1-1-2010

Unified Role Assignment Framework For Wireless Sensor Networks

Manish Mahendra Kumar Kochhal
Wayne State University

UNIFIED ROLE-ASSIGNMENT FRAMEWORK FOR WIRELESS SENSOR NETWORKS

by

MANISH M. KOCHHAL

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2010

MAJOR: COMPUTER ENGINEERING

Approved by:

Advisor Date

DEDICATION

To

All the teachers and masters from past, present, and future

ACKNOWLEDGMENTS

I consider myself really fortunate to have Dr. Loren Schwiebert as my advisor, guide, and a friend. Loren has been a wonderful advisor, providing me with support, encouragement, and an endless source of ideas. His breadth of knowledge and his enthusiasm for research amazes and inspires me. It was a wonderful experience to work with Loren in several fields including Interconnection Networks, Mobile Ad hoc Networks, Wireless Sensor Networks, Artificial Retina and Biomedical sensor research. I thank him for all the time he has spent with me, discussing everything from research to career choices, reading my papers, and advice on my research progress. My life has been enriched professionally, intellectually and personally by working with Loren.

I also thank the NSF, the computer science department, the electrical and computer engineering department for funding my research and studies at Wayne State.

The Unified Role-based Assignment Framework for ad hoc Wireless Sensor Networks is joint work with Loren Schwiebert and Sandeep Gupta. I thank them for many interesting discussions about sensor networks, ad hoc networks, and wireless radio in general. I would also like to thank Ayad Salhieh, Kamran Jamshaid, Fernando Martincic, Yong Xi, and Changli Jiao for providing many insights and detailed discussion about everything from Wireless Sensors, Security, Simulations and even indian-arab-chinese-american cultural diversity.

I would like to thank my project leads/managers who permitted me to work on my research during my office spare time. I also thank my colleague Dr. Venkatesh Ramaswamy at Airvana in helping me understand the complexity of developing a network model involving a number of diverging parameters. In that regard, I am

very much indebted to Professor Daniel Grosu for providing insights into the complexity of developing role-assignment algorithms for optimizing QoS and resource utilization using Game Theory and Mechanism Design.

I am very thankful to Professor Dr. Santalucia for giving me an opportunity to work with him to develop software for his research in Bio-Informatics. The algorithms and techniques used as well as the volume of coding that we did in such a short time helped me a lot in developing and honing my programming skills.

I am indebted to my parents Mahendra Kochhal and Sushma Kochhal for everything that they have given to me. I thank them for the sacrifices they made so that I could grow up in a learning environment. They have stood by me in everything I have done, providing constant support, encouragement, and love.

I would also like to thank my younger brother Amit, who has always been ready to motivate and encourage me in whatever endeavors I undertook. He seems to have mature insights about many things in life both personal and professional which is usually expected from an elder brother rather than a younger sibling.

I am grateful for the motivation and encouragement provided to me by my in-laws Dr. Rajesh Diwakar and Dr. Shanti Suman Diwakar during my dissertation writing phase. In that regard, I am also thankful to my brother-in-law Saurabh Diwakar for his support and practical advice.

I am indebted to my wife Sonal for the sacrifices she made while I was busy working at Airvana and at the same time finishing my dissertation. She has tirelessly worked hard to keep my life simple at home so that I can continue focussing on my dissertation. Considering that she herself was taking five courses as part of her MBA program and at the same time nursing her pregnancy, I feel humbled by her strength and simplicity. I am also thankful to this existence for awarding me with

a son, Abhishal.

