor Networks

Rather than proposing a new programming approach to in-network processing, Active Sensor Networks [134] proposes an architecture for implementing a programming model's underlying runtime. The Maté virtual machine (a tiny bytecode interpreter) [132] is extended by generalizing its simple VM into an architecture for building application specific virtual machines (ASVMs). ASVMs address three of Maté's main limitations: flexibility, concurrency, and propagation.

Introducing lightweight scripting to a network makes it easy to process data at, or very close to, its source. This processing can improve network lifetime by reducing network traffic, and can improve scalability by performing local operations locally. Similar approaches have appeared before in other domains. Active disks proposed pushing computation close to storage as a way to deal with bandwidth limitations, active networks argued for introducing in-network processing to the Internet to aid the deployment of new network protocols, and active services suggested processing at IP end points.

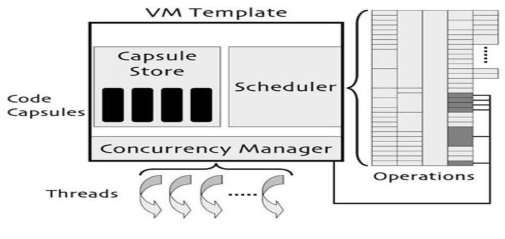


Figure 10: ASVM architecture
(from [221])

Active Sensor Network introduces dynamic computation into a sensor network. Active networking is most similar, but the differing goals and constraints of the Internet and sensor networks lead to very different solutions. Fig.10 shows the ASVM functional decomposition. ASVMs have three major abstractions: handlers, operations and capsules. Handlers are code routines that run in response to system events, operations are the units of execution functionality, and capsules are the units of code propagation. ASVMs have a threaded execution model and a stack based architecture.

5.3.4 Object State Machine (OSM)

Kasten et al. propose Object State Machine (OSM) [120], a programming model and language for sensor nodes based on finite state machines. OSM provides an abstraction for managing the complexity of event triggered programming. OSM extends the event paradigm with states and transitions, such that the invocation of actions becomes a function of both the event and the program state. OSM introduces state attributes that allow sharing of information among actions. They can be considered local variables of a state with support for automatic memory management. OSM specifications can be compiled into sequential C code that requires only minimal runtime support, resulting in efficient and compact systems. OSM borrows the concept of hierarchical and parallel composition of state machines from Statecharts [83] as well as the concept of broadcast communication of events within the state machine. From SyncCharts [12] it adopted the concept of concurrent events. Variables are typically not fundamental entities in control oriented FSM models. Models that focus both on the transformative domain (data processing, stream processing) and the control oriented domain, typically include variables as intrinsic entities. Finite State Machines with Datapath (FSMD) [60] introduced variables to the FSM model in order to reduce the number of states that have to be declared explicitly. Like OSM, this model allows programmers to choose to specify program state explicitly (with machine states) or implicitly with variables. FSMD are flat, that is, they do not support hierarchy and concurrency, and variables have global scope and lifetime. On the contrary, variables in OSM are bound to a state hierarchy. OSM allows access to the values of events in computational actions. A valued event is visible in the scope of both the source and the target state of a transition (in out and in actions, respectively). A number of frameworks for programming individual sensor nodes have been proposed. With one exception, all frameworks fall into one of two basic categories: event-based systems (such as Maté [132]) and multi-threaded systems (SensorWare [33]). Applications built according to either model have an implicit notion of program state. In contrast, in OSM program state can be modelled explicitly.

5.3.5 Regiment

Regiment [163, 164] is a functional macroprogramming language for sensor networks. The basic concept is similar to Kairos [79] providing programming abstraction by expressing the global behaviour of distributed computations. The essential data model in Regiment is based on region streams, which represent spatially distributed, time varying collections of node state. A region stream might represent the set of sensor values across all nodes in an area or the aggregation of sensor values within that area. Regions provide a means of expression spatial and logical relationships between sensor nodes, transparent data sharing between nodes, and efficient reduction operations within regions. Abstract regions expose the tradeoff between resource usage and the accuracy of collective operations, allowing applications to tune energy and bandwidth consumption to meet accuracy targets.

Regiment is a purely functional language, which gives the compiler considerable leeway in terms of realizing region stream operations across sensor nodes and exploiting redundancy within the network. Regiment allows the user to perform general operations (e.g., MAP, FOLD, and FILTER), which map a function over, aggregate over, or filter all the data in a region. The system determines where and when data is stored and operations are performed in the network.

5.3.6 Semantic Streams

Semantic Streams [221] is a framework that allows users to pose declarative queries over semantic interpretations of sensor data. For example, instead of querying raw sensor data, the user can query vehicle speeds; the system decides which sensor data and which operations to use to infer the vehicle speeds. The user can also place constraints on values such as the confidence with which the speed was measured or the amount of energy consumed to measure the speeds. This framework is designed to work in a shared sensor infrastructure, where multiple queries may coexist for extended periods of time, instead of a hand designed, single purpose sensor network. Semantic Streams takes a semantic service programming model and provides a service description language and a query processor that support the programming model.

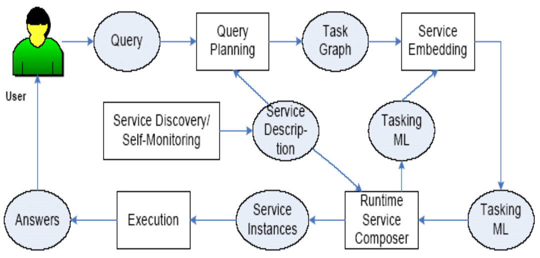


Figure 11: Planning and Execution in Semantic Streams

(from [221])

Semantic Streams is similar to the approaches described above in that the user issues a query specifying global behaviour. One main difference is that the user is required to understand which operations to run over the raw sensor data and how to interpret the meaning of the results. Semantic Streams allows the user to issue queries over semantic values directly without addressing which data or operations are to be used. The advantages of semantic queries are analogous to those of macroprogramming in general: the user of macroprogramming need not specify the best time and place to execute each operation, while the user of semantic queries need not specify which operations to run or which data to run them over. This allows the user to make less low level decisions while allowing the system an extra degree of freedom to optimise during execution. In Fig.11, the user first poses a query to the query processor, which derives an acceptable service graph. That graph is passed to the execution engine along with all variable unification and constraint sets resulting from planning. The execution engine may call back to the query processor during replanning.

5.3.7 Abstract Region

TinyDB [146], Cougar [108], and IrisNet [161] provide a high-level SQL or XML based query interface to sensor network data. Queries are deployed into the network, streaming results to one or more base stations, and aggregation is used to reduce communication overhead. systems have tremendous value and abstract away many details of communication, aggregation, and filtering. However, they are not well-suited for developers who wish to implement specific behaviour at a lower level than the query interface. For example, TinyDB is focused on relaying aggregate data along a spanning tree rooted at a base station. While this mechanism can support complex algorithms such as contour finding, TinyDB must expose an internal *contour finding operator* to queries. In [220], *Abstract Regions* ars proposed, providing a set of communication abstractions that can be used to implement higher-level services, such as queries. Their ultimate aim is creating a framework for programming sensor networks, and using abstract regions as a building block for a high-level programming language for sensor networks. They define *Region* as a group of geographically or topological related nodes. The essential idea is to capture communication patterns, locality, and resource tradeoff in a high level language that compiles down to the detailed behaviour of individual nodes. Shielding programmers from the details of message routing, in-network aggregation, and achieving a given fidelity under a fixed resource budget should greatly simplify application development for this new domain. At the same time, the communication abstraction should yield control over resource usage and make it possible for applications to balance the tradeoff between energy/bandwidth consumption and the accuracy of collective operations. The notion of communicating within, and computing across, a neighbourhood (for a range of definitions of *neighbourhood*) is a useful concept for sensor applications. Similar concepts are evident in other communication models for sensor networks, although often exposed at a much higher level of abstraction. For example, directed diffusion [108] and TinyDB [146] embody similar concepts but lump them together with additional semantics. Abstract regions are fairly low level and are intended to serve as building blocks for these higher-level systems. Abstract regions can be used to implement a form of directed diffusion. Other communication abstractions include GHT [176], Spatial Programming [30], DIFS [75], and SPIN [90]. These systems are generally focused on a specific communication or aggregation model rather than supporting a wide range of applications.

5.3.8 Market Based Macroprogramming (MBM)

Market based macroprogramming (MBM) [147] is a new paradigm for achieving globally efficient behaviour in sensor networks. Rather than programming the individual, low level behaviours of sensor nodes, MBM defines a virtual market where nodes sell *actions* (such as taking a sensor reading or aggregating data) in response to global price information. Nodes take actions to maximize their own utility, subject to energy budget constraints. The behaviour of the network is determined by adjusting the price vectors for each action, rather than by directly specifying local node actions, resulting in a globally efficient allocation of network resources. In this approach, individual sensor nodes act as self interested agents that operate in a virtual market, and receive profit for performing simple, local actions in response to globally-advertised price information. Sensor nodes run a very simple cost evaluation function, and global behaviour is induced throughout the network by advertising price information that drives nodes to react. The prices can be dynamically tuned by the centralized market maker to meet systemwide goals of lifetime, accuracy, or latency based on the needs of the sensor network programmer.

5.3.9 Generic Role Assignment

Römer et al. have examined Generic Role Assignment [183], where sensor nodes are assigned user defined roles based on their capabilities. Almost any sensor network application requires some form of self configuration, where sensor nodes take on specific functions or roles in the network without manual intervention. These roles may be based on varying sensor node properties (e.g. available sensors, location, network neighbours) and may be used to support applications requiring heterogeneous node functionality (e.g., clustering, data aggregation). Their approach of role assignment is similar to cellular automata, where the state of a particle in a regular arrangement is completely defined by the previous values of a neighbourhood of particles around it. They argue that the assignment of user defined roles is a fundamental part of a wide range of sensor network applications. Consequently, a framework for assigning roles to sensor nodes in an application-specific manner could significantly ease sensor network programming, and outline the general structure of such a framework with a first approach to its realization. One possible way to implement Römer et al.'s approach could be on top of neighbourhood programming abstractions such as Abstract Region [220] or Hood [222]. Fig.12 shows an overview of Generic Role Assignment architecture.

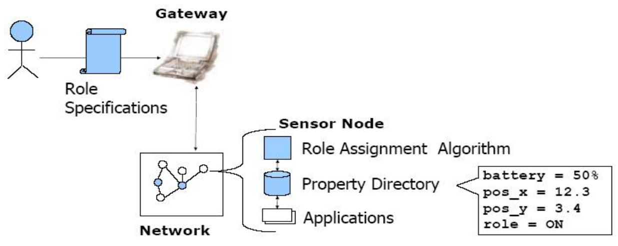


Figure 12: Generic Role Assignment Architecture (from [183])

5.3.10 Declarative Resource Naming (DRN)

Declarative Resource Naming (DRN) [109] is influenced by Spatial Programming (SP) [30, 88]. SP shares a vision of programming for wireless networks of embedded systems as a unit, simplifying resource accesses as variable accesses, exposing the space property to the programmers, hiding network details, and supporting imperative programming. Spatial Programming uses Smart Messages to provide content-based spatial references to embedded resources. To simplify the programming of Wireless Networks of Embedded Systems (WNES) [109] propose

Declarative Resource Naming (DRN) to program WNES as a whole (i.e., macroprogramming) instead of several network-ed entities. DRN allows programmers to describe declaratively a set of desired resources by their runtime properties and to map this set to a variable. Using DRN, resource accesses are simplified to completely network-transparent accesses of variables. DRN provides both individual and group accesses to the desired set. Group accesses (i.e., parallel accesses) reduce total access time and energy consumption because of possible in-network processing. Additionally, we can associate each set with tuning parameters (e.g., timeout, energy budget) to bound access time or to tune resource consumption. However, SP supports only sequential resource accesses whereas DRN supports both sequential and parallel accesses. Accessing resources in parallel can significantly reduce the total access time and the overall energy consumption (by enabling in-network processing). Additionally, SP is purely imperative programming but DRN is a hybrid between declarative programming and imperative programming. Unlike the DRN binding, the SP binding is, by default, static, even though dynamic binding in SP is provided as an option. This problem is similar to that of TCP. Packet loss and unbound acknowledgement delay are handled using timeout.

5.3.11 *Maté*

Maté [132] is a byte code interpreter (VM) running on TinyOS, that provides safe program execution environments, runtime re-programming, and an event-driven stack-based architecture. The interpreter provides high-level instructions (such as an atomic message send) which the machine can interpret and execute. Each virtual machine instruction executes in its own TinyOS task. Code is broken into capsules and it operates self distribution (diffusion) controlling identification and versions. However it maintains a static set of execution events, limited assembly level programming and single, centralized shared variables. It also provides reliability. The process of re-programming a sensor network is simple using Maté. When a new program is created and is to be deployed, it is given a newer version number and broadcast to the network. Once a capsule with a newer version is received, the mote will install it and forward it on to its neighbours. Over time the new program will disseminate through the network via broadcast.

6. MIDDLEWARE TECHNOLOGY

Middleware usually lies between the operating system and the application in traditional environments, where functionality of the operating system is well established. For WSNs, however, the interfaces of operating systems are still a research issue, and many applications execute hardware operations directly without operating system components. Thus, middleware for WSNs needs to have a clear future vision so that all technologies supporting WSNs fit properly with future middleware. In general, middleware for sensor networks can be defined as software that provides data aggregation, control and management mechanism adapting to the target application's need, where data are collected from sensor networks. Existing research on sensor networks has focused on the hardware technology, thus a bottom-up approach has been pursued. However, recent progress in sensor network technology, and an expansion of the associated research community, has brought a consensus that a general technique is required for accessing sensor data from external applications. But a middleware as a service-oriented, top-down, standard framework, linked with external applications, has not yet been advocated. Recent evolution of sensor network technology highlights the importance of data aggregation and management. This section describes the current state of middleware research on sensor networks.

Blumenthal [29] describes the task of middleware for WSN is providing easy access from the complex sensor networks, and followings are four key requirements:

- Scalable for resource constraints.
- Generic for different applications with common interface.
- Adaptive to reconfigure its structure.
- Reflective to change the behaviour depending on the environments and circumstances.

Römer [181] emphasises:

- Event-based and data centric communication: Event based reactive communication protocols and content-based communication are necessary, as for the WWW.
- A holistic view of the Internet and sensor networks: The scope of middleware is not only to aggregate information from sensor nodes but also to cover devices and networks connected to the WSN.
- Application knowledge in nodes: A mechanism to embed the application knowledge into the infrastructure and the WSN should be provided.
- An adaptive fidelity algorithm: This requires infrastructure to provide appropriate mechanisms for selecting parameters, or whole algorithms, which solve a certain problem with the best quality under given resource constraints.
- Automatic configuration: WSN nodes must operate unattended, which means that middleware for WSNs has to provide new levels of support for automatic configuration and error handling.

Yu [234] describes design principles as follows:

- Data centric: Data centric mechanisms for data processing and querying within the network should be provided. Due to its simplicity, flexibility, and robustness, cluster based network architecture has been widely used in the design and implementation of network protocols and collaborative signal processing applications for WSNs. A cluster based architecture is suitable for hosting the data centric processing paradigm from both geographical and system design perspectives.
- Application knowledge: Integrating knowledge of applications into the services provided by the middleware is important. A tradeoff needs to be explored between application specificity and generality of the middleware.
- Localized algorithms: Localized algorithms should be used collectively to achieve a desired global objective while providing system scalability and robustness. Since the cluster based architecture localizes

the interaction of sensor nodes, and hence the coordination and control overhead within a restricted vicinity, it is reasonable to regard each cluster as a basic function unit of the middleware. Consequently, the middleware performs as a distributed software composed of multiple clusters.

- Lightweight: Middleware should be lightweight in terms of the computation and communication requirements.
- Trading QoS of application: Resource sharing when resources are limited means that it is very likely that the performance requirements of all the running applications cannot be satisfied simultaneously. Therefore, it is necessary for the middleware to trade the QoS of various applications against each other.

The main functionality of middleware for sensor networks is to support the development, maintenance, deployment, and execution of sensing based applications. This includes mechanisms for formulating complex high-level sensing tasks, communicating these tasks to the WSN, coordination of sensor nodes to split a task and distribute it to the individual sensor nodes, data fusion for merging the sensor readings of the individual sensor nodes into a high-level result, and reporting the result back to the task issuer. Moreover, appropriate abstractions and mechanisms for dealing with the heterogeneity of sensor nodes should be provided.

In this section, we introduce existing middlewares for sensor networks based on the design principles described above. We classify middleware into several categories: First, with the distributed database approach, a data driven approach is described. Secondly, an event-based approach, to deal with real-time data in the sensor networks, and thirdly QoS oriented middleware is described. Fourthly, Internet oriented middleware aims at constructing the infrastructure of information processing and fifthly, an agent based approach is described Finally, we include traditional centralized sensor systems for contrast. All the approaches attempt to address the resource constrained nature of sensor network environments. The classification in this section is based on the the design principles discussed in previous sections. We focus on the application interface and the required resources, see also the programming model in section 5.

6.1 Data Driven Approach

This section introduces the research on a middleware framework based on the data driven approach. In future, a large number of sensor devices will be deployed, and the management of their data will become complex. The user should be able to access the necessary data without knowledge of the sensor network. Recently, the data driven approach to manage sensor data is increasing in popularity, where each node keeps the data, and those nodes execute retrieval and aggregation (in-network aggregation) with on-demand based operation to deliver the data to the external applications. This approach supports no asynchronous state formations. The data driven approach derives from a database abstraction, where applications define collective node states to represent portions of their world model. An important issue here is expressiveness to define certain collective node states based on the interface language, such as SQL queries. When queries become complex and dynamic, it will be too complex to bridge to atomic functions in WSN.