Finally, I would like to thank my teachers at the Advaita Meditation Center, Dr. Ed Kowaloff, Dr. John Lehmann, Dr. Don Moir, Danae Wharton, and Lalla Mchugh for introducing me to the world of meditation. In that regard, I am also grateful for the wonderful opportunity to meditate with my classmates Marina Carboni, Mark Nohelty, and others. Their meditative presence and insights have helped me remain relaxed and simultaneously be aware in my professional, academic, and personal activities. I still cherish and remember the many interesting discussions that gave me a chance to learn something new from all of you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 – Introduction

The ability of a network to provide efficient networking services in terms of supporting diverging requirements such as the application-specified QoS and fair network resource usage, while dynamically adapting to the problems in the network is a very challenging and difficult issue. This problem becomes even more exacting when the devices forming such a collaborative network are heterogeneous and are crucially constrained in energy, computational, and communication capabilities. The unpredictability of the wireless communication media adds a third dimension to this challenge. In addition, the vision of having unattended and un-tethered network operation makes it even more complicated to provide even basic network activities like network discovery, network organization, routing, event monitoring, data aggregation, and network management.

An example of such a network is an ad hoc deployed wireless sensor network (WSN) that is envisioned to provide target sensing, data collection, information manipulation, and dissemination in a single, distributed, and integrated networked paradigm. Wireless sensor networks (WSNs) are made possible by the continuing improvements in embedded sensor, VLSI, and wireless radio technologies. WSNs have many possible applications in the scientific, medical, commercial, and military domains. Examples of these applications include environmental monitoring, smart homes and offices, surveillance, intelligent transportation systems, and many others.

Figure 1.1(a) shows a wireless sensor node that is crucially resource constrained to single-handedly provide an accurate and detailed sampling of the in-situ environ-

ment. WSNs therefore pursue high sensing redundancy by deploying a large number of sensors. For scalability concerns, such a large network is queried in an ad hoc fashion by a remote base station for any interesting events. Neighboring sensors that perceive change in physical readings around their area of coverage collaborate with each other to reach a consensus in order to accept genuine events and discard any spurious events. These readings are aggregated, compressed, and relayed multi-hop by sensors en-route to a data collection point (or a sink) and then eventually to a remote base station. Scalability concerns again dictate distributed protocol solutions for providing these in-network services in a localized and resource-efficient manner. Figure 1.1(b) highlights this fundamental sensor network communication paradigm.

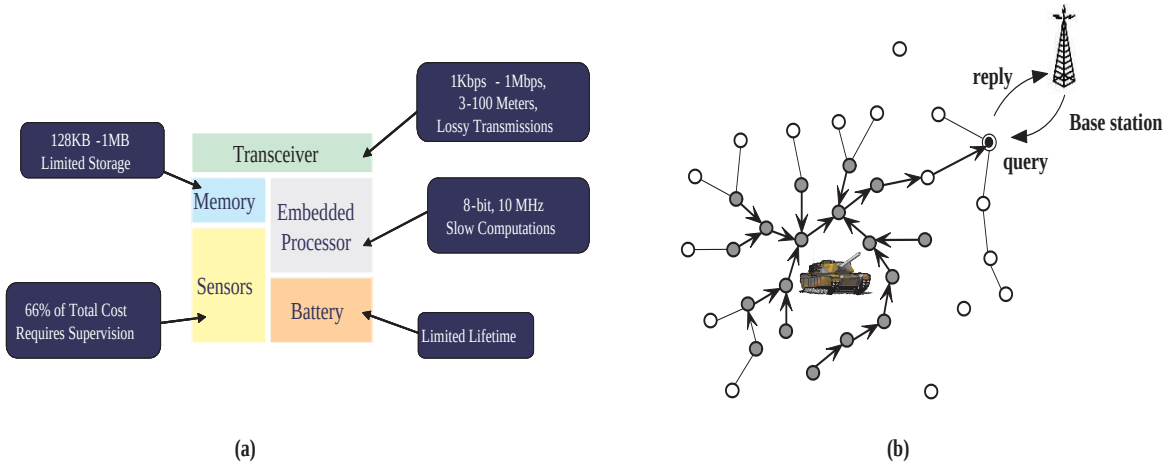

Figure 1.1: (a) Resource constrained sensor node and (b) Sensor network communication model.

One of the crucial design challenges in wireless sensor networks is energy efficiency. This is because individual sensor nodes use a small battery as a power source and re-charging or replacing batteries in a remote environment is not feasible. Thus, to achieve a longer network lifetime, one has to tackle energy efficiency at all levels

of the sensor network infrastructure. Since the wireless radio is the primary energy consumer in a sensor node, systematic management of network communications becomes critical.