6.1.1 Cougar

In many existing sensor network frameworks, the sensor data has been assumed to be transmitted and stored in the gateway server according to a preprogrammed procedure. In such an approach, a different data collection scheme requires a change in the program. Moreover, the overhead due to load on the gateway server, and redundant data transmission, cannot be neglected. Cougar [61, 229, 32] is an architecture that treats a sensor network as a distributed database, where a large number of sensor nodes are connected through a multi-hop wireless network and each node keeps sensor data. A query optimiser is located on the gateway node to generate distributed query processing plans after receiving queries from outside. The query plan is created according to catalog information and the query specification. Such a query plan specifies both the data flow (between sensors) and an exact computation plan (for each sensor). The plan is then disseminated to all relevant sensor nodes. Control structures are created to synchronize sensor behaviour, and the query is started. At run time, data records flow back to the gateway node as in-network computation happens on-the-fly. Bonnet [32] characterises, and emphasizes spatial and temporal continuity about query processing in sensor networks for Cougar development.

- Monitoring queries are long running contnously.
- The desired result of a query is typically a series of notifications of system activity (periodic or triggered by special situations).
- Queries need to correlate data produced simultaneously by different sensors.
- Queries need to aggregate sensor data over time windows.
- Most queries contain some condition restricting the set of sensors that are involved (usually geographical conditions).

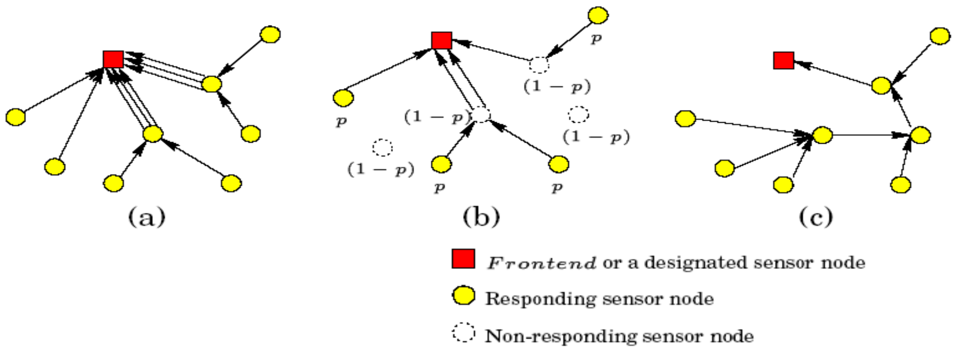
The usual relational database approach on query operation is not sufficient for real-time query processing, which the sensor network requires. Thus, in Cougar, it is proposed to divide the model of the sensor data into a user expression and an internal expression. First, the user expression is a query, and an Abstract Data Type (ADT) is defined for the sensor, and it proposes the query language of the syntax similar to SQL. For instance, query processing for a monitor can be described as follows.

$$SELECT\ R.s.getTemp()\ FROM\ R$$
$$WHERE\ R.floor\ =\ 3\ AND\ \$every(60);$$

R means the ADT of the sensor, and temperature is described as *(R.s.getTemp())* that is notified from the sensor, which exists on the third floor *(R.floor=3) (every(60))* every 60 seconds. Cougar interprets such an enquiry expression, and the mapping is done to an internal expression of the enquiry execution plan (Query Execution Plan). The query processing that continues timewise by executing this enquiry execution plan can be achieved.

6.1.2 SINA

SINA (Sensor Information Networking Architecture) [194, 209] is a middleware architecture that abstracts the network of a sensor node as a distributed object for query



(a)
(b)
(c)
p
(1 − p)
Frontend or a designated sensor node
Responding sensor node
Non-responding sensor node

Figure 13: Response Implosion in SINA
(from [209])

operation, and task allocation. An overview of the middleware architecture is shown in Fig. 14. SINA aims to achieve scalability and low power consumption in sensor networks. SINA consists of the following function components.

- Hierarchical clustering: The sensor nodes contain the function to build the hierarchical cluster structure dynamically.
- Attribute based Name management: The sensor node is managed by the name based on the attribute but note ID. For instance,
  $[type = temperature, location = NE, temperature = 103]$
  means all the sensors indicating 103 degrees in the northeast division.
- Position management: The position of the sensor node is measured, and managed by GPS etc.

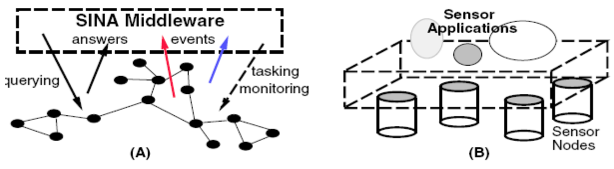


Figure 14: SINA Middleware Model
(from [194])

Using the above functions, the execution environments of SINA cooperate with each other among sensor nodes. This operation mechanism can be programmed by using SQTL (Sensor Query and Tasking Language). Internal processing of SQTL is converted into a script similar to the Query Execution Plan of Cougar, and executed in an execution engine: Sensor Execution Environment(SEE). SQTL is a query language that looks like SQL for the user application, and it can access sensor information by using SQTL.

The following three primitive operations aim to achieve effective information aggregation:

- Sampling Operation: When all the sensors respond simultaneously, there will be an explosion (response implosion) (Fig. 13 (a)), thus when the adjoining node responds, a probabilistic operation (Fig. 13 (b)) either to respond or not is done.
- Self orchestrated operation: An intentional operation delay for the response implosion.
- Diffused computation operation: An operation that gives restricted communication only with a node that is adjacent (Fig. 13 (c)).

6.1.3 TinyDB

TinyDB [145, 94, 223, 73, 146] is an enquiry processing system for sensor networks that operates on TinyOS. In TinyDB, the concept of query processing (acquisitional query processing(ACQP)) is introduced. When query processing occurs the sensor node performs sensing in order to respond to the query. High level queries are decomposed and distributed to the networks. It is necessary to write a program in the C language, corresponding to the content of the query on TinyOS and its processing. In ACQP of TinyDB, the SQL is enhanced for query processing and this query is converted to internal code, and executed for data retrieval and aggregation. For instance, the description that looks like the following SQL is used. Consider the following query:

$SELECTnodeid, light, temp$
$FROM\ sensors$
$SAMPLE\ PERIOD\ 1s\ FOR\ 10s$

This query specifies that each device should report its own id, light, and temperature readings (contained in the virtual table sensors) once per second for 10 seconds. Results of this query stream to the root of the network in an online fashion, via the multi-hop topology, where they may be logged or output to the user. The output consists of a stream of tuples, clustered into 1s time intervals. Each tuple includes a timestamp corresponding to the time it was produced. When a query is converted to internal codes, it optimises power consumption and communication in TinyDB. Fig. 15 lists the key new techniques introduced in TinyDB, summarizing what queries they apply to and when they are most useful.

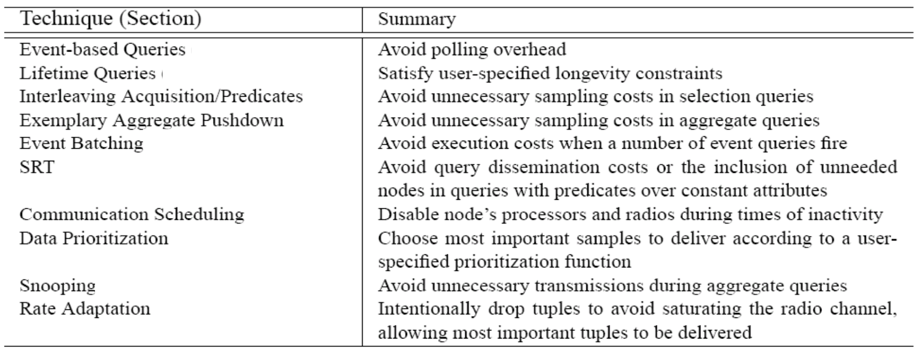
TinyDB approach is similar to Cougar, but it considers more optimisations in resource-constrained sensor network environments.

6.1.4 DFuse: A Framework for Distributed Data Fusion

DFuse [126] focuses on the following key problems:

- What are basic processing elements that compose the data fusion?
- How to assign the data aggregation tasks to specific sensor nodes dynamically?

DFuse is a framework for data fusion application development on decentralized distributed sensor networks. The architecture of DFuse emphasizes the following two characteristics:



Technique (Section)	Summary
Event-based Queries	Avoid polling overhead
Lifetime Queries	Satisfy user-specified longevity constraints
Interleaving Acquisition/Predicates	Avoid unnecessary sampling costs in selection queries
Exemplary Aggregate Pushdown	Avoid unnecessary sampling costs in aggregate queries
Event Batching	Avoid execution costs when a number of event queries fire
SRT	Avoid query dissemination costs or the inclusion of unneeded nodes in queries with predicates over constant attributes
Communication Scheduling	Disable node's processors and radios during times of inactivity
Data Prioritization	Choose most important samples to deliver according to a user-specified prioritization function
Snooping	Avoid unnecessary transmissions during aggregate queries
Rate Adaptation	Intentionally drop tuples to avoid saturating the radio channel, allowing most important tuples to be delivered

Figure 15: Summary of acquisitional query processing techniques in TinyDB (from [145])

Fusion API: The fusion API offers programming ease for a complex sensor fusion application. The API allows any synthesis operation on stream data to be specified as a fusion function, ranging from simple aggregation (such as min, max, sum, or concatenation) to more complex perception tasks (such as analyzing a sequence of video images). The API consists of structure management, correlation control, computation management, memory management, status and feedback handling, and failure/latency handling. Integrated operations on a variety of stream data over a complex aggregation process such as the analysis of the video images are enabled. In DFuse, the application is described as a brief data flow graph, called *Task Graph* where the Fusion API is used. The network is dynamically formed dynamically based on *Task Graph*, and optimisation is done afterwards by dynamic relocation described as follows.

A distributed algorithm for fusion function placement and dynamic relocation: There is a combinatorially large number of options for placing the fusion functions in the network. Hence, finding an optimal placement that minimizes communication is difficult. Also, the placement needs to be reevaluated quite frequently considering the dynamic nature of WSNs. Their approach is a heuristic based algorithm to find a good (according to some predefined cost function) mapping of fusion functions to the network nodes. The mapping is reevaluated periodically to address dynamic changes in nodes power levels and network behaviour.

DFuse uses a distributed role assignment algorithm for placing fusion points in the network. Role assignment is a mapping from a fusion point in an application task graph to a network node. The distributed role assignment algorithm is triggered at the root node. The inputs to the algorithm are an application task graph (assuming the source nodes are known), a cost function, and attributes specific to the cost function. The output is an overlay network that optimises the role to be performed by each node of the network. A network node can play one of three roles: end point (source or sink), relay, or fusion point. An end point corresponds to a data source or a sink. The network nodes that correspond to end points and fusion points may not always be directly reachable from one another. In this case, data forwarding relay nodes may be used to route messages among them. The routing layer is responsible for assigning a relay role to any network node. The role assignment algorithm assigns only the fusion point roles. The cost function includes *Minimize transmission cost (MT)*, *Minimize power variance (MPV)*, and *Minimize ratio of transmission cost to power (MTP)*. For example, a fusion function f with m input data sources (fan-in) and n output data consumers (fan-out), the transmission cost for placing f on node k is formulated as:

$$C_{MT}(k,f) = \sum_{i=1}^{m} t(source_i) * hopCount(input_i, k) + \sum_{j=1}^{n} t(f) * hopCpount(k, output_j)$$

Here, *t(x)* represents the transmission rate of the data source x, and *hopCount(i, k)* is the distance (in number of hops) between node i and k. The cost is optimised by moving the position at which a *Fusion Point* is allocated to minimize this cost function to the adjacent node. The example that the position of the node, where the *Fusion Point* is allocated, moves is shown in Fig. 16 Linear optimisation is performed in this example. If all the inputs to a fusion node are coming via a relay node (Figure 16(A)), and there is data contraction at the fusion point, then the relay node will become the new fusion node, and the old fusion node will transfer its responsibility to the new one (Figure 16(B)). In this case, the fusion point is moving away from the sink, and coming closer to the data source points. Similarly, if the output of the fusion node is going to a relay node, and there is data expansion, then again the relay node will act as the new fusion node. In this case, the fusion point is coming closer to the sink and moving away from the data source points.

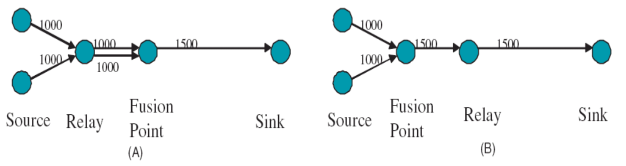


Figure 16: Linear Optimisation Example (from [126])

DFuse shows that the operation time of the network is extended by power saving and stabilization effects by introducing dynamic role allocation with an evaluation experiment with iPAQ.

6.1.5 TinyLIME

TinyLIME [57] is extended from the decentralized data sharing middleware LIME. LIME is an extension of Linda, providing memory sharing. TinyLIME is enhanced for a mobile environment for a sensor network platform.

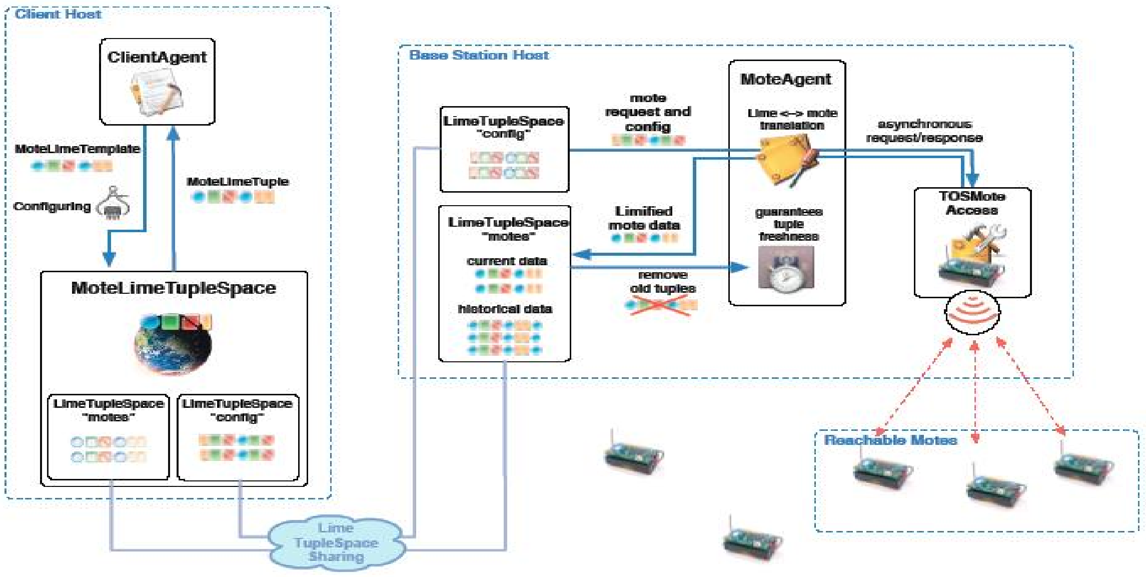


Figure 17: Architecture Overview of TinyLIME
(from [57])

Linda: Linda enables two or more systems to share a tuple space using reading (rd), writing (out) and deleting (in). For instance, $< “foo'', 9, 27.5 >$ can be written by the operation $out(< “foo'', 9, 27.5 >)$ and and it can be read by the operation $rd(< “foo'', ?integer, ?float >)$.

LIME: A coordinated tuple space is formed from the partitioned tuple spaces that each distributed system maintains. This occurs only when the maintaining nodes connect to the LIME system.

That is, communication via the shared memory space is possible only while the system maintains the mutual connections. For instance, the common tuple space that operates by building LIME into PDAs can be achieved only while the PDAs comprise an ad hoc network, and data exchange operations similar to Linda are possible.

Abstraction of sensor network: The TinyLIME model has been designed and implemented for the Crossbow Mote platform, exploiting the functionalities of TinyOS. On standard hosts, TinyLIME is implemented as a layer on top of LIME without requiring any modification, therefore reasserting the versatility of the LIME model and middleware. In TinyLIME, a tuple space partition resides on each sensor node, and a coordinated tuple space is formed when connecting it with the host with one hop or multi hop (see Fig.17). The sensor data can be read by executing the read operation of Linda such as *rd(p)* for this coordinated tuple space. Thus, TinyLIME can abstract the collection of data from the sensor network as an operation on the shared memory space (tuple space). This works as middleware by offering an abstracted interface to the application programmer in the sensor network.

In TinyLIME, the client host accesses the sensor node through the class named *MoteLimeTupleSpace*. One tuple is usually shown by <*sensor type, sensor value, number of the epoch of the sensor node, timestamp*>. The sensor data is read by executing the *rd* operation with template *MoteLimeTemplate* along with this sensor type. The retrieval is demanded from the tuple space of the Base Station Host when there is no data appropriate to the tuple space. The Base Station is regularly updated with sensor data and exports it to the tuple space.

6.2 Event Based Approach

A main purpose of sensor use in applications is preventing disasters and crimes by observing abnormalities. When an abnormality is observed, an alert should be raised in real-time to the user, and it requires event driven data processing mechanism. In another scenario, applications need to continuously collect and integrate data generated from a large and physically dispersed contingent of sensor nodes. There are many devices exchanging data, whilst some information sources and sinks may not be present in the network at that moment. Therefore, request/response communication is not adequate. For example, a client that requests instantaneous updates of information would need to continuously poll the information providers leading to network overload and congestion. Moreover, as energy is a scarce resource, unnecessary information requests should be avoided.

When designing real-time systems, the time triggered approach is expensive in the case where the expected rate of primitive event occurrence is low. An alternative is to use an event triggered approach where the execution is driven by the events. Event-driven communication is an asynchronous paradigm that decouples senders and receivers. Its clients are event publishers and event subscribers among which one-to-many and many-to-many communication is supported by a message transmission and notification service. An extension to this basic model allows messages to be associated with topics. In this case, subscribers only receive messages associated with the topic(s) to which they have subscribed. The publish/subscribe paradigm has become popular, because asynchronous and multipoint communication is well suited for constructing reactive distributed computing applications.