With the need to dynamically sense, monitor, and track multiple events or a single event appearing randomly at different places in the sensor network, it becomes necessary for several network services such as target detection and tracking, data gathering, forwarding and routing sensing data to a data sink to simultaneously compete for network resources. This calls for resource arbitration among services which itself needs to be complemented by imperative inter-resource communications among nodes, thus resulting in further degradation in application *QoS* by way of increasing latency and decreasing network lifetime. Thus, distributed resource management is a complicated multi-dimensional problem. It needs to be collaboratively and adaptively tackled across multiple services and nodes to support collective arbitration of the use of local and remote resources. It also needs to identify protocol scenarios where it can meet application-desired QoS and implement these solutions efficiently. Similarly, in scenarios where desired requirements are mutually exclusive and cannot be met, it should identify performance tradeoffs and implement these as protocol contingencies.

In this dissertation, we propose a *unified role based abstraction* framework that provides a common platform for protocols to coherently resolve important services versus resources issues in terms of an adaptive organization and scheduling of network resources. Our framework models application entities as roles and network dynamics as changes in node capabilities. In other words, application services are mapped as network roles played by local sensors with node resources used as rules for role identification. The unified role-based abstraction framework thus logically

unifies both the tasks and the resources needed to execute these tasks as roles and rules, respectively.

In distributed systems, network services are usually executed by a collection of tasks operating over a set of resources locally provisioned around a certain k-hops vicinity. We abstract the basic tasks and their principal resource requirements in terms of elementary roles such as idler, sensor, cacher, processor, and transmitter, which in turn correspond to the use of no resources, sensing, memory, and the radio transceiver resources, respectively. These roles, though abstract and implicit, expose role-specific resource controls for load balancing by way of role assignment and scheduling. This natural service to role mapping allows existing protocol solutions to meet differing application requirements in terms of agreeable *tradeoffs* or *performance thresholds* (if possible) without loss of generality. To the best of our knowledge, a generic role-based framework that provides a simple and unified network management solution has not been proposed previously.

The role-based abstraction framework revolves around the concept of nodes assuming roles depending upon available resources to collaboratively provide application services. For WSNs, the requirement of services basically stems from the events being sensed locally by the sensors. One potential ramification of such a requirement is that although the use of sensing resources is mostly pre-allocated at the site of an event, the use of other resources such as processing and storage can be offloaded to a later hop. This use of distant resources in a distributed fashion gives rise to additional communication overhead. The role-abstraction framework is extensible and supports composition of simpler roles into more complex user-defined roles to enable a specific protocol solution of a certain service requirement. In this dissertation, we consider an event monitoring service to be expressed in

terms of user-defined complex roles consisting of a region of sensing *collaborators* that are managed by a sensing *coordinator*; *router* roles organized as a group of neighboring *relays* who forward information from these coordinators to a remote base station. We develop adaptive and energy-efficient role-assignment protocols for mapping such data-aggregation solutions to a set of resource-constrained wireless sensor nodes forming a network.

This chapter highlights the fundamental concepts of *hierarchy*, *approximation*, *aggregation*, and *redundancy* that are used to deal with complexities across large-scale distributed systems. These concepts form the basis of our proposed role-based framework. The use of a layered protocol stack for an embedded resource-constrained communication system, though useful, has limitations in terms of the interfaces it exposes to other layers. Chapter 1 emphasizes these limitations where different protocols request diverging optimizations across layers. A cross-layer approach is thus warranted. Alternatively, a need for protocols to be made available as independent and open entities is identified as the best approach.