This section describes middleware that aims at event processing for sensor networks and some middleware that uses a publish/subscribe paradigm.

6.2.1 DSWare

Data Service Middleware (DSWare) [136] is middleware which takes a data-centric approach by defining the common data service and group based service parts of various applications. DSWare lies between the application layer and the network layer providing a data service abstraction to applications.

The real-time event service handles unreliability of individual sensor reports, correlation among different sensor observations, and inherent real-time characteristics of events. The event service supports confidence functions, which are based on data semantics, including the relative importance of sub events and historical patterns. When the failure rate is high, the event service enables partial detection of critical events to be reported in a timely manner. DSWare performs routing in real-time taking power consumption into account. DSWare consists of six function components as shown in Fig.18.

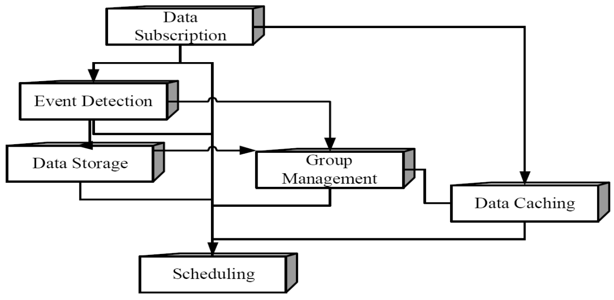


Figure 18: DSWare Framework
(from [136])

1) DataStorage: DSWare aims to distribute the data on specific sensor nodes or aggregated the data in groups for load balancing and to improve reliability.

Data that describes different occurrences of some type of activity can be mapped to certain locations so that future queries for this type of data do not require flooding to the whole network. The Data Storage Component in DSWare provides similar mechanisms to store information according to its semantics with efficient data lookup, and it is robust under node failure. Correlated data can be stored in geographically adjacent regions to enable possible aggregation and in-network processing.

2) Data Caching: The Data Caching Service provides multiple copies of the data most requested. This data is spread out over the routing path to reduce communication, increase availability and accelerate query execution. A simplified feedback control scheme is used to decide dynamically whether to place copies of the data around the frequently queried nodes.

3) Group Management: The Group Management component uses cooperation between group members to achieve reliability of sensor information and detection and exclusion of abnormal sensor nodes. Normally functioning sensors within a geographic area provide similar sensor values. A value that most nodes in a group agree on should have higher confidence than a value that is in dispute or varies widely. Based on similar observations by nearby sensors in a sufficiently dense area, erroneous results from the particular sensor nodes can be recognized. The suspicious nodes can be discarded in later coordination and computations in order to provide more reliable measurements. Some tasks require cooperation of multiple sensors. Movement and speed approximations require more than one sensor to combine their observations to calculate the direction and velocity. When a region has adequate density of sensors, a portion of them can be put into sleep mode to save energy.

4) Event Detection: An observation is the low level output of a sensing device during a sensing interval. It is a measurement of the environment. An event is an activity that can be monitored or detected in the environment and is of interest to the application. The types of event that can be detected and are potentially of interest are pre-registered according to the specific applications. Events are categorized into different types and may also be atomic events and compound events. An atomic event refers to an event that can be determined based only on an observation of a sensor.

Suppose we have registered the following events:
A high temperature event represents the observation that the temperature is higher than a specified threshold. A light event represents an occurrence of a sharp change in the light intensity. An acoustic event represents the occurrence of an unusual sound matching a certain signature. An explosion event might be defined as the three events above being reported in the same region within a specified time interval.

A confidence function specifies the relationships among sub events of a compound event with other factors that affect the decision such as relative importance, sensing reliability, historic data, statistical model, fitness of a known pattern and proximity of detections. In reality, an event always has meaningful contexts, which can be modelled using a Finite State Machine (FSM). For example, in a residential monitoring system, morning, afternoon, and evening can be states of this system. Moreover, each sub event has an absolute validity interval (avi) associated with it. The avi depicts the temporal consistency between the environment and its observed measurement. Continuing the explosion example, the temperature sub-event can have a longer avi because high temperature usually will last for a while, while the light sub-event may not last long because in an explosion, a sharp increase in the intensity of light would happen only for a short period of time. To register an event of interest, an application submits a request in the following SQL-like statement:

```
INSERT INTO EVENT_LIST
      (EVENT_ID, RANGE_TYPE, DETECTING_RANGE,
        SUBEVENT_SET, REGISTRANT_SET, REPORT_DEADLINE,
        DETECTION_DURATION [, SPATIAL_RESOLUTION ]
         [, ACTIONS])
VALUES      ()
```

The $RANGE\_TYPE$ can be $GROUP$ or $AREA$. The Detecting Range is the groups description (e.g., Group ID) or the areas coordinates range.The Subevent Set defines a set of sub events and their timing constraints, for instance as follows:

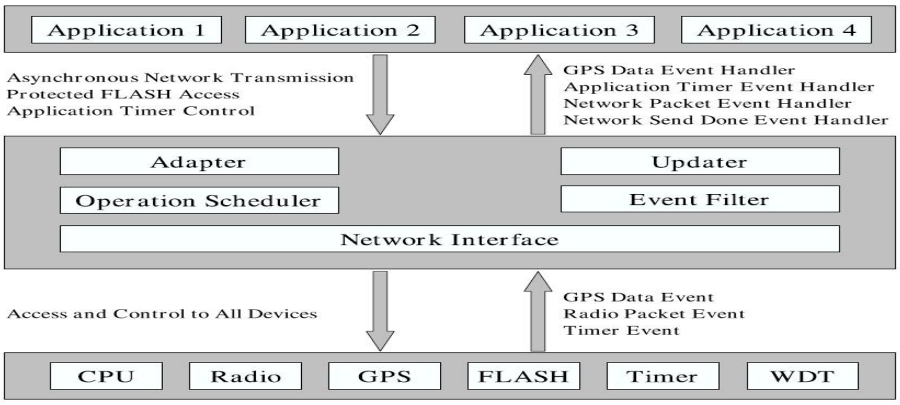


Figure 19: Layered architecture and interfaces in Impala
(from [141])

```
Subevent_Set { Time_window,
               Phase_set,
               Confidence_function,
               Min_confidence,
               (sub-event_1, avi1),
               [(sub-event_2, avi2),...]}
```

5) Data Subscription: As a type of data dissemination service, Data Subscription queries are very common in sensor networks. These queries have their own characteristics, including relatively fixed data feeding paths, stable traffic loads for nodes on the paths, and possible merges of multiple data feeding paths. When several base stations subscribe for the data from the same node at different rates, the *Data Subscription Service* places copies of the data at some intermediate nodes to minimize the total amount of communication. In Fig.20, when there are multiple subscribers (node 1 and node 2) for the data at node 0, the *Data Subscription Service* detects the proximity of the two paths and merges these two paths by placing a copy of the data at node 5 and lets node 5 send data to the two subscribers during each requesting interval.

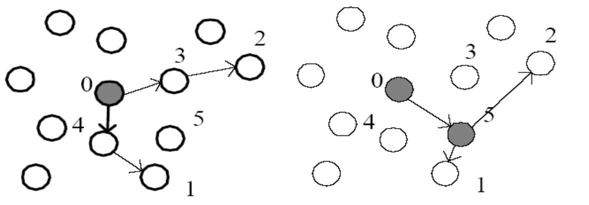


Figure 20: Data Subscription in DSWare
(from [136])

6) Scheduling: The *Scheduling* component schedules other components. Two most important scheduling options are energy aware and real-time scheduling.

6.2.2 Impala

Impala [141, 140] has been built as part of the ZebraNet, in which sensing nodes are placed on free ranging wildlife to perform long-term migration studies on a collection of animals in an ecosystem. It is a middleware architecture that enables application modularity, adaptivity, and repairability in wireless sensor networks. Fig.19 shows the system architecture of Impala. The upper layer contains all the application protocols, while the lower layer contains three middleware agents: the *Application Adapter*, the *Application Updater*, and the *Event Filter*. Impala is essentially a runtime system that acts as a lightweight event and device manager for each mobile wireless sensor node in the system. The applications, the *Application Adapter*, and the *Application Updater* are all programmed into a set of event handlers, which are invoked by the Event Filter when the associated events are received. The *Application Adapter* changes the communication protocol according to the executing context. Improvements in performance, power consumption and robustness are achieved by adapting the communication protocol according to the runtime conditions. The *Application Updater* renews software automatically. The *Event Filter* captures and dispatches events to the above system units and initiates chains of processing. Impala has five types of event: *Timer Event* for the timer, *Packet Event* signals that a network packet has arrived. Impala has two types of packets, application-to-application packets and updater-to-updater packets. The intended receiver of the packet handles these events. *Send Done Event* for signalling that a network packet has been sent or its send has failed. (*Data Event* for signalling that the sensing device is ready to read, and *Device Event* for signalling that a device failure is detected. The *Application Adapter* handles these events.

Adaptation is implemented through Impala's event-based programming model, and occurs in response to a range of events. Some events, like timer events, signal that time has passed since the last status check; the adapter may then choose to query application or system states in order to determine if any adaptation should be performed. Other events, like device events, have sources external to Impala and signal an external event for Impala to respond to, such as the failure of a particular radio transceiver; the adapter should then examine the impact of the failure and determine whether to dispatch another application to work around it.

6.2.3 EnviroTrack

EnviroTrack [1] is the first programming support for sensor networks that explicitly supports tracking mobile objects. Its abstractions and underlying mechanisms are well-suited to monitoring targets that move in the physical world. Dynamic group administration is performed, and a leader is maintained by periodic message exchanges among the group. EnviroTrack is a middleware layer that exports a new address space in the sensor network (see Fig.21). In this space, physical events in the external environment are the addressable entities. This type of addressing is convenient for applications that need to monitor environmental events. For example, a surveillance application that monitors vehicle movement behind enemy lines may assign unique labels to individual vehicles. Their state can then be addressed by reference to these labels. Moreover, computing or actuation objects can be attached to individual addresses.

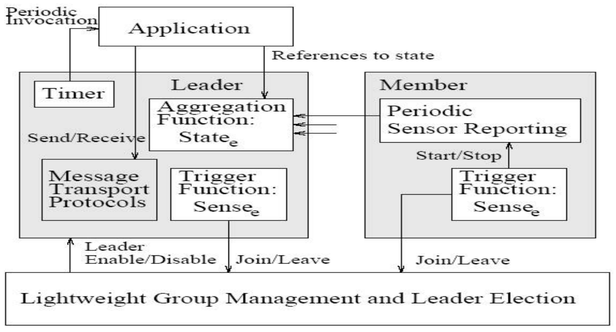

Periodic Invocation
Application
References to state
Timer
Leader
Aggregation Function: State$_e$
Member
Periodic Sensor Reporting
Send/Receive
Start/Stop
Message Transport Protocols
Trigger Function: Sense$_e$
Trigger Function: Sense$_e$
Leader Enable/Disable
Join/Leave
Join/Leave
Lightweight Group Management and Leader Election

Figure 21: Middleware Architecture in EnviroTrack

(from [1])

The attached computation or actuation is then performed in the physical neighbourhood of the named entity. Hence, for example, a microphone could be turned on at some network address (e.g., one that names a vehicle in the external environment) to listen in on the corresponding environmental object. As the named vehicle moves, the middleware will turn on the appropriate nearby node microphones so that a non-interrupted audio stream is delivered to the receiver, despite the mobile nature of the source. Communication can also occur between two mobile endpoints. For example, a walking soldier with a PDA may track the position of a suspect vehicle detected elsewhere in the network. In essence, EnviroTrack supports:

- Exporting a novel logical address space in which external environmental objects are the labelled entities.

- Allowing arbitrary data, computation, or actuation to be attached to such logical network addresses. These data, computation, and actuation are encapsulated in an abstraction as tracking objects.

The EnviroTrack middleware library implements a set of protocols that offload from an application developer the details of inter object communication and object mobility, as well as the maintenance of tracking objects and their state. It abstracts away the fact that computation associated with the object may be distributed and performed by all sensor nodes in the vicinity of the tracked physical entity. As the tracked entity moves, the identity and location of the sensor nodes in its neighbourhood change, but the tracking object representing it remains the same. The programmer thus interacts with a changing group of sensor nodes through a simple, uniquely addressable, object interface. Fig.22 shows the programming model.

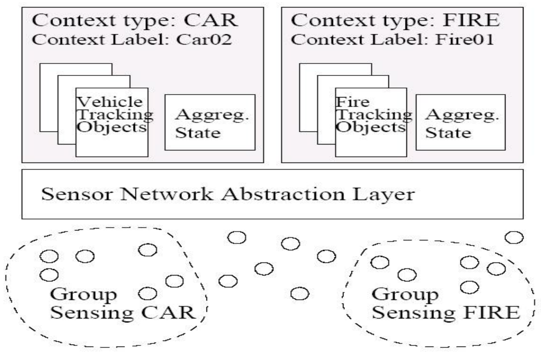


Figure 22: EnviroTrack Programming Model

(from [1])

Group management services, shown at the bottom of Fig.21 maintain coherence of context labels, where a group of sensors identifying the same entity in the environment produce a single context label. Fig.23 shows the outline of an object tracking.

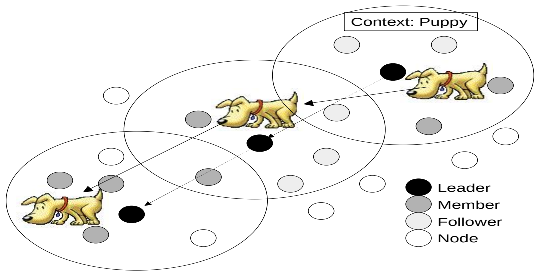


Figure 23: Tracking Objects in EnviroTrack

Contexts are created when a node first senses a condition. The node immediately starts a leader election process in which it randomly chooses a small timeout value. A node which times out first sends a message informing its neighbours that it is leader. Upon receipt of this message, other nodes sensing the same condition become members. A nodes communication radius has to be larger than twice its sensing radius such that all nodes sensing the same target are within each others communication range. An elected group leader sends periodic heartbeats, which are received by all group members. Leader heartbeats have three purposes. First, they inform current members that the leader is alive. Should the leader die, a new leader election is started after a timeout. Second, they carry application state that must persist across leader handoffs. This state is recorded by all member nodes. This mechanism allows new leaders to continue computations of failed leaders from the last state received. An application can explicitly create a persistent state primitive and read it. Finally, heartbeats are overheard past the groups perimeter thus informing neighbouring nodes of the existence of a context label. Nodes that cannot sense the target themselves but know of its existence from nearby leader heartbeats are called group followers. If these nodes subsequently sense the condition, they join the present group instead of forming a new context label. The mechanism ensures that multiple spurious context labels do not emerge around the same target. When the leader gets out of sensory range from the target, it sends a leader handoff message which initiates a new leader election. The resulting behaviour is that a group with a unique leader is created around each target. Membership changes and leader (and state) handoffs occur automatically as the target moves.

6.2.4 CORTEX

CORTEX [27] provides a programming model that supports the development of applications constructed from mobile sentient objects taking into account the provision of incremental real-time and reliability guarantees as well as the design of an open, scalable system architecture that reflects the heterogeneous structure and performance of the networks. Publish/Subscribe based middleware for ad hoc networks is used by the sentient object model to disseminate context and other data.

A sentient object is an encapsulated entity, with its interfaces being sensors and actuators. Sensors are defined as entities that produce events in reaction to a real world stimulus, whilst actuators are defined as entities which consume events and react by attempting to change the state of the real world. The sentient object contains a sensory capture component, which performs sensor fusion based on Bayesian networks. It provides an efficient approach to intelligent reasoning based on a hierarchy of contexts. Fig.24 shows a sentient object and its internals.

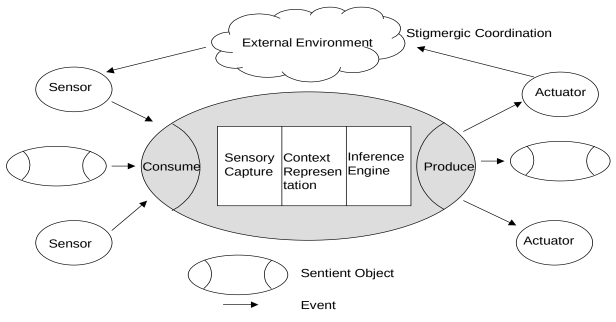


Figure 24: Sentient Object Model (from [27])

Internally, a sentient object consists of three major functional components:

- The sensory capture component is responsible fusing the outputs of multiple sensors, and uses probabilistic models, including Bayesian networks, to deal with inherent sensor uncertainties.
- The context representation component maintains a hierarchy of potential contexts in which an object can exist, and the current active context.
- The inference engine component is a production rule based inference engine and supporting knowledge base, giving objects the ability to intelligently control actuation based on their context.

The CORTEX middleware supports diverse application domains such as cooperating sentient vehicles [200] and smart living environments. A particular configuration of the middleware for the mobile ad-hoc networks (MANETs) is shown in Fig.25. This configuration was targeted towards the cooperating sentient vehicles application, where context-aware, autonomous vehicles travel from a given source to destinations and cooperate with other vehicles to avoid collisions, obey road side traffic lights and give way to pedestrians.

In more detail, sentient objects consume events from a variety of different sources including sensors and event channels, fuse them to derive higher-level contexts, reason about them using expert system logic (based on a CLIPS inference engine), and produce output events whereby they actuate the environment or interact with other objects. This example is exploring the area of autonomous vehicle navigation in which vehicles, represented as mobile sentient objects, have the objective of travelling along a given path, defined by a set of GPS way points. Every vehicle acts as a sentient object that cooperates with other vehicles (sentient objects) by inter vehicle communication mechanisms and with other infrastructure objects (e.g. traffic lights or speed signals).