The chapter then delves into the importance of the most basic requirement of any network and that is the architectural organization of its communicating entities or nodes. This analysis leads to the realization that a network organization is so fundamental that any specific architectural imposition limits the communication flexibility of nodes collaborating to execute various application services. We perceive network organization as the partitioning of nodes into groups that allows the network to meet two important network benefits of protocol scalability and localization. From the network layer, we move to our proposed unified role framework where we decompose services into tasks and assign these as roles to nodes in the network. We then come full circle where we realize that the arrangement of roles

and the communication pattern among them determines the network organization. Similarly, a specific network organization allows only a specific assignment of roles in that architecture for any service.

We provide a detailed introduction to wireless sensor networks (WSNs), their objectives, the technology enablers of such a system, and the constraints and challenges posed by various applications. We give an extreme example of a chemical spill in an industrial facility manned with different types of sensors that detect fire, storage of chemicals, and the health of its workers. We will explore how sensors in the facility acquire roles in the network to support multiple mission-critical applications desiring various services at different QoS levels. Finally, we discuss in brief our research contributions, motivations, and the organization of this dissertation.

1.1 Wireless Sensor Networks

1.1.1 Technological drivers

As technologies in hardware integration such as VLSI (Very Large Scale Integration), MEMS (Micro-Electro-Mechanical Systems), and low power radio advance several new opportunities for embedded systems design emerge with diverse challenges and characteristics in contrast to the traditional desktop and server systems. One of the most interesting applications of the embedded and networked design systems are wireless sensor networks (WSNs). The networked sensor is principally enabled by "Moore's Law" pushing computing and storage into smaller, cheaper, and lower-power units.

Additionally, trends such as complete systems on a chip (SOC), integrated low-power communication, sensing devices that interact with the physical world, and advances in battery technologies are equally significant. The combination of these

technologies along with ubiquitous connectivity to the Internet makes it possible to envision embedded autonomous devices that unobtrusively interact with the physical world and make this information available to remote servers and desktop systems. The sensors will interact with the physical environment to detect light, heat, position, movement, chemical presence, and so on. In each of these areas, the technology is leading to significant developments that makes networked sensors an exciting regime to apply systematic design methods.

1.1.2 Applications: Civil, Medical, Industrial, and Military

Wireless sensor networks have the following broad application objectives:

1. Reliable monitoring of a variety of environments.
2. Enable in-network information gathering and processing.
3. Integrate physical sensing and controlling capability with a communication-oriented infrastructure, say Internet.
4. Support a variety of applications with varying levels of QoS across several domains, hardware, and software (see figure 1.2).

An example of sensor networks used for civilian purposes could be smart homes and offices. Here sensors could be deployed to control appliances and electrical devices in the house. Sensors can be used to provide better lighting and heating in office buildings. The Pentagon building has used sensors extensively in this regard.

Sensor networks can be used for medical purposes as well. Examples include the use of biomedical sensors to monitor the glucose level, heart rate, and detect cancers. In hospitals, a network of sensors on the body of the patient and in-vivo can monitor the vital signs and record anomalies. An array of sensors on