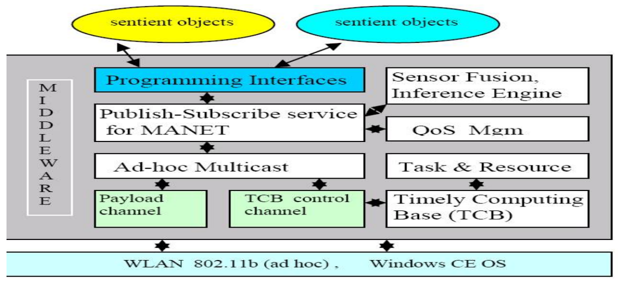


Figure 25: CORTEX Middleware Architecture (from [27])

6.2.5 Mires

Mires [208] is middleware based on the publish/subscribe paradigm for messaging in WSN. The operation assumes a WSN, of hierarchical structure, communicating with the nodes of a wired network on which applications reside via a single sink node that is a gateway between the WSN and the applications. Initially, the nodes in the sensor network create advertisements for their available topics (e.g. temperature and humidity) collected from local sensors. Next, the advertisement messages are routed to the sink node using a multi-hop routing algorithm. A user application (e.g. via a graphical user interface) connected to the sink node is able to select (i.e. subscribe to) the advertised topics of interest that are to be monitored. Messages corresponding to these subscriptions are then broadcast down to the sensor network nodes. After receiving the subscriptions, sensor nodes publish their collected data via the sink node to the network-based applications.

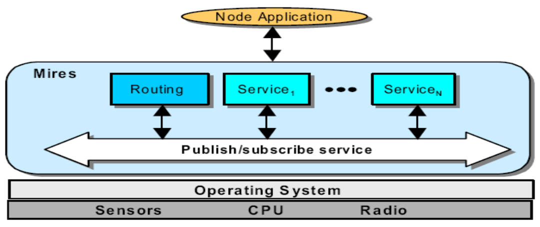


Figure 26: Mires Overview (from [208])

Additional services (e.g. a data aggregation service) may easily be integrated with the publish/subscribe service if they implement the appropriate interfaces. Fig.26 shows an overview of the Mires architecture. From the bottom to the top, the first block corresponds to the sensor nodes hardware components. It generally includes a micro controller unit, one or more sensors and a radio transceiver. These components are directly interfaced and controlled by the operating system (OS). The low level services provided by the OS can be accessed through standard interfaces. Mires has a simple architecture, however it implements high-level publish/subscribe by providing services and routing while hiding the low level complexity of the sensor network.

6.2.6 *Scope*

Scope [210] is a generic abstraction for the definition of groups of nodes. Multi purpose WSNs will be heterogeneous in most cases: for each application only a specific type of sensor node or some parts of the monitored environment may be relevant. Scope proposes a middleware framework based on publish/subscribe messaging within the group of nodes, which can be in geographically close proximity, sensor type, and so forth. The multi purpose networks can be tackled from two directions. On the one hand, there are low level implementations that focus on programming individual sensors and their sensing/acting and communication facilities directly such as TinyOS. On the other hand, there is work on declarative query processors offering high-level interfaces [146], which do not allow explicit control on the level of individual nodes. The difficulty lies in finding a good tradeoff between the universality of such high-level interfaces and the degree to which application specific details can be passed and utilized for the optimisation of routing and resource scheduling. Scope aims to bridge the gap between high and low level interfaces and enable the partitioning of WSN functionality. Scope as a middleware building block facilitates the construction of tailored services in multi purpose WSNs. Fig.27 shows an abstract model of Scope applications in a WSN.

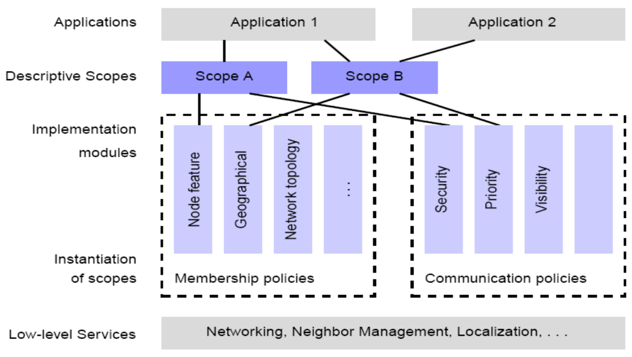


Figure 27: Abstract Model of Scope Applications (from [210])

6.3 QoS Oriented Approach

In this section, middleware that aims to provide quality of sensor data and reliability of the collected data, depending on application requirements, is described.

6.3.1 *MiLAN*

MiLAN (Middleware Linking Applications and Networks) [89, 150, 159] is middleware for sensor networks for applications that require QoS on the sensor information, such as monitoring patients for medical treatment. Here, QoS of sensor information indicates its reliability, that is, a correct value from a sensor of reliability 1.0 without fail can be requested by an application. MiLAN allows sensor network applications to specify their quality needs, and adjusts the network characteristics to increase application lifetime while still meeting those quality needs. Fig.28 shows a high-level diagram of a system that employs MiLAN.

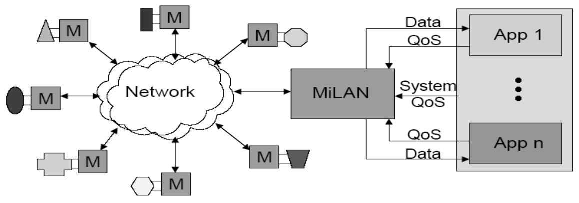


Figure 28: MiLAN System Overview (from [89])

Unlike traditional middleware that sits between the application and the operating system, MiLAN has an architecture that extends into the network protocol stack, as shown in Fig.29

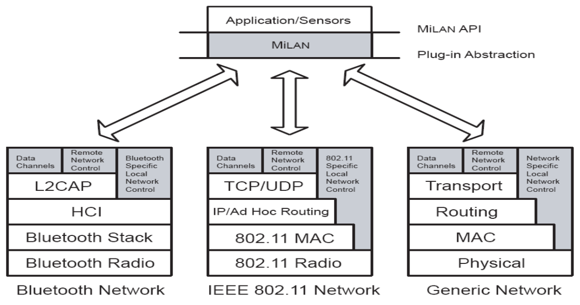


Figure 29: MiLAN Protocol Stack (from [89])

As MiLAN is intended to sit on top of multiple physical networks, an abstraction layer is provided that allows network specific plug-ins to convert MiLAN commands to protocol-specific commands that are passed through the usual network protocol stack. Therefore, MiLAN can continuously adapt to the specific features of whichever network is being used for communication (e.g. determining scatternet formations in Bluetooth networks etc.) in order to best meet the applications' needs over time.

In order to determine how to best serve the application, MiLAN must know (1) the variables of interest to the application, (2) the required QoS for each variable, and (3) the level of QoS that data from each sensor or set of sensors can provide for each variable. Note that all of these may change based on the application's current state. During initialization of the application, this information is conveyed from the application to MiLAN via *State based Variable Requirements* and *QoS* graphs. These sets of sensors define the application feasible set Fa, where each element in Fa is a set of sensors that provides QoS greater than or equal to the application-specified minimum acceptable QoS for each specified variable. For example, in a personal health monitor, for a patient in medium stress with a high heart rate, normal respiratory rate, and low blood pressure, the application feasible sets in Fa that MiLAN should choose to meet the specified application QoS are shown in Table 1. MiLAN must choose which element of

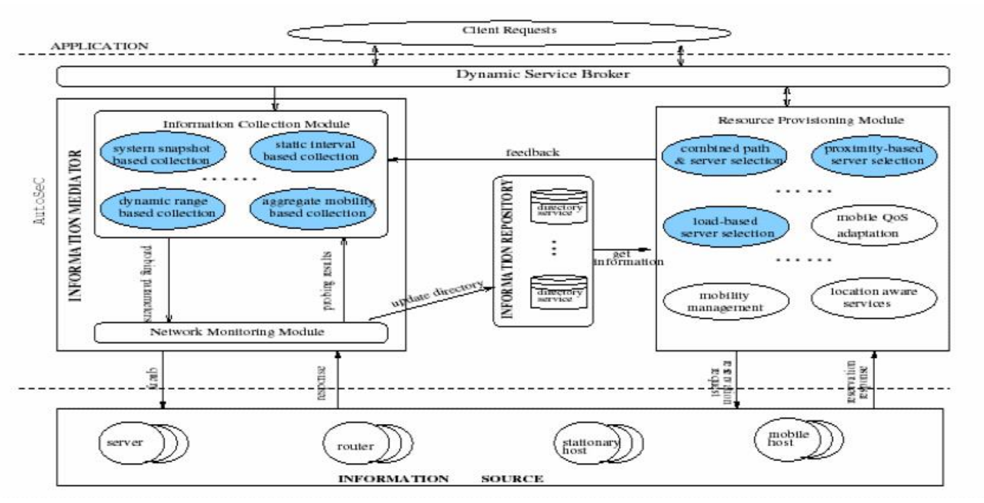


Figure 30: Dynamic Service Broker Framework in AutoSec
(from [80])

Fa should be provided to the application. This decision depends on network level information.

MiLAN dynamically creates the combination of sensors to adapt to the application's QoS request and improve physical resources (e.g. transmission distance and bandwidth). Thus, it has the function to adjust the trade-off between QoS and the cost of the sensor network (for instance, energy consumption).

6.3.2 QUASAR

Unlike conventional distributed database systems, a sensor data architecture must handle extremely high data generation rates from a large number of small autonomous components. And unlike the emerging paradigm of data streams, it is infeasible to think that all this data can be streamed into the query processing site, due to server bandwidth and energy constraints of battery-operated wireless sensors. Then, Quality-Aware Sensing Architecture (QUASAR) [131, 82] proposes the sensor data architecture, that must become quality-aware, regulating the quality of data at all levels of the distributed system, and supporting user applications' quality requirements in the most efficient manner possible. For example, QUASAR aims to process the quality-aware queries (Q_aQ_s) such as "Retrieve the sensor IDs and temperatures (within $\pm 5^o$ C) of all sensors whose temperature is above 30^o in a certain area R".

6.3.3 AutoSec

Automatic Service Composition, AutoSeC [80], is a dynamic service broker framework for effective utilization of resources within a distributed environment. Distributed applications have QoS requirements that can be translated into the underlying system level resources and, in this respect, AutoSeC does resource management within a sensor network by granting access control to applications in order to meet QoS requirements on a per sensor basis. Crucially, meeting these requirements entails the AutoSeC middleware dynamically choosing a combination of information collection and resource provisioning policies from a given set based on user needs and system state. Since the choice of policies is done by AutoSeC, the application developer or system administrator is relieved from the tedious task of having to make a choice from the set of policies that are available.

Fig.30 presents an overview of the AutoSec framework. AutoSeCs *directory service* is centralized and stores system state information concerning three categories of parameters: (1) network parameters e.g. bandwidth, end-to-end delay, (2) server parameters e.g. buffer capacity, CPU utilization, disk space, and (3) client parameters e.g. client connectivity, capacity etc. Each of these parameters can be represented by either one value or a range of values with an upper and lower value. The *information collection module* determines whether to update the information repository with the current system image. This is influenced by the rate at which samples are collected. The higher the sampling interval, the higher the accuracy of the directory services information, and the higher the overhead introduced into the system. Hence the information collection module has to maintain a balance between the information accuracy and the overhead introduced by the directory service maintenance. The *resource provisioning module* uses information received from the directory service regarding the current system state to perform resource allocation. It uses intelligent mechanisms to choose appropriate resources e.g. servers, routing paths to service QoS based requests. It takes into account the current network and server utilization parameters provided by the directory service to allocate resources in a way that optimises overall system performance. Once new clients are assigned a path and/or server, they set up a connection to the assigned server on the assigned path. Routers and servers along the assigned path check their residual capacity and perform either of two actions; admit the connection and reserve resources or reject the request. Once the connection of a client to a server terminates, the client makes a termination request and the resources along the formerly assigned path are reclaimed by the resource provisioning module. The *network monitoring modules* are distributed, with each module monitoring parts of the entire network. Each module probes routers and servers to collect system state information. These probes consolidate the sample values collected by the routers and servers before they forward them to the monitor. The network monitoring modules finally transmit the information to the information collection module which performs updates on the direc-

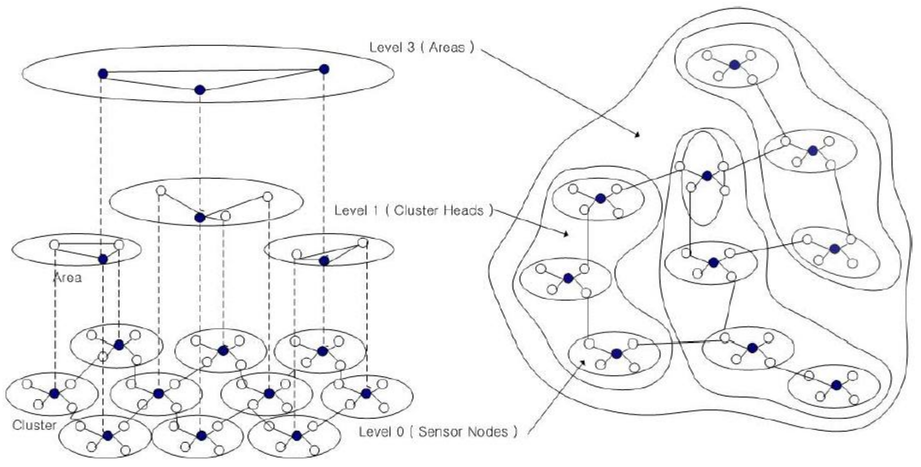


Figure 31: 3-level hierarchical cluster-based network
(from [107])

tory service. AutoSeCs *dynamic service broker* performs all the decision making functions regarding what combinations of resource provisioning and information collection policies that satisfy user requests and match current system conditions. It also decides when to switch between these policies at run-time.

6.4 Internet Oriented Approach

To date, work in the sensor network community has focused on collecting and aggregating data from specific networks with an associated base station. The problem of delivering this data to external systems is typically left open. Unlike the traditional closed network setting for specific applications, middleware for multi purpose sensor networks should offer a more open platform where various applications can access the sensor information. Use of the Internet is an intuitive approach to achieve open sensor network middleware. In this section, middleware that aims to integrate sensor networks and the Internet is described.

6.4.1 Web based query management

In [107], a web based query and management programming model using a gateway is proposed. Basically all sensor readings go to a gateway, a powerful node that has connection to the outside world. Most of the data processing, aggregation, as well as query transformation and the web front end to the database module reside in the gateway. The system can be configured using the web front end via the gateway. A WSN is assumed with a 3-level regional hierarchy (shown in Fig.31), and the entire network employs 3-level hierarchy: areas, clusters, and sensor nodes. To facilitate scalable operations in sensor networks, sensor nodes should be aggregated to form clusters based on their power levels and proximity. A cluster head is elected from the sensor nodes belonging to the same cluster, and is responsible for acquiring data from the sensor nodes in its cluster. An area consists of cluster heads. That is, the entire sensor network has some areas, an area has some clusters, and a cluster head has some sensor nodes. Such a 3-level hierarchical structure is useful for managing the sensor network regionally.

6.4.2 Data Collection Network (DCN)

The Hourglass project [195] proposes an Internet based infrastructure that handles aspects of naming, discovery, schema management, routing, and aggregating data from potentially many geographically diverse sensor networks, called Data Collection Network (DCN). DCN addresses the following problems:

- Intermittent connectivity: How does a DCN manage communications with mobile or poorly connected entities that may exhibit intermittent connectivity with the rest of the infrastructure? How does the DCN ensure that the data flow is not disrupted during disconnection.
- Management of resource: How does a DCN infrastructure become aware of, and broker access to, a wide range of sensor networks and services that may exist in different administrative domains, each with different interfaces and access rights?
- Service composition: How do applications tie together a suite of services for processing data flowing from sensor networks? What is the model for mapping application data requirements onto individual services, and how are those services instantiated and managed? How do applications integrate application-specific processing into an existing DCN?
- Supporting heterogeneity: How does a DCN infrastructure provide services in the presence of resource-constrained devices? What minimal functionality is needed by all participants? How should a DCN accommodate devices of varying capabilities?

An example of an Hourglass system structure with one realized circuit is shown in Fig.32. A circuit can be described by a set of circuit links between service providers (SPs) and the schema of data travelling over these links. Data flow in Hourglass is based on a circuit, which is a data path through the system that ensures that an application receives the data in which it is interested. A circuit includes intermediate services that perform operations on the data. Services are organized into distinct service providers that capture a single administrative domain. Each service provider includes a circuit manager, which is responsible for the set-up and management of circuits, and a registry, which aids service discovery. Hourglass services have well specified interfaces that are used to communicate

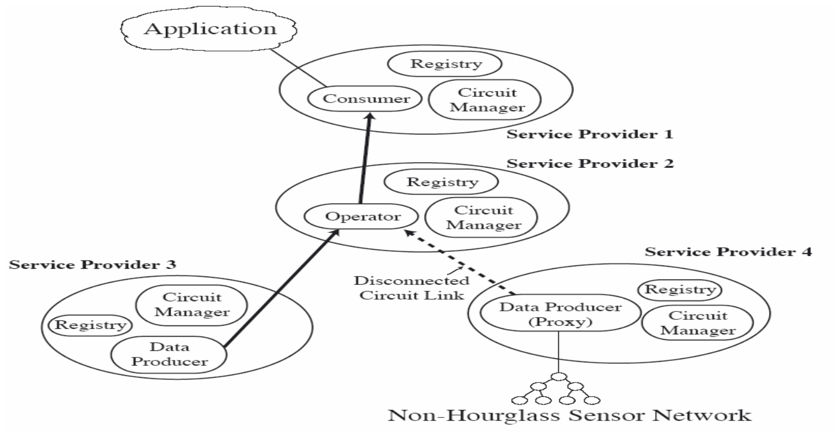


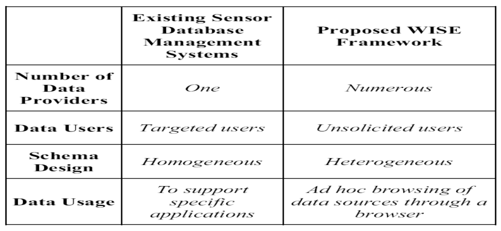
Figure 32: Example of Hourglass System
(from [195])

with other services, the circuit manager, and the registry for circuit establishment, data routing, circuit disconnection, and service discovery. An existing entity can join an Hourglass system by implementing the core functionality required by these interfaces.