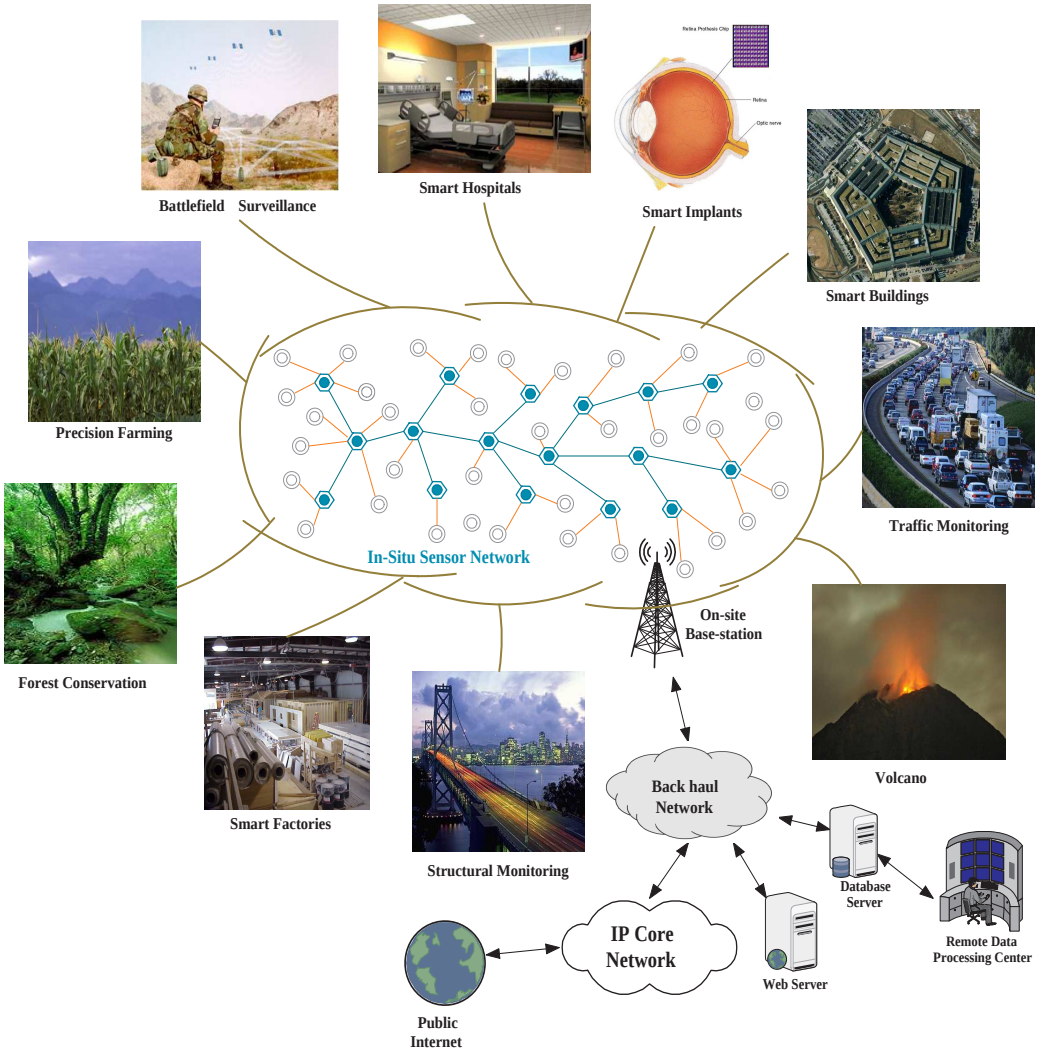

Figure 1.2: Applications of Wireless Sensor Networks.

a chip can be implanted to substitute for and supplement a missing or defective part of the body. The Artificial Retina Implant Project at Wayne State University uses a retinal prosthetic sensor chip to allow the patient suffering from macular degeneration of the retina to perceive limited vision. Similarly, sensors are used for cochlear implants.

Remote deployment of military sensor networks can be used for tactical monitoring of enemy troop movements. These networks provide opportunities for soldiers to achieve situational awareness in terms of supporting real-time troop collaboration, status exchange, and coordination.

Industrial applications of sensor networks include monitoring and controlling agricultural crop conditions. In enterprise scale manufacturing and retail companies, sensor networks can be used to monitor inventory and support in-process parts tracking. These networks can automatically report problems at various stages such as in-plant manufacturing, packaging, and equipment maintenance. RFIDs (Radio Frequency ID) are examples of sensor nodes used in retail shops as theft deterrent and for customer tracing.

Figure 1.2 shows a number of applications of sensor networks. It also highlights the elementary organization of the underlying sensor network infrastructure. For certain application domains, sensors can be deployed randomly or placed in-situ in a regular fashion. In the figure, we see a group of randomly deployed sensors self-organizing themselves into a 2-level hierarchical network organization. The nodes at the higher level form a backbone network to route queries and sensing data among sensors and to the basestation. The on-site basestation has additional hardware, uninterruptible power supply, and a powerful radio to communicate with the whole network. In certain cases, a number of basestations can be deployed within and

on the perimeter of the network to facilitate better communication with the sensor network.