The structure of a circuit is specified in the Hourglass Circuit Descriptor Language (HCDL). The HCDL is an XML defined language that applications use to define desired circuits to be established by Hourglass. An HCDL circuit description can exist in three forms: the nodes of an unrealized circuit are not tied to actual service instances in the system, a partially realized circuit contains some of the bindings, whereas a fully realized circuit includes all of the service endpoints for the services used by the circuit. An unrealized circuit includes constraints on the services that may be instantiated at a certain node in the circuit. It is the task of the circuit manager to resolve an unrealized circuit into a realized one, subject to the constraints imposed by the application. Data flowing along a circuit may be filtered, aggregated, compressed, or temporarily buffered by a set of services that exist in the Hourglass infrastructure. These primitives allow applications to construct powerful assemblies of sensor networks and infrastructure services. Circuits are constructed with the aid of a registry that maintains information on resource availability and a circuit manager that assigns services and links to physical nodes in the network. In addition, Hourglass supports a set of in-network services such as filtering, aggregation, compression, and buffering stream data between source and destination.

6.4.3 WISE

WISE (Web-based Intelligent Sensor Explorer) [169, 104] is a framework to provide the services: (1) a publishing facility to allow sharing of real-time sensor data on the Web; (2) a search mechanism to enable unsolicited users to discover relevant sensors; and (3) an intelligent browser to provide the user interface to view, analyze, and query the sensor data streams in intelligible ways. The future Internet, where innumerable sensing systems will be deployed by numerous publishers, exposes different data which can be shared freely with a wide range of unsolicited users, much like the way web pages are publicly shared today. The Web Services approach is taken in WISE, because this technology has become mature, with well-defined specifications. Unlike existing work, the WISE environment is not a sensor database management system. Rather, it is an extension to the Web to enable exchange of sensor data and sharing of sensor applications. A summary of the differences between the current database approach and the proposed environment is shown in Fig.33.



	Existing Sensor Database Management Systems	**Proposed WISE Framework**
Number of Data Providers	*One*	*Numerous*
Data Users	*Targeted users*	*Unsolicited users*
Schema Design	*Homogeneous*	*Heterogeneous*
Data Usage	*To support specific applications*	*Ad hoc browsing of data sources through a browser*

Figure 33: Sensor DB Management Systems and WISE
(from [169])

Fig.34 shows a high level WISE environment. To publish the data, the geographers provide relevant metadata for their sensors; and a service description as well as a WSDL file is automatically created on a Sensor Data Server (SDS) to hold the metadata. The service description is then uploaded to a UDDI registry. Sometime later, a biologist looking for related sensors enters search criteria into a local Sensor Data Browser (SDB). This software connects to the UDDI registry and looks for relevant service descriptions. As the browser displays a list of relevant service descriptions on the screen, the biologist identifies the desired sensor and clicks on it. In response, the browser interacts with the SDS using the operations defined in a Sensor Stream Control Protocol (SSCP). As the browser gets the real-time stream using a Sensor Stream Transport Protocol (SSTP), it feeds the data to a Visualizer for display. The various components are described in the following subsections.

6.4.4 IrisNet

IrisNet [74, 162, 161] is middleware for globally distributed sensing systems. Sensors themselves are resourceful and capable of providing dense sensor data such as video streams. The project follows a traditional two tier architecture, where sensor nodes are the leaves and servers are the core. IrisNet takes the design approach of building and querying wide area sensor databases. The system behaviour is programmable in the sense that senselets are

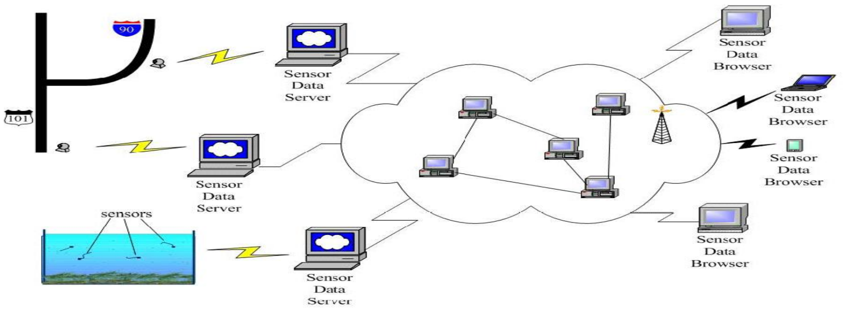


Figure 34: High-level environment in Wise
(from [169])

placed on sensor nodes, which can filter, store, process and analyze the collected data locally. IrisNet offers a programming interface where the service provider can easily add the new sensor service. Data are only transmitted when queried, and both query and response data are routed intelligently to the requesting user. This work deals with widely-deployed, resource-abundant, powered sensors like webcams and organizes them into a distributed database. The entire data-base is treated logically as a single XML document which is partitioned among multiple sites. The techniques proposed are effective for fragmenting and partitioning a database, routing and processing queries, and caching remote data locally. This work, however, does not consider streaming data and only works with a predefined database, i.e. all the sensors are assumed to be owned by the same organization.

6.5 Agent Based Approach

Frameworks for software agent development within the agent community are well-developed and common; a representative example is the Java Agent Framework (JAF). The agent community has begun to consider the use of agents for sensor network applications as well. In this section agent based middleware is described. Gator Tech Smart House [93] is an good example that exploits an agent technology and service oriented architecture at lower layers. The use of service oriented programmable spaces is broadening the traditional programmer model. They utilize OSGi platform building a surrogate service bundle. Smart house further specifies a programming model for smart environment embodied in a middleware. In which the implementation of a smart environment is divided into four layers, physical (sensor), sensor platform (sensor surrogate), context and knowledge (system level services) and application layers.

6.5.1 Sensorware

SensorWare [33, 34] is a general middleware framework based on agent technology, where the mobile agent concept is exploited. Mobile control scripts in *Tcl* model network participants functionalities and behaviours and routing mechanisms to destination areas. Agents migrate to destination areas performing data aggregation reliably. The script can be very complex and diffusion gets slower when it reaches destination areas. Wireless ad hoc sensor networks have emerged as one of the key growth areas for wireless networking and computing technologies. So far these networks/systems have been designed with static and custom architectures for specific tasks, thus providing inflexible operation and interaction capabilities. Sensorware's vision is to create sensor networks that are open to multiple transient users with dynamic needs. The replication and migration of such scripts in several sensor nodes allows the dynamic deployment of distributed algorithms into the network. SensorWare, defines, creates, dynamically deploys, and supports such scripts. SensorWare is designed for iPAQ devices with megabytes of RAM. The verbose program representation and on-node Tcl interpreter can be acceptable overheads, however they are not yet on a mote.

6.5.2 RUNES

The project RUNES (Reconfigurable Ubiquitous Networked Embedded Systems) [185] aims to enable the creation of large-scale, widely distributed, heterogeneous networked embedded systems that interoperate and adapt to their environments. A general goal is to provide an adaptive middleware platform and application development tools that allow programmers the flexibly to interact with the environment where necessary, whilst affording a level of abstraction that facilitates ease of application construction and use. The RUNES approach to middleware provision is to build the middleware in terms of a well-defined, language-independent component model which is supported by a minimal runtime API. The required heterogeneous realization of the component model Fig. 35, which is a local model, for various types of devices is achieved by providing different implementations of the runtime API, and by implementing components themselves in various ways. For example, on a PDA running a standard OS, components might be sets of Java classes or as Linux shared objects; whereas on a sensor motes micro controller, components might be implemented simply as segments of machine code.

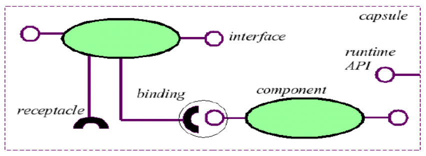


Figure 35: Elements of RUNES Component Model
(from [185])

This provides support for dynamic adaptation to changing conditions; a fundamental requirement in the context-aware scenarios typical of networked embedded systems. Moreover, the RUNES middleware reaches down into layers that typically belong to the network and the operating system, therefore providing a unified approach to configuration, deployment and reconfiguration at multiple levels of abstraction. Distribution is assumed to be built on top of this foundational layer. The component model itself is complemented by two further architectural elements: component frameworks and reflective meta models. A meta model can be used to manage system reconfiguration in a distributed environment. A high level view of the reconfiguration meta model is shown in Fig. 36. The meta model is based on principles of code mobility and can be used dynamically to transfer code, state and data between RUNES nodes.

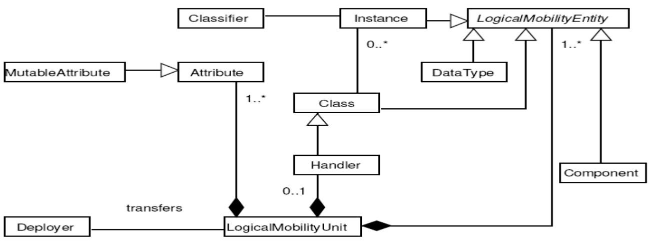


Figure 36: Reconfiguration Meta-Model (from [185])

6.6 Centralized Approach

In this section, as a contrast to the recent decentralized systems, traditional centralized sensing systems are introduced. These systems aim to support specific applications such as location systems.

6.6.1 Active BAT

Sentient computing is a type of ubiquitous computing which uses sensors to perceive its environment and react accordingly. A use of the sensors is to construct a world model, which allows location-aware or context-aware applications to be constructed. One research prototype of a sentient computing system was the work at AT&T Laboratories in the 1990s and the research continues at our Computer Laboratory by means of the Active BAT system [84]. This is a low power, wireless, indoor location system accurate up to 3cm. It uses an ultrasound time-of-light trilateration technique to provide accurate physical positioning.

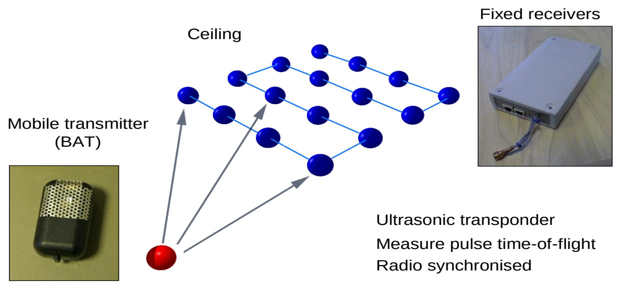


Figure 37: Active BAT

Users and objects carry Active BAT tags. In response to a request that the controller sends via short-range radio, a BAT emits an ultrasonic pulse to a grid of ceiling mounted receivers. At the same time that the controller sends the radio frequency request packet, it also sends a synchronized reset signal to the ceiling sensors using a wired serial network. Each ceiling sensor measures the time interval from reset to ultrasonic pulse arrival and computes its distance from the BAT. The local controller then forwards the distance measurements to a central controller, which performs the trilateration computation. Statistical pruning eliminates erroneous sensor measurements caused by a ceiling sensor hearing a reflected ultrasound pulse instead of one that travelled along the direct path from the BAT to the sensor. The SPIRIT (SPatially Indexed Resource Identification and Tracking) [84] provides a platform for maintaining spatial context based on raw location information derived from the Active BAT location system. It uses CORBA to access information and spatial indexing to deliver high-level events such as 'Alice has entered the kitchen' to listening context aware applications. SPIRIT models the physical world in a bottom up manner, translating absolute location events for objects into relative location events, associating a set of spaces with a given object and calculating containment and overlap relationships among such spaces, by means of a scalable spatial indexing algorithm. However, this bottom-up approach is not as powerful in expressing contextual situations.

6.6.2 SCAFOS

SCAFOS [121] is a framework summarizing a vast rate of low level sensor data into forms which can be used by high-level applications. SCAFOS addresses similar problems to macroprogramming. It is based on the Active BAT, thus data comes from the centralized database, where sensor data is received from the Active BAT system. Primitive events, especially those arising from sensors, e.g., that a user is at position (x; y; z), are too low level to be meaningful to applications. Existing models for creating higher-level, more meaningful events, from low level events, are insufficient to capture the user's intuition about abstract system state. Furthermore, there is a strong need for user-centred application development, without undue programming overhead. Applications need to be created dynamically, and remain functional, independently of the distributed nature and heterogeneity of sensor driven systems and even while the user is mobile. Both issues combined necessitate an alternative model for developing applications in a real-time, distributed sensor driven environment such as Sentient Computing. SCAFOS provides powerful tools for inferring abstract knowledge from low level, concrete knowledge, verifying its correctness and estimating its likelihood. Such tools include Hidden Markov Models, a Bayesian Classifier, Temporal First Order Logic, the theorem prover SPASS and the production system CLIPS. Secondly, SCAFOS provides support for simple application development through the XML-based SCALA language. By introducing the new concept of a generalized event, an abstract event, defined as a notification of changes in abstract system state, expressiveness compatible with human intuition is achieved when using SCALA. The applications that are created through SCALA are automatically integrated and operate seamlessly in the various heterogeneous components of the context aware environment even while the user is mobile or when new entities or other applications are added or removed in SCAFOS.

7. FUTURE CHALLENGES

In this section, we describe the future perspectives for research in WSNs.

Hardware: Hardware research seems to be slowing down. Based on prototype hardware from current research, interest is moving to the development of commercial products.

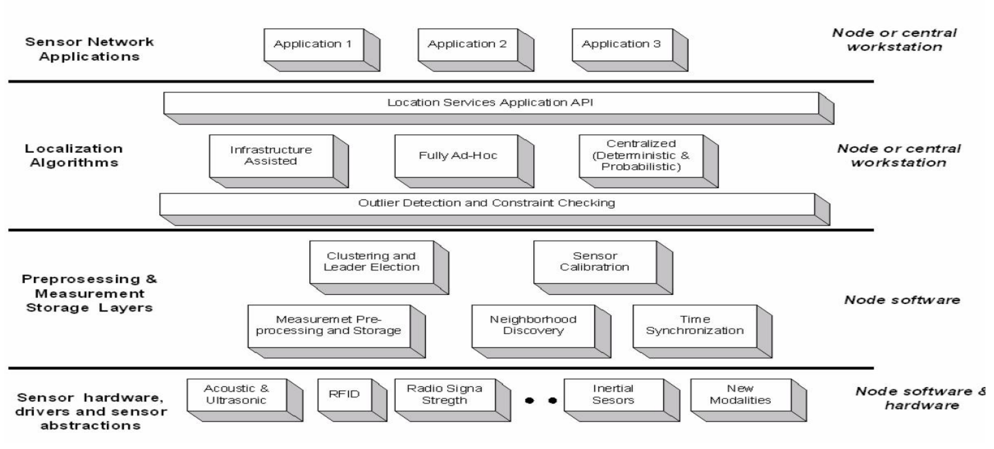

Sensor Network Applications
Application 1
Application 2
Application 3
Node or central workstation
Location Services Application API
Localization Algorithms
Infrastructure Assisted
Fully Ad-Hoc
Centralized (Deterministic & Probabilistic)
Node or central workstation
Outlier Detection and Constraint Checking
Preprocessing & Measurement Storage Layers
Clustering and Leader Election
Sensor Calibration
Measuremet Pre-processing and Storage
Neighborhood Discovery
Time Synchronization
Node software
Sensor hardware, drivers and sensor abstractions
Acoustic & Ultrasonic
RFID
Radio Signa Stregth
Inertial Sesors
New Modalities
Node software & hardware

Figure 38: SOA Model

Research interests are moving towards more applied areas. The world will surely change if Motes in millimeter squares, like smart dust, are achieved. However, the miniaturization of Mote by MEMS has yet to be realized and is by no means guaranteed. Contributions to sensor miniaturisation from Europe are desirable.

7.1 System Architecture

A great deal of research on system architecture is predicated on hardware assumptions (e.g. resource constrained devices such as Mote). Hardware progress will be dramatic, as in the past, so there should be more research targeting the support of less resource-constrained hardware and focussing on building intelligent systems. For example, Batlin et al. [22] propose dynamic task allocation that applies research from intelligent robots and multi agent systems. Capturing the concept of WSNs in a wider view is important in expanding the sensor network research area. Another example is that the synchronization update rules, led by a system of pulse-coupled oscillators, realizes the synchronization of states at all nodes [99]. Also, recent rapid progression of macroprogramming includes the use of amorphous computing. These new movements show understanding of WSN's characteristics, especially, amorphous computing demonstrates system coordination. The current stage of amorphous computing has not yet been integrated with programming, but future progress is likely to be significant.

7.2 Network

There is a lot of research that modifies and enhances existing network technology to adapt to the resource constrained sensor network environments. The recent ongoing macroprogramming research makes us reconsider the necessity of general network routing mechanisms, which may not be required under some macroprogramming. The goal of network communication and control is to provide simple and versatile abstractions for communication, data dissemination, and remote execution. The framework for network coordination can be used to implement algorithms for leader election, group management, and in-network aggregation. Thus, we expect to see not only an abstraction of networks but also abstractions of sensor data, application's purposes, and so forth, that provide task-oriented communication mechanisms.

7.3 Middleware

The majority of current middleware for WSNs is based on the data centric approach, and a fundamental idea naturally came from database management systems (DBMS). The database community has taken the view that declarative programming, through a query language, provides the right level of abstraction for accessing, filtering, and processing relational data. The middlewares that take a database approach such as [146] provides an interface for data collection but they do not provide general purpose distributed computation. For example, it is complex to implement arbitrary aggregation and filtering operators and communication patterns with query languages. Thus, more general interfaces for global network programming are desirable.

The evolution of DBMS as a middleware in wired networks led to CORBA and Web Services, that focus on the interfaces with other systems. This predicts that the evolution of middleware for WSNs will be towards an integrated application service. Research for sensor networks with different architectures, to operate cooperatively, will become important in the future. Moreover, standardization problems that past middleware technology faced will be repeated in WSNs.

A service is an interesting concept to be applied in WSNs. It may be a role on a sensor node, or a function providing location information. Services allow cascading without previous knowledge of each other, and enable the solution of complex tasks, where functional blocks are separated in order to increase flexibility and enhance scalability of sensor network node functions. A key issue is to separate the software from the underlying hardware and to divide the software into functional blocks with a proper size and functionality.

Thus far, much current sensor network research has been applied to environment monitoring systems, where the environments are extreme. However, sensor networks will be part of our daily life in future and we must control such surroundings (e.g. as smart spaces). Middleware research is to control sensor networks in a daily scenario but interaction with the user has not yet been addressed. Estrin and others [69] summarize two design ideas below, which point out the importance of data centric and application specific design. They insist that the design principles should be fundamentally different from existing computer network design because of resource constraints in sensor network environments.

- Data Centric: More consideration of the management of sensor data than only node management.

- Application Specific: Consideration of application environments, physically and socially.

Here, middleware technology based on the *Human Centric* approach to support our daily life is desirable. Such human centric middleware requires even more intelligent information processing systems. Current research on the allocation of a dynamic role to the sensor node is only a beginning of applying an intelligent, decentralized algorithm in this research area.

There have been many efforts to provide programming paradigms or virtual machines, and progress in macroprogramming is significant. However, macroprogramming may be a approach or a technology to support and realize programming paradigms for WSNs but is not essential. Many existing middlewares take different approaches such as Spatial Programming using smart messages [30]. Thus far, these two research communities are not working together. An appropriate integration of the traditional approach of middleware with new technologies will be desirable instead each technology attempting to support a specific application scenario. In other words, middleware's mission is creating new paradigms to combine new technologies for WSNs.

Much research for middleware currently focusses on providing generic programming paradigms to ease application developers' tasks, such as data processing or object monitoring. In future, WSNs will carry high bandwidth, low latency streaming data from a variety of sources, such as cameras and multimedia. This requires sophisticated in-network processing (e.g. image fusion and object tracking). Providing various data fusion techniques in a generic manner will be a task of middleware for WSNs.

To realize a vision of the sensor network as a "Nerve system buried under the building and the city", integrating intelligent distributed computing with middleware is necessary, and middleware research will become more important in the future.

7.4 Service Oriented Architecture

There has been an effort to architect middleware for WSNs using service oriented architecture (SOA) (RUNES [185], P2PComp [70], and one.world [77]). In this section, we describe a middleware architecture that envisages an integrated service oriented architecture. The middleware for WSNs should offer an open platform for users to utilize seamlessly various resources in physically interacting environments, unlike the traditional closed network setting for specific applications.

When designing the middleware for sensor networks, heterogeneity of information over global distributed systems must be considered. The sensed information by the devices is aggregated and combined into higher-level information or knowledge and may be used as context. The publish/subscribe paradigm becomes powerful in such environments. For example, a publisher node in event broker middleware can act as a gateway from a WSN, performing data aggregation and distributing filtered data to other networks based on contents. Event broker nodes that offer data aggregation services can efficiently coordinate data flow. Especially with the distributed event-based middleware over peer-to-peer (P2P) overlay network environments, the construction of event broker grids will extend the seamless messaging capability over scalable heterogeneous network environments. Event Correlation will be a multi-step operation from event sources to the final subscribers, combining information collected by wireless devices into higher-level information. Service Oriented Architecture (SOA) is a well proven concept for distributed computing environments. It decomposes applications, data, and middleware into reusable services that can be flexibly combined in a loosely coupled manner. SOA maintains agents that act as software services performing well-defined operations. This paradigm enables the users to be concerned only with the operational description of the service. All services have a network addressable interface and communication via standard protocols and data formats (i.e., messages). SOA can deal with aspects of heterogeneity, mobility and adaptation, and offers seamless integration of wired and wireless environments.

Generic service elements are context model, trust and privacy, mobile data management, configuration, service discovery, event notification, and the following are the key issues addressed for our design.

- Flexible discovery mechanisms for ad hoc networks, which provide the reliable discovery of newly or sporadically available services.

- Support for adaptive communication modes, which provides an abstract communication model underlying different transport protocols. Notably, event-based communication is suitable for asynchronous communication.

Peer-to-peer networks and grids offer promising paradigms for developing efficient distributed systems and applications. Grids are essentially P2P systems. The grid community recently initiated a development effort to align grid technologies with Web Services: the Open Grid Services Architecture (OGSA) [166] lets developers integrate services and resources across distributed, heterogeneous, dynamic environments and communities. The OGSA model adopts the Web Services Description Language (WSDL) to define the concept of a grid service using principles and technologies from both the grid and Web Services. The architecture defines standard mechanisms for creating, naming, and discovering persistent and transient grid-service instances. The convergence of P2P and Grid computing is a natural outcome of the recent evolution of distributed systems, because many of the challenging standards issues are quite closely related.

The Open Services Gateway Initiative (OSGi) [167] is focused on the application layer and open to almost any protocol, transport or device layers. The three key aspects of the OSGi mission are multiple services, wide area networks, and local networks and devices. Key benefits of the OSGi are that it is platform independent and application independent. In other words, the OSGi specifies an open, independent technology, which can link diverse devices in the local home network. The central component of the OSGi specification effort is the services gateway. The services gateway enables, consolidates, and manages voice, data, Internet, and multimedia communications to and from the home, office and other locations.

7.4.1 *Service Semantics*

Service semantics is an important issue, in addition to the service definition, so that services can be coordinated in the space. The model of the real world is of a collection of objects, where objects maintain state using sensor data, and

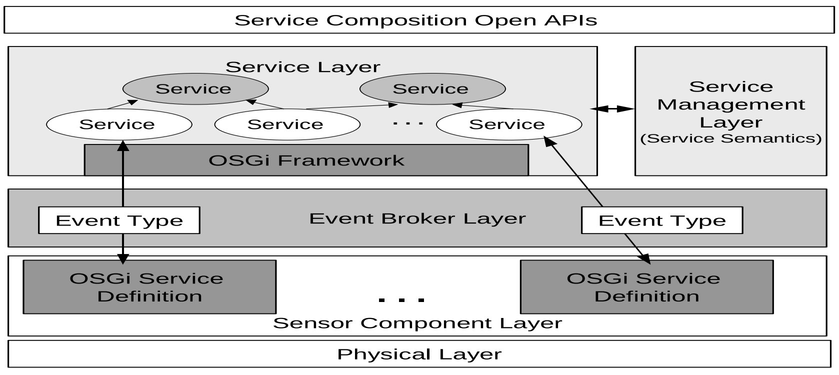


Figure 39: Middleware Architecture with Wireless Sensor Data

applications' queries and subscriptions are a relevant sets of objects. Fig.40 shows an example of object mappings among applications, middleware and sensor components.

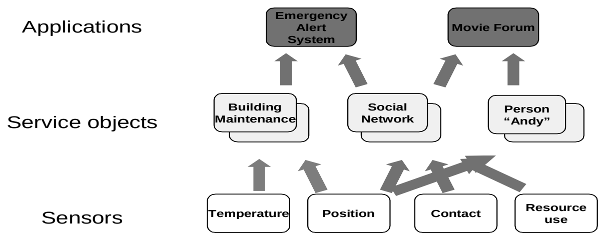


Figure 40: Mapping Real World to Applications

Objects are tightly linked to event types in an event broker. Exploiting semantics will let the pervasive space's functionality and behaviour develop and evolve. Space specific ontologies will enable such exploitation of knowledge and semantics in ubiquitous computing.

7.4.2 Layer Functionality

The brief functionality of each layer is shown below (see Fig.39).

Physical Layer: This layer consists of the various sensors and actuators.

Sensor Component Layer: A sensor component layer can communicate with a wide variety of devices, sensors, actuators, and gateways and represent them to the rest of the middleware in a uniform way. A sensor component effectively converts any sensor or actuator in the physical layer to a software service that can be programmed or composed into other services. Developers can thus define services without having to understand the physical world. Decoupling sensors and actuators from sensor platforms ensures openness and makes it possible to introduce new technology as it becomes available.

Event Broker Layer: This layer is a communication layer between Sensor components and the Service layer. It supports asynchronous communication using the publish/subscribe paradigm. Event filtering, aggregation, and the correlation service is a part of this layer.

Service Layer: This layer contains the Open Services Gateway Initiative (OSGi) framework, which maintains leases of activated services. Basic services represent the physical world through sensor platforms, which store service bundle definitions for any sensor or actuator represented in the OSGi framework. A sensor component registers itself with the service layer by sending its OSGi service definition. Application developers create composite services via the Service Management Layer's functions to search existing services and using other services to compose new OSGi services. Canned services, which may be useful globally, could create a standard library.

A context is represented as an OSGi service composition, where the context can be obtained. The context engine is responsible for detecting, and possibly recovering from, such states.

Service Management Layer: This layer contains an ontology of the various services offered, and the appliances and devices connected to the system. Service advertisement and discovery use service definitions and semantics to register or discover a service. Service definitions are tightly related to the event types used for communication in the Event Broker Layer including composite formats. The reasoning engine determines whether certain composite services are available.

Application Interface: An application interface provides open interfaces for applications to manage services, including the management of contexts.

8. CONCLUSIONS

Middleware has been a key technology in supporting distributed systems by providing common communication mechanisms. For WSN applications, distributed programming abstractions and middleware will be important. However, to support resource-constrained devices, it may not be sufficient to extend existing approaches. Heavyweight middleware, with overloaded abstraction layers, will exceed the capabilities of WSNs. The structure of the WSN may also be highly application dependent, and many services are not independent of the application semantics (e.g. application specific data processing combined with data routing). Data aggregation, event correlation, realtime, asynchronous and intermittent communication are additional challenges.

Design challenges for WSN applications may be categorized as follows. First, wireless networking: given the hardware limitations and physical environment in which the nodes must operate, along with application level requirements. The algorithms and protocols must be designed to provide a robust and energy efficient communication mechanism. Design of physical layer methods such as modulation, routing issues and mobility management must be solved. Second, applications/middleware: at the application/middleware layer, processes aim to create effective new capabilities for efficient extraction, manipulation, transport, and representation of information derived from sensor data. In most applications, WSNs have various functional components: detection and data collection, signal processing, data aggregation, and notification. By integrating sensing, signal processing, and communication functions, a WSN provides a natural platform for hierarchical information processing. It allows information to be integrated at different levels of abstraction, ranging from detailed microscopic examination of specific targets to a detailed view of the aggregate behaviour of targets. Any events in the environment can be processed on three levels: node, local neighbourhood, and global.

The sensor network is an area of research where various technologies cover a wide field. However, it is not easy to coordinate these different research areas. Obtaining hardware is being improved while the development environment and management software are not yet sufficient. Development is difficult without the accumulation of special know-how on hardware and building an operating system in a sensor node. Building hardware, software, and the middleware as reliable base components will be a key issue to communicate precisely among researchers, which will contribute to dramatic progress in sensor network research.

We believe that the issues described in this paper touch on several directions for required research and technologies for WSN applications. Their applications and potential benefits are wide-ranging and could ultimately break the barrier between the physical and digital worlds. There are huge obstacles to overcome, not only in terms of technology, but also in sociology, security and ecology before WSNs become the reality.

Acknowledgement. This research is funded by EPSRC (Engineering and Physical Sciences Research Council) under grant GR/557303. We would like to thank Jon Crowcroft and Andrea Passarella (University of Cambridge) for constructive comments and suggestions.

9. REFERENCES

[1] Abdelzaher, T. et al. EnviroTrack: Towards an environmental computing paradigm for distributed sensor networks. *Proc. ICDCS*, 2004.

[2] Abelson, H. et al. Amorphous Computing. *CACM*, 43(5):7482, 2000.

[3] Abrach, H. et al. MANTIS: system support for multimodAl NeTworks of in-situ sensors. *Proc. WSNA*, 2003.

[4] Al-Karaki, J.N. et al. Routing Techniques in Wireless Sensor Networks: A Survey. *IEEE Wireless Communications*, Vol. 11, No. 6, pp.6-28, 2004.

[5] Akkaya, K. et al. An Energy-Aware QoS Routing Protocol for Wireless Sensor Networks. *Proc. MWN*, 2003.

[6] Akkaya, K. et al. A survey on routing protocols for wireless sensor networks. *Elsevier Ad Hoc Network Journal*, 2004.

[7] Akyildiz, I. et al. Wireless sensor and actor networks: research challenges. *Elsevier Ad Hoc Networks*, 2, pp.351-367, 2004.

[8] Akyildiz, I. et al. Wireless sensor networks: a survey. *Elsevier Computer Networks*, 38, 2002.

[9] Akyildiz, I. et al. On Exploiting Spatial and Temporal Correlation. *Proc. WiOpt*, 2004.

[10] Akyildiz, I. et al. A Survey on Sensor Networks. *IEEE Communications Magazine*, 2002.

[11] Alan, C. et al. Communicating real-time state machines. *IEEE Trans. Softw. Eng.*, vol. 18, no. 9, pp. 805816, 1992.

[12] Andre, C. et al. Representation and analysis of reactive behaviors: A synchronous approach. *Proc. CESA*, 1996.

[13] ARGO. ARGO - Global Ocean Sensor Network. *www.argo.ucsd.edu.*.

[14] Avancha, S. et al. Wireless Sensor Networks. *Kluwer Academic/Springer Verlag Publishers*, 2003.

[15] Awerbuch, B. et al. An On-Demand Secure Routing Protocol Resilent to Byzantine Failures. *Proc. ACM Workshop on Wireless Security (WiSe)*, 2002.

[16] Bakshi, A. et al. Algorithm design and synthesis for wireless sensor networks. *Proc. ICPP*, 2004.

[17] Bakshi, A. et al. The Abstract Task Graph: A Methodology for Architecture-Independent Programming of Networked Sensor Systems. *Proc. EESR*, 2005.

[18] Bakshi, A. et al. System-level Support for Macroprogramming of Networked Sensing Applications. *Proc. Pervasive*, 2005.

[19] Balarin, F. et al. Hardware-software co-design of embedded systems: the POLIS approach. *Kluwer Academic Publishers*, 1997.

[20] Baldus, H. et al. Reliable Set-Up of Medical Body-Sensor Networks. *Proc. EWSN*, 2004.

[21] Basagni, S. et al. Secure Pebblenets. *Proc. MobiHoc*, 2001.

[22] Batalin, M.A. et al. Call and response: experiments in sampling the environment. *Proc. SenSys*, 2004.

[23] Beckwith, R. et al. Pervasive Computing and Proactive Agriculture. *Proc. PERVASIVE*, 2004.

[24] Bergamo, P. et al. Collaborative Sensor Networking Towards Real-Time Acoustical Beamforming in Free-Space and Limited Reverberance. *IEEE Transactions on Mobile Computing*, Vol.3, No.3, pp. 211-224, 2004.

[25] Beutel, J. et al. Prototyping Wireless Sensor Network Applications with BTnodes. *Proc. EWSN*, 2004.

[26] Bhargava, S. et al. Security Enhancements in AODV Protocol for Wireless Ad Hoc Networks. *Proc. Vehicular Technology Conference* , 2001.

[27] Biegel, G. and Cahill, V. A Framework for Developing Mobile, Context-aware Applications. *Proc. PerCom*, 2004.

[28] Blumenthal, J. et al. Wireless Sensor Networks - New Challenges in Software Engineering. *Proc. ETFA*, 2003.

[29] Blumenthal, J. et al. SeNeTs - Test and Validation Environment for Applications in Large-Scale Wireless Sensor Networks. *Proc. INDIN*, 2004.

[30] Borcea, C. et al. Spatial programming using smart messages: Design and implementation. *Proc. ICDCS*, 2004.

[31] Bonfils, B. et al. Adaptive and decentralized operator placement for in-network query processing. *Proc. IPSN*, 2003.

[32] Bonnet, P. et al. Towards Sensor Database Systems. *Proc. MDM*, 2001.

[33] Boulis, A. et al. Design and implementation of a framework for efficient and programmable sensor networks. *Proc. MobiSys*, 2003.

[34] Boulis, A. et al. Aggregation in sensor networks: An energy - accuracy tradeoff. *Proc. IEEE workshop on Sensor Network Protocols and Applications*, 2003.

[35] Boussinot, F. et al. The ESTEREL Language. *Proc.*

IEEE, vol. 79, no. 9, pp. 12931304, 1991.

[36] Brennan, S.M. et al. Radiation Detection with Distributed Sensor Networks. *IEEE Computer*, Vol. 37, No. 8, pp. 57-59, 2004.

[37] Braginsky, D. et al. Rumor Routing Algorithm for Sensor Networks. *Proc. WSNA*, 2002.

[38] Britton, M. et al. The SECOAS project: Development of a self organizing,wireless sensor network for environmental monitoring. *Proc. workshop SANPA*, 2004.

[39] Bulusu, N. et al. Self-Configuration Localization Systems. *PhD Dissertation, UCLA*, 2002.

[40] Burrell, J. et al. Vineyard Computing: Sensor Networks in Agricultural Production. *IEEE Pervasive Computing*, Vol.3, No.1, pp. 38-45, 2004.