The basestation acts as a gateway between the remote data processing center and the sensor network. In case of applications similar to biological implants, the sensor network, the basestation, and the data-processing facility are all in the same location, that is, usually with the person who has the implant. In case of hazardous and unpredictable environments, the back haul network acts as a transit network to the offshore location where powerful computers mine the data collected from the sensor network and maintain the relevant information in the database server. An additional on-site web server makes this information available to the public via the Internet.

1.1.3 System Overview: Hardware and Software

Wireless sensor networks have one fundamental requirement that is inherent across innumerable applications domains. And that is the requirement to sense. This requirement forms the basis for nodes to participate and form a network to share this information. Thus, the choice of the hardware and software for any sensing system is highly influenced by the characteristics of the sensing event, the application level requirements for data collection from the network, and the nature of the deployment environment. From a hardware perspective, typical sensor nodes differ in terms of capabilities to include a low end on-chip processor, small memory, sensors along with analog-to-digital converters, and transceiver chips, all of these assembled to form a tiny, programmable, application-specific, radio-equipped sensing device.

Figure 1.3 highlights the basic architecture of the most popular Mica2 sensor

node platform from Crossbow Technology Inc. (CrossBow). Since the philosophy was to have a sensor platform that forms a truly ad hoc network, the Mica2 was equipped to be powered by 2 AA batteries. This resulted in the use of a low power CPU and a wireless radio so as to increase the lifetime of the Mica2 mote. To allow for sensing flexibility, a 51-bit I/O connector was provided on-board that is reusable across software development, debugging, and deployment life-cycles. The Mica2 mote, when mounted to a programming board, allows software developed on the host computer to be downloaded and installed on the mote through a serial port connecting the host computer and the programming board (refer to figure 1.5 (a) and (d)). Newer generations of the Mote platform allow for more novel ways of code development and installation. Similarly, a JTAG host debugger connecting to the Mica2 mote through the 51 pin I/O connector allows for real-time debugging and code profiling.

Figure 1.4 shows a number of sensor boards that integrate custom sensors to sense the physical environment for light, temperature, humidity, barometric pressure, and sometimes even record live video. Figure 1.4 (d) shows a sensing board that has both the sensors and actuators that can act upon certain environmental conditions. The nodes are called XSMs (Extreme Scale Mote), and were designed by The Ohio State University and CrossBow Technology for the DARPA Extreme Scaling project, code-named "ExScal". They feature a variety of sensors and actuators including a magnetometer, a microphone, four passive infrared receivers, a photocell, a sounder, and feedback LEDs.

The application for which ExScal uses these high end sensors is to detect and classify multiple intruder types over an extended perimeter. This would be ideal for protecting an area that is too vast to be patrolled by human guards such as an oil