[41] Butler, Z. et al. Event-Based Motion Control for Mobile-Sensor Networks. *IEEE Pervasive Computing*, Vol.2, No.4, pp. 34-42, 2003.

[42] Butler, Z. et al. Networked Cows: Virtual Fences for Controlling Cows. *WAMES*, 2004.

[43] Carter, S. et al. Secure Position Aided Ad hoc Routing Protocol. *Proc. CCN02*, 2002.

[44] Cerpa, A. et al. ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies. *IEEE Transactions on Mobile Computing*, Vol.3, No.3, pp. 272-285, 2004.

[45] Chang, J.H. et al. Maximum Lifetime Routing in Wireless Sensor Networks. *Proc. ATIRP*, 2000.

[46] Chaudhary, S. et al. Architecture of Sensor based Agricultural Information System for Effective Planning of Farm Activities. *Proc. SCC*, 2004.

[47] Chen,S . et al. Database-Centric Programming for Wide-Area Sensor Systems. *Proc. DCOOS*, 2005.

[48] Chen, W. et al. Dynamic Clustering for Acoustic Target Tracking in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, Vol.3, No.3, pp. 258-271, 2004.

[49] Cheng, X. et al. TPS: A Time-Based Positioning Scheme for Outdoor Wireless Sensor Networks. *Proc. Infocom*, 2004.

[50] Cheong, E. et al. Tinygals: a programming model for event driven embedded systems. *Proc. SAC*, 2003.

[51] Chiasserini, C.F. et al. Modeling the Performance of Wireless Sensor Networks. *Proc. Infocom*, 2004.

[52] Chong, C. et al. Sensor Networks: Evolution, Opportunities and Challenges. *Proc. IEEE*, 91(8), 2003.

[53] Chu, M. et al. calable Information-Driven Sensor Querying and Routing for ad hoc Heterogeneous Sensor Networks. *International Journal of High Performance Computing Applications*, 2002.

[54] Costa, P. et al. The RUNES Middleware: A Reconfigurable Component-based Approach to Networked Embedded Systems. *Proc. PIMRC*, 2005.

[55] Culler, D. et al. Overview of Sensor Networks. *IEEE Computer Magazine*, August, 2004.

[56] Culpepper, B.J. et al. Design and analysis of Hybrid Indirect Transmissions (HIT) for data gathering in wireless micro sensor networks. *SIGMOBILE Mobile Computing and Communications Review*, Vol.8, No.1, pp.6183, 2004.

[57] Curino, C. et al. TinyLIME: Bridging Mobile and Sensor Networks through Middleware. *Proc. PerCom*, 2005.

[58] Dai, H. et al. TSync: a lightweight bidirectional time synchronization service for wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol.8, No.1, pp. 125 139, 2004.

[59] Dai, H. et al. ELF: an efficient log-structured flash file system for micro sensor nodes. *Proc. SenSys*, 2004.

[60] Daniel, D. et al. Introduction to high level synthesis. *IEEE Des. Test*, vol. 11, no. 4, pp. 4454, 1994.

[61] Demers, A. et al. The Cougar Project: a work-in-progress report. *ACM SIGMOD*,Vol. 32, No. 4, pp. 53-59, 2003

[62] Du, W. et al. A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge. *Proc. Infocom*, 2004.

[63] Dunkels, . et al. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. *Proc. EmNetSI*, 2004.

[64] Ee, C.T. et al. Congestion control and fairness for many-to-one routing in sensor networks. *Proc. SenSys*, 2004.

[65] Elson, J. et al. Time Synchronization for Wireless Sensor Networks. *Proc. Int. Parallel and Distributed Processing Symposium*, 2001.

[66] Elson, J. et al. Sensor networks: A bridge to the physical worlds. *Wireless Sensor Networks Book, Kluwer Academic Publishers*, 2004.

[67] Enz, C.C. et al. WiseNET: An Ultralow Power Wireless Sensor Network Solution. *IEEE Computer*, Vol. 37, No. 8, pp.62-70,2004.

[68] Essa, I.A. et al. Ubiquitous Sensing for Smart and Aware Environments. *IEEE Personal Communications*, pp.47-49, 2000.

[69] Estrin, D. et al. Next Century Challenges: Scalable Coordination in Sensor Networks. *Proc. MobiCom*, 1999.

[70] Ferscha, A. et al. A Light-Weight Component Model for Peer-to-Peer Applications. *Proc. MDC*, 2004.

[71] Fang, Q. et al. Locating and Bypassing Routing Holes in Sensor Networks. *Proc. Infocom*, 2004.

[72] Gay, D. et al. The nesC language: A holistic approach to networked embedded systems. *Proc. SIGPLAN*, 2003.

[73] Gehrke, J. et al. Query Processing in Sensor Networks. *IEEE Pervasive Computing*, Vol.3, No.1, pp. 46-55, 2004.

[74] Gibbons, P.B. et al. IrisNet: An Architecture for a Worldwide Sensor Web. *IEEE Pervasive Computing*, Vol.2, No.4, pp. 22-33, 2003.

[75] Greenstein, B. et al. A distributed index for features in sensor networks. *Proc. SNPA*, 2003.

[76] Greenstein, B. et al. A sensor network application construction kit (SNACK). *Proc. SenSys*, 2004.

[77] Grimm, T. et al. A system architecture for pervasive computing. *Proc. ACM SIGOPS European Workshop*, 2000.

[78] Gui, C. et al. Power conservation and quality of surveillance in target tracking sensor networks. *Proc. MobiCom*, 2004.

[79] Gummadi, R. et al. Macro-programming Wireless Sensor Networks using Kairos. *Proc. DCOSS*, 2005.

[80] Han, Q. et al. AutoSeC: An Integrated Middleware Framework for Dynamic Service Brokering. *IEEE Distributed Systems Online*, Vol.2, No.7, 2001.

[81] Han, C. et al. A dynamic operating system for sensor nodes. *Proc. MobiSys*, 2005.

[82] Han, Q. et al. Energy Efficient Data Collection in Distributed Sensor Environments. *Proc. ICDCS*, 2004.

[83] Harel, D. et al. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, vol. 8, no. 3, pp. 231274, 6 1987.

[84] Harter, A. et al. The Anatomy of a Context-Aware Application. *Proc. MobiCom*, 1999.

[85] Harvard Univ . *http://www.eecs.harvard.edu/ werner/projects/volcano/*.

[86] He, T. et al. SPEED: A stateless protocol for real-time communication in sensor networks. *Proc. ICDCS*, 2003.

[87] He, T. et al. Energy-efficient surveillance system using wireless sensor networks. *Proc. MobiSys*, 2004.

[88] Heidemann, J. et al. Building efficient wireless sensor networks with low-level naming. *Proc. ACM SOSP*, 2001.

[89] Heinzelman, W. et al. Middleware to support sensor network applications. *IEEE Network*, Vol. 18, No.1, pp.6-14, 2004.

[90] Heizelman, W. et al. Adaptive protocols for information dissemination in wireless sensor networks. *Proc. Mobicom*, 1999.

[91] Heinzelman, W. et al. nergy-efficient communication protocol for wireless sensor networks. *Proc. the Hawaii International Conference System Sciences*, 2000.

[92] Heizelman, W. et al. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, Vol. 1, No.4, pp. 660-670, 2002.

[93] Helal, S. et al. The Gator Tech Smart House: A Programmable Pervasive Space. *IEEE Computer*, Vol. 38, No. 3, pp. 50-60, 2005.

[94] Hellerstein, J.M. et al. eyond Average:Toward Sophisticated Sensing with Queries. *Proc. IPSN*, 2003.

[95] Helmy, A. et al. CAPTURE: location-free contact-assisted power-efficient query resolution for sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol.8, No.1, pp. 27 47, 2004.

[96] Hill, J. et al. System architecture directions for networked sensors. *Proc. ACM Int. Conf.on Architectural Support for Programming Languages and Operating Systems*, 2000.

[97] Hoblos, G. et al. Optimal design of fault tolerant sensor networks. *Proc. IEEE EEE International Conference on Control Applications*, 2000.

[98] Holmquist, L.E. et al. Building Intelligent Environments with Smart-Its. *IEEE Computer Graphics and Applications*, vol.24 No.1, pp. 56-64, 2004.

[99] Hong, Y.W. et al. Time synchronization and reach-back communications with pulse-coupled oscillators for UWB. *Proc. Ultra Wideband Systems and Technologies*, 2003.

[100] Hu, Y. et al. SEAD: Secure Ef.cient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. *Proc. WMCSA*, 2002.

[101] Hu, Y. et al. Ariadne: A Secure On-Demand Routing Protocol for Ad hoc Networks. *Proc. MobiCom*, 2002.

[102] Hu, L. et al. Localization for mobile sensor networks. *Proc. MobiCom*, 2004.

[103] Hu, X. et al. A novel route update design for wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol.8, No.1, pp. 18 26, 2004.

[104] Hua, K.A. et al. WISE: A Web-Based Intelligent Sensor Explorer Framework for Publishing, Browsing, and Analyzing Sensor Data over the Internet. *Proc. ICWE*, 2004.

[105] Hui, J. et al. The dynamic behavior of a data dissemination protocol for network programming at scale. *Proc. SenSys*, 2004.

[106] Hull, B. et al. Mitigating congestion in wireless sensor networks. *Proc. SenSys*, 2004.

[107] Hwang, . et al. A Design and Implementation of Wireless Sensor Gateway for Efficient Querying and Managing through World Wide Web. *IEEE Transactions on Consumer Electronics*, Vol. 49, No. 4, pp. 1090-1097, 2003.

[108] Intanagonwiwat, C. et al. Directed diffusion: a scalable and robust communication paradigm for sensor networks. *Proc. Int. Conf. on Mobile Computing and Networking*, 2000.

[109] Intanagonwiwat, C. et al. Declarative Resource Naming for Macroprogramming Wireless Networks of Embedded Systems. *Univ. Calif. San Diego Technical Report CS2004-0800*, 2004.

[110] Jacobs, A. et al. A Framework for Comparing Perspectives on Privacy and Pervasive Technologies. *IEEE Pervasive Computing*, Vol.2, No.4, 2003.

[111] Ji, X. et al. ensor Positioning in Wireless Ad-hoc Sensor Networks with Multidimensional Scaling. *Proc. Infocom* , 2004.

[112] Juang, P. et al. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. *Proc. ASPLOS X*, 2002.

[113] Kalpakis, K. et al. Maximum lifetime data gathering and aggregation in wireless sensor networks. *Proc. NETWORKS*, 2002.

[114] Kambil, A. et al. Auto-ID across the value chain: from dramatic potential to greater efficiency and profit. *Tech. Rep. ACN-AUTOID-BC-001, MIT*, 2002.

[115] Kang, J. et al. End-to-end channel capacity measurement for congestion control in sensor networks. *Proc. SANPA*, 2004.

[116] Kang, P. et al. Smart Messages: A Distributed Computing Platform for Networks of Embedded Systems. *The Computer Journal*, Vol. 47, No. 4, pp. 475-494, 2004.

[117] Karl, H. et al. A short survey of wireless sensor networks. *Technical Report TKN-03-018, Technocal University Berlin*, 2003.

[118] Karlof, C. et al. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad Hoc Networks*, 1(2-3), 2003.

[119] Karlof, C. et al. TinySec: a link layer security architecture for wireless sensor networks. *Proc. SenSys*, 2004.

[120] Kasten, O. et al. Beyond Event Handlers: Programming Wireless Sensors with Attributed State Machines. *Proc. IPSN*, 2005.

[121] Katsiri, E. et al. An Extended Publish/Subscribe Protocol for Transparent Subscriptions to Distributed Abstract State in Sensor-Driven Systems using Abstract Events. *Proc. DEBS*, 2004.

[122] Kim, Y. et al. Modeling and analyzing the impact of location inconsistencies on geographic routing in wireless networks. *Proc.* , Vol.8, No.1, pp. 48 60, 2004.

[123] Klavins, E. Distributed Algorithms for Cooperative Control. *IEEE Pervasive Computing*, Vol. 3, No. 1, pp. 56-65, 2004.

[124] Kling, R. et al. Intel Mote: An Enhanced Sensor Network Node. *Proc. Int. Workshop on Advanced Sensors, Structural Health Monitoring and Smart Structures*, 2003.

[125] Kochhal, M. et al. Role-based hierarchical self organization for wireless ad hoc sensor networks. *Proc. WSNA*, 2003.

[126] Kumar, R. et al. DFuse: A Framework for Distributed Data Fusion. *Proc. SenSys*, 2003.

[127] Kumar, S. et al. On k-coverage in a mostly sleeping sensor network. *Proc. MobiCom*, 2004.

[128] Kumar, M. et al. Efficient data aggregation middleware for wireless sensor networks. *Proc. MASS*, 2004.

[129] Kwon, T. J. et al. Efficient Flooding with Passive Clustering (PC) in Ad Hoc Networks. *Computer Communication Review*, 32(1):4456, 2002.

[130] Lampe, M. et al. The Potential of RFID for Moveable Asset Management. *Workshop on Ubiquitous Commerce, Ubicomp 03*, 2003.

[131] Lazaridis, I. et al. QUASAR: Quality Aware Sensing Architecture. *ACM SIGMOD Record*, 2004.

[132] Levis, P. et al. Mat: a tiny virtual machine for sensor networks. *Proc. USENIX/ACM ASPLOS X*, 2002.

[133] Levis, P. et al. TOSSIM: accurate and scalable simulation of entire TinyOS applications. *Proc. SenSys*, 2003.

[134] Levis, P. et al. Active Sensor Networks. *Proc. USENIX/ACM Symposium NSDI*, 2005.

[135] Lewis, P. et al. Wireless Sensor Networks - Smart Environments: Technology Protocols and Applications. *Chapter 2*, Wiley-InterScience, 2005.

[136] Li, S. et al. Event Detection Services Using Data Service Middleware in Distributed Sensor Networks. *Proc. Int. Workshop on Information Processing in Sensor Networks*, 2003.

[137] Li, Q. et al. Global Clock Synchronization in Sensor Networks. *Proc. Infocom*, 2004.

[138] Lindsey, S. et al. PEGASIS: Power Efficient GAthering in Sensor Information Systems. *Proc. IEEE Aerospace Conference*, 2002.

[139] Liu, J. et al. State-Centric Programming for Sensor-Actuator Network System. *IEEE Pervasive Computing*, Vol.2, No.4, pp.50-62, 2003.

[140] Liu, T. et al. Impala: a middleware system for managing autonomic, parallel sensor systems. *Proc. ACM SIGPLAN*, 2003.

[141] Liu, T. et al. Implementing software on resource-constrained mobile sensors: experiences with Impala and ZebraNet. *Proc. MobiSys*, 2004.

[142] Liu, H. et al. Design and Implementation of a Single System Image Operating System for Ad Hoc Networks. *Proc. MobiSys*, 2005.

[143] Lucarelli, D. et al. Decentralized synchronization protocols with nearest neighbor communication. *Proc. SenSys*, 2004.

[144] Ma, Y. et al. System Lifetime Optimization for Heterogeneous Sensor Networks with a Hub-Spoke Topology. *IEEE Transactions on Mobile Computing*, Vol.3, No.3, pp. 286-294, 2004.

[145] Madden, S. et al. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. *Proc. OSDI*, 2002.

[146] Madden, S. et al. TinyDB: An Acqusitional Query Processing System for Sensor Networks. *ACM Transactions on Database Systems*, Vol.30, No.1, 2005.

[147] Mainland, G. et al. Using virtual markets to program global behavior in sensor networks. *Proc. SIGOPS European Workshop*, 2004.

[148] Mainwaring, A. et al. Wireless Sensor Networks for Habitat Monitoring. *Proc. WSNA*, 2002.

[149] Manjeshwar, A. et al. TEEN : A Protocol for Enhanced Efficiency in Wireless Sensor Networks. *Proc. International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, 2001.

[150] Mark, A. et al. Providing Application QoS through Intelligent Sensor Management. *Proc. SNPA*, 2003.

[151] Marti, M. et al. Shooter Localization in Urban Terrain. *IEEE Computer*, Vol. 37, No. 8, pp. 60-61, 2004.

[152] Marti, M. et al. The flooding time synchronization protocol. *Proc. SenSys*, 2004.

[153] Martinez, K. et al. Environmental Sensor Networks. *IEEE Computer*, Vol. 37, No. 8, pp. 50-56, 2004.

[154] Martinez, K. et al. GLACSWEB: A Sensor Web for Glaciers. *Proc. EWSN*, 2004.

[155] Melodia, T. et al. Optimal Local Topology Knowledge for Energy Efficient Geographical Routing in Sensor Networks. *Proc. Infocom*, 2004.

[156] Michahelles, F. et al. Applying Wearable Sensors to Avalanche Rescue. *Computers and Graphics*, 27(6):839-847, 2003.

[157] Mills, D.L. Internet Time Synchronization: The Network Time Protocol. *IEEE trans. Communications*, 39(10), 1991.

[158] Moore, D. et al. Robust distributed network localization with noisy range measurements. *Proc. SenSys*, 2004.

[159] Murphy, A. et al. MiLAN: Middleware Linking Applications and Networks. *TR-795, University of Rochester, Computer Science*, 2002.

[160] Nagpal, R. et al. Organizing a global coordinate system from local information on an ad hoc sensor network. *Proc. workshop IPSN*, 2003.

[161] Nath, S. et al. IrisNet: An Architecture for Enabling Sensor-Enriched Internet Service. *ntel Research Pittsburgh Technical Report IRP-TR-03-04*, 2003.

[162] Nath, S. et al. A Distributed Filtering Architecture for Multimedia Sensors. *Proc. BaseNets*, 2004.

[163] Newton, R. et al. Region Streams: Functional Macroprogramming for Sensor Networks. *Proc. DMSN*, 2004.

[164] Newton, R. et al. Building up to Macroprogramming: An Intermediate Language for Sensor Networks. *Proc. IPSN*, 2005.

[165] Niculescu, D. et al. Positioning in ad hoc sensor networks. *IEEE Network*, Vol. 18, No. 4, pp. 24- 29, 2004.

[166] Open Grid Services Architecture *http://www.ggf.org/ogsa-wg/*.

[167] The OSGi Alliance The OSGi framework. *http://www.osgi.org*, 1999.

[168] Papadimitratos, P. et al. Secure Routing for Mobile Ad hoc Networks. *Proc. CNDS*, 2002.

[169] Peng, R. et al. A Web Services Environment for Internet-Scale Sensor Computing. *Proc. SCC*, 2004.

[170] Perrig, A. et al. SPINS: Security Protocols for Sensor Networks. *Wireless Networks Journal (WINET)*, 8(5):521.534, September 2002.

[171] Perrig, A. et al. Security in wireless sensor networks. *CACM*, Vol. 47, No. 6, pp. 53-57, 2004.

[172] Polastre, J. et al. Versatile low power media access for wireless sensor networks. *Proc. SenSys*, 2004.

[173] Prakash, R. et al. Causality and the Spatial-Temporal Ordering in Mobile Systems. *Mobile Networks and Applications*, 9(5):507-516, 2004.

[174] Ramasamy, H.V. et al. Quantifying the cost of providing intrusion tolerance in group communication systems. *Proc. DSN*, 2002.

[175] Rao, R. et al. Purposeful Mobility for Relaying and Surveillance in Mobile Ad Hoc Sensor Networks. *IEEE Transactions on Mobile Computing*, Vol.3, No.3, pp.225-232, 2004.

[176] Ratnasamy, A. et al. GHT: A geographic hash table for data-centric storage in sensornets. *Proc. WSNA*, 2002.

[177] Ravelomanana, V. et al. Extremal Properties of Three Dimensional Sensor Networks with Application. *IEEE Transactions on Mobile Computing*, Vol.3, No.3, pp. 246-257, 2004.

[178] Reason, J.M. et al. A study of energy consumption and reliability in a multi-hop sensor network. *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol.8, No.1, pp. 8497, 2004.

[179] Rodoplu, V. et al. Minimum energy mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 1999.

[180] Rentala, P. et al. Survey on Sensor Networks. *Technical Report, UTDCS-33-02, University of Texas at Dallas*, 2002.

[181] Römer, K. et al. Middleware Challenges for Wireless Sensor Networks. *ACM Mobile Computing and Communication Review*, October 2002.

[182] Römer, K. et al. The Design Space of Wireless Sensor Networks. *IEEE Wireless Communications*, 2004.

[183] Römer, K. et al. Generic role assignment for wireless sensor networks. *Proc. ACM SIGOPS European Workshop*, 2004.

[184] Römer, K. Time Synchronization in Ad Hoc Networks. *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc01)*, 2001.

[185] Reconfigurable Ubiquitous Networked Embedded Systems . *http://www.ist-runes.org*, 2005.

[186] Sadagopan, N. et al. The ACQUIRE mechanism for efficient querying in sensor networks. *Proc. International Workshop on Sensor Network Protocol and Applications*, 2003.

[187] Sames, D. et al. Developing a heterogeneous intrusion tolerant corba system. *Proc. DSN*, 2002.

[188] Satyanarayanan, M. et al. Of Smart Dust and Brilliant Rocks. *IEEE Pervasive Computing*, Vol.2, No.4, pp.2-4, 2003.

[189] Schurgers, C. et al. Energy efficient routing in wireless

sensor networks. *MILCOM Proceedings on Communications for Network-Centric Operations: Creating the Information Force*, 2001.

[190] Schramm, P. et al. A Service Gateway for Networked Sensor Systems. *IEEE Pervasive Computing*, Vol.3, No.1, pp. 66-74, 2004.

[191] Seada, K. et al. Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. *Proc. SenSys*, 2004.

[192] Shah, R. et al. Energy Aware Routing for Low Energy Ad Hoc Sensor Networks. *Proc. WCNC*, 2002.

[193] Shang, Y. et al. Improved MDS-Based Localization. *Proc. Infocom*, 2004.

[194] Shen, C.C. et al. Sensor Information Networking Architecture and Applications. *E Personal Communications Magazine*, Vol.8, No. 4. pp. 52-59, 2001.

[195] Shneidman, J. et al. Hourglass: An Infrastructure for Connecting Sensor Networks and Applications. *Harvard Technical Report TR-21-04*, 2004.

[196] Shnayder, V. et al. Simulating the power consumption of large-scale sensor network applications. *Proc. SenSys*, 2004.

[197] Shum, L. et al. Distributed Algorithm Implementation and Interaction in Wireless Sensor Networks. *Proc. workshop SANPA*, 2004.

[198] Sibley, G.T. et al. Robomote: A Tiny Mobile Robot Platform for Large-Scale Sensor Networks. *Proc. ICRA*, 2002.

[199] Simon, G. et al. Sensor network-based counter sniper system. *Proc. Sensys*, 2004.

[200] Sivaharan, T. et al. Cooperating Sentient Vehicles for Next Generation Automobiles. *Proc. WAMES*, 2004.

[201] Sivrikaya, F. et al. Time synchronization in sensor networks: a survey. *IEEE Network*, Vol. 18, No. 4, pp.45- 50, 2004.

[202] Slijepcevic, S. et al. Power efficient organization of wireless sensor networks. *Proc. ICC*, 2001.

[203] Smaragdakis, G. et al. SEP: A Stable Election Protocol for clustered heterogeneous wireless sensor networks. *Proc. SANPA*, 2004.

[204] Sohrabi, K. et al. Protocols for Self Organization of a Wireless Sensor Network. *Personal Communication Magazine*, 7:1627, 2000.

[205] Sohrabi, K. et al. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, Vol.7, No.5, pp.16-27, 2000.

[206] Sohrabi, K. et al. Methods for Scalable Self-Assembly of Ad Hoc Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, Vol.3,No.4, pp. 317-331, 2004.

[207] Son, D. et al. The Effect of Mobility-Induced Location Errors on Geographic Routing in Mobile Ad Hoc and Sensor Networks: Analysis and Improvement Using Mobility Prediction. *IEEE Transactions on Mobile Computing*, Vol.3, No.3, pp. 233-245, 2004.

[208] Souto, E. et al. A message-oriented middleware for sensor networks. *Proc. workshop on Middleware for pervasive and ad-hoc computing*, 2004.

[209] Srisathapornphat, C. et al. Sensor Information Networking Architecture. *Proc. Int. Workshops on Parallel Processing* , 2000.

[210] Steffan, J. et al. Scoping in wireless sensor networks. *Proc. workshop on Middleware for pervasive and ad-hoc computing*, 2004.

[211] Subramanian, L. et al. An Architecture for Building Self Configurable Systems. *Proc. IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing*, 2000.

[212] Subramanian, L. et al. An architecture for building selfconfigurable systems. *Proc. MobiHoc*, 2000.

[213] Szewczyk, R. et al. An analysis of a large scale habitat monitoring application. *Proc. SenSys*, pp. 214 - 226, 2004.

[214] Tilak, S. et al. A Taxonomy of Wireless Micro-Sensor Network Models. *Mobile Computing and Communications Review*, Vol.6, No.2, 2002.

[215] Trakadas, P. et al. Efficient routing in PAN and sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol.8, No.1, pp. 10 17, 2004.

[216] UAV. The 29 Palms Experiment: Tracking vehicles with a UAV-delivered sensor network. *tinyos.millennium.berkeley.edu/29Palms.htm.*

[217] Vahid, F. et al. SpecCharts: A VHDL Front-End for Embedded Systems. *Proc. Trans. on CAD*, vol. 14, no. 6, pp.694706, 1995.

[218] Wanat, R. et al. Enabling Ubiquitous Sensing with RFID. *IEEE Computer*, pp. 84-86, 2004.

[219] Wang, G. et al. Movement-Assisted Sensor Deployment. *Proc. Infocom*, 2004.

[220] Welsh, M. et al. Programming sensor networks with abstract regions. *Proc. NSDI*, 2004.

[221] Whitehouse, K. et al. Hood: A neighborhood abstraction for sensor networks. *Proc. MobiSys*, 2004.

[222] Whitehouse, K. et al. Semantic streams: A framework for declarative queries and automatic data interpretation. *Technical Report MSR-TR-2005-45, Microsoft Research*, 2005.

[223] Woo, A. et al. Networking support for query processing in sensor networks. *CACM*, Vol. 47, No. 6, pp. 47-52, 2004.

[224] Wood, A. et al. Denial of service in sensor networks. *IEEE Computer*, 35(10):54.62, 2002.

[225] Wu, M. et al. Novel Component Middleware for Building Dependable Sentient Computing Applications. *ACM Workshop on Component-oriented approaches to Context-aware computing, ECOOP*, 2004.

[226] Xu, N. et al. A wireless sensor network for structural monitoring. *Proc. SenSys*, 2004.

[227] Xu, N. A Survey of Sensor Network Applications. University of Southern California, 2003.

[228] Xu, Y. et al. Geography-informed energy conservation for ad hoc routing. *Proc. MobiCom*, 2001.

[229] Yao, Y. et al. The Cougar Approach to In-Network Query Processing in Sensor Networks. *ACM SIGMOD*, Vol. 31, No. 3, pp.9-18, 2002.

[230] Yoneki, E. et al. Unified Semantics of Event Correlation over Time and Space in Hybrid Network Environments. *Proc. CoopIS*, 2005.

[231] Younis, M. et al. Energy-Aware Routing in Cluster-Based Sensor Networks. *Proc. MASCOT*, 2002.

[232] Younis, O. et al. Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach. *Proc. Infocom*, 2004.

[233] Yu, Y. et al. Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks. *Technical Report, UCLA-CSD TR-01-0023*, 2001.

[234] Yu, Y. et al. Issues in designing middleware for wireless sensor networks. *IEEE Network*, Vol. 18, No.1, pp. 15-21, 2004.

[235] Zhang, W. et al. Optimizing Tree Reconfiguration for Mobile Target Tracking in Sensor Networks. *Proc. Infocom*, 2004.

[236] Zhang, P. et al. Hardware design experiences in ZebraNet. *Proc. SenSys*, 2004.

[237] Zhao, F. et al. Wireless Sensor Networks : An Information Processing Approach. *Morgan Kaufmann*, 2004.

[238] Zhou, G. et al. Impact of radio irregularity on wireless sensor networks. *Proc. MobiSys*, 2004.

[239] *http://www.cs.berkeley.edu/projects/parallel/castle /split-c/.*

APPENDIX

	Deploy-ment	Mobility	Infra-struc-ture	Topology	Density (Size)	Connec-tivity	Lifetime	Node Ad-dress	Aggrega-tion	Dissemi-nation	RT
1.GreatDuck	manual-onetime	static	base, gateway	star of clusters	dense (10s-100s)	connected	7 months	id	global diffusion	send to base	-
2.ZebraNet	manual-onetime	all, con-tinuous, passive	base, GPS	graph	dense (10s-100s)	sporadic	one year	id	global diffusion	send to base	-
3.Glacier	manual-onetime	all, con-tinuous, passive	base, GPS, GSM	start	sparse (10s-100s)	connected	several months	id	global diffusion	send to base	-
4.Hearding	manual-onetime	all, con-tinuous, passive	base, GPS	graph	dense (10s-100s)	intermit-tent	days to weeks	id	global diffusion	send to base	-
5.Ocean	random, iterative	all, con-tinuous, passive	satellite	star	sparse (1300)	intermit-tent	4-5 years	-	neighbor global diffusion	forward to satel-lite	-
6.Grape	manual-onetime	static	base	tree	sparse (20m) (100s)	connected	several months	id	global diffusion	multi tier bcast	-
7.Avalanche	manual-onetime	all, con-tinuous, passive	rescuer's PDA	start	dense (10s-100s)	connected	days	id	global diffusion	bcast 802.11x	-
8.VitalSign	manual	all, con-tinuous, passive	ad hoc	single hop	dense (10s)	connected	days to months	id	composite event, diffusion	bcast	RT
9.Tracking	random	all, oc-casional, passive	UAV	graph	sparse (10s-1000s)	intermit-tent (UAV)	weeks to years	id	composite event, diffusion	bcast	RT
10.ToolBox	manual	all, oc-casional, passive	base	star	sparse (10s)	connected	months to years	id	global diffusion	send to base	-
11.Supply	manual	all, oc-casional, passive	base	star	sparse (10s)	connected	months to years	id	global diffusion	send to base	-
12.Volcano	manual	static	base	star	sparse (10s)	connected	days	id	streaming	send to base	-

Table 2: Wireless Sensor Network Application Classification

1. Bird Observation on Great Duck Island: WSN is being used to observe the breeding behavior of a small bird called Leach's Storm Petrel [148] on Great Duck Island. Nodes can measure humidity, pressure, temperature, and ambient light level. Burrow nodes are equipped with infrared sensors to detect the presence of the birds. The burrows occur in clusters and the sensor nodes form a multihop ad hoc network. The base station computer is connected to a database back-end system via a satellite link. Sensor nodes sample their sensors about once a minute and send their readings directly to the database back-end system.

2. ZebraNet: A WSN is being used to observe the behavior of wild animals within a spacious habitat (e.g., wild horses, zebras, and lions) [112] at the Mpala Research Center in Kenya. Of particular interest is the behavior of individual animals. Animals are equipped with sensor nodes. An integrated GPS receiver is used to obtain estimates of their position and speed of movement. Whenever a node enters the communication range of another node, the sensor readings and the identities of the sensor nodes are exchanged. At regular intervals, a mobile base station (e.g., a car or a plane) moves through the observation area and collects the recorded data from the animals it passes.

3. Glacier Monitoring: A WSN is being used to monitor sub-glacier environments at Briksdalsbreen, Norway [154]. A lengthy observation period of months to years is required. Sensor nodes are deployed in drill holes at different depths in the glacier ice and in the till beneath the glacier. Sensor nodes are equipped with pressure and temperature sensors and a tilt sensor for measuring the orientation of the node. Sensor nodes communicate with a base station deployed on top of the glacier. The base station measures supra-glacial displacements using differential GPS and transmits the data collected via GSM.

4. Cattle Herding: A WSN is being used to implement virtual fences, with an acoustic stimulus being given to animals that cross a virtual fence line [42]. Movement data from the cows controls the virtual fence algorithm that dynamically shifts fence lines. For the first experiment, each sensor node consists of a PDA with a GPS receiver, a WLAN card, and a loudspeaker for providing acoustic stimuli to the cattle as they approach a fence. The nodes form a multi-hop ad hoc network, forwarding movement data to a base station The base station transmits fence coordinates to the nodes.

5. Ocean Water Monitoring: The ARGO project [13] is using a WSN to observe the temperature, salinity, and current profile of the upper ocean. The goal is a quantitative description of the state of the upper ocean and the patterns of ocean climate variability. The nodes are dropped from ships or planes. The nodes cycle to a depth of 2000m every ten days. Data collected during these cycles is transmitted to a satellite while nodes are at the surface.

6. Grape Monitoring: A WSN is being used to monitor the conditions that influence plant growth (e.g., temperature, soil moisture, light, and humidity) across a large vineyard in Oregon [23]. In a first version of the system, sensor nodes are deployed across a vineyard in a regular grid about 20 meters apart. A temperature sensor is connected to each sensor node via a cable. A laptop computer is connected to the WSN via a gateway to display and log the temperature distribution across the vineyard. The sensor nodes form a two-tier multi-hop network, with nodes in the second tier sending data to a node in the first tier.

7. Rescue of Avalanche Victims: A WSN is being used to assist rescue teams in saving people buried in avalanches [156]. The goal is to better locate buried people and to limit overall damage by giving the rescue team additional indications of the state of the victims and to automate the prioritization of victims (e.g., based on heart rate, respiration activity, and level of consciousness). For this purpose, people at risk (e.g., skiers, snowboarders, and hikers) carry a sensor node that is equipped with an oximeter (a sensor which measures the oxygen level in blood), and which permits heart rate and respiration activity to be measured.

8. Vital Sign Monitoring: Wireless sensors are being used to monitor vital signs of patients in a hospital environment [20]. Compared to conventional approaches, solutions based on wireless sensors are intended to improve monitoring accuracy whilst also being more convenient for patients. Various medical sensors (e.g., electrocardiogram) may be subsequently attached to the patient.

9. Tracking Military Vehicles: A WSN is being used to track the path of military vehicles (e.g., tanks) [216]. Sensor nodes are deployed from an unmanned aerial vehicle (UAV). Magnetometer sensors are attached to the nodes in order to detect the proximity of tanks. Nodes collaborate in estimating the path and velocity of a tracked vehicle. Tracking results are transmitted to the unmanned aerial vehicle.

10. Smart Tool Box: Tools are equipped with RFID tags, and the tool box contains a mobile RFID system (including a tag reader antenna integrated into the tool box) [130]. The tool box issues a warning for safety reasons if a worker attempts to leave the building site while any tools are missing from his or her box. The box also monitors how often and for how long tools have been in use.

11. Smart Supply Chain: Smart identification technology can significantly improve the efficiency of supply chains and the internal logistics processes of companies [114]. In such scenarios, the automatic identification and localization of goods at instance level can help to prevent faulty deliveries. Every bottle of mineral water, the box containing the bottles, and the container for the boxes are tagged. RFID readers are installed along the supply chain to check whether the correct quantity of goods and the correct product instances have passed.

12. Monitoring Volcanic Eruptions: WSN monitors eruptions at Volcano Tungurahua, an active volcano in central Ecuador. This network consisted of five tiny, low-power wireless sensor nodes, three equipped with a specially-constructed microphone to monitor infrasonic (low-frequency acoustic) signals emanating from the volcanic vent during eruptions. Over 54 hours of continuous infrasound data are collected, transmitting signals over a 9 km wireless link back to a base station at the volcano observatory, (see [85] for more details).