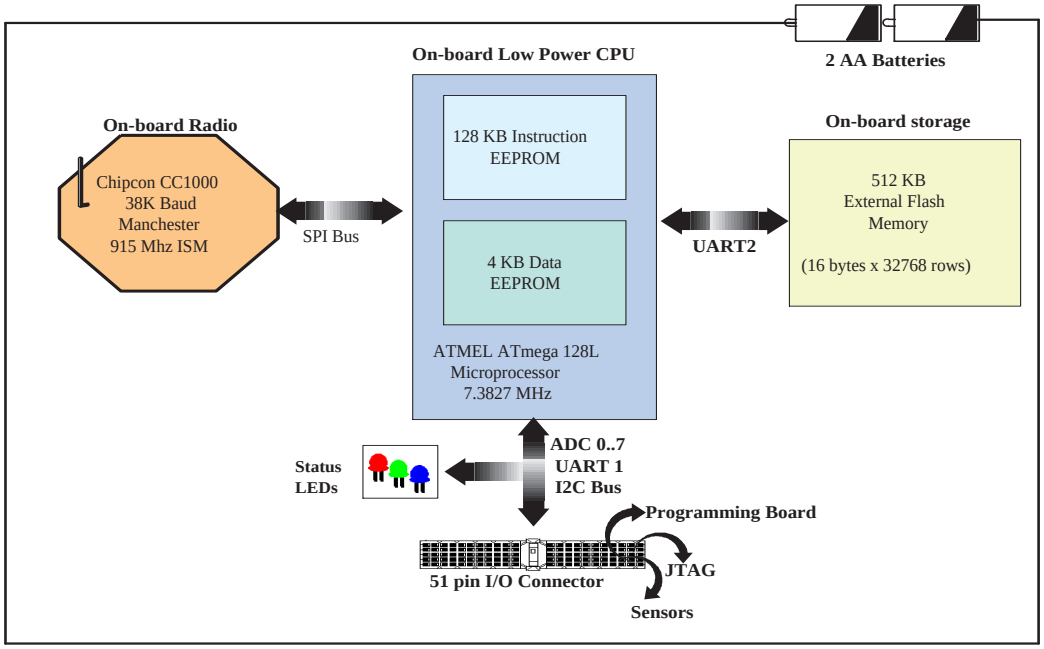

Figure 1.3: Basic architecture of a MICA2 sensor platform

pipeline or a national border. The XSM motes were organized under a second tier of devices called Extreme Scale Stargates (XSS) (shown in figure 1.5(e)) running the Intel Stargate platform. ExScal customized the stargates by adding an 802.11b Wireless Networking card with requisite software, an external antenna, a housing for the device, and a battery pack. The Stargates were placed strategically in the topology such that most motes were able to communicate with a stargate. These, tier-2 nodes ran a controller application that served to orchestrate the localization and reprogramming services at Tier 1. They also facilitated retrieving data from the motes to be analyzed on PCs (Tier 3).

Figure 1.5 shows a number of complete sensing system boards that integrate custom sensors, TCP/IP, 802.11b, Zigbee, and serial protocol stacks.

[a]Photo courtesy Crossbow Technology Inc. WWW link: `http://www.xbow.com/`

[b]Photo courtesy of the ExScal project at The Ohio State University. WWW link: `http:`

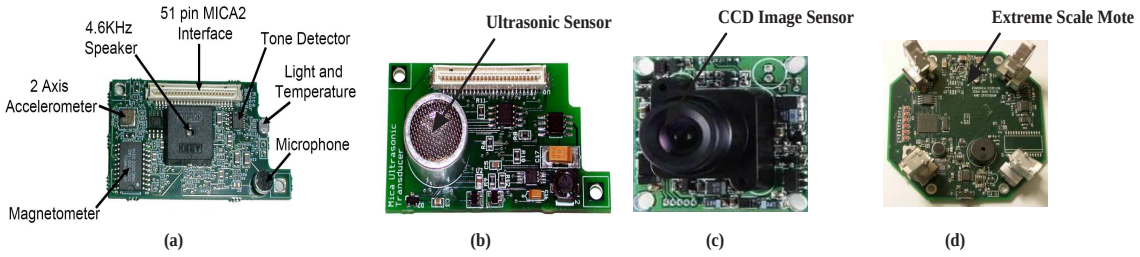

Figure 1.4: Multi modal Sensor Boards[a]

A critical requirement of the networked sensor platform is the design of a software architecture that bridges the gap between raw hardware capabilities and a useful system. The demands here are numerous. It must be efficient in terms of memory, processor, and power requirements, so that it falls within the constraints of the hardware. It must also be agile enough to allow multiple applications to simultaneously use system resources such as communication, computation, and memory. The extreme constraints of these devices makes it impractical to use legacy systems (e.g. say UNIX).

TinyOS (Hill, 2000), a tiny micro-threaded operating system is a prototype platform currently being developed at the University of California at Berkeley specifically for wireless sensors. TinyOS has the following features:

1. It is single threaded
2. It supports an open source development environment
3. It has a component-oriented programming language (NesC).
4. It's design ideology is to sleep as often as possible to save power.
5. It supports high concurrency and is interrupt driven (no polling).

[a] //cast.cse.ohio-state.edu/exscal/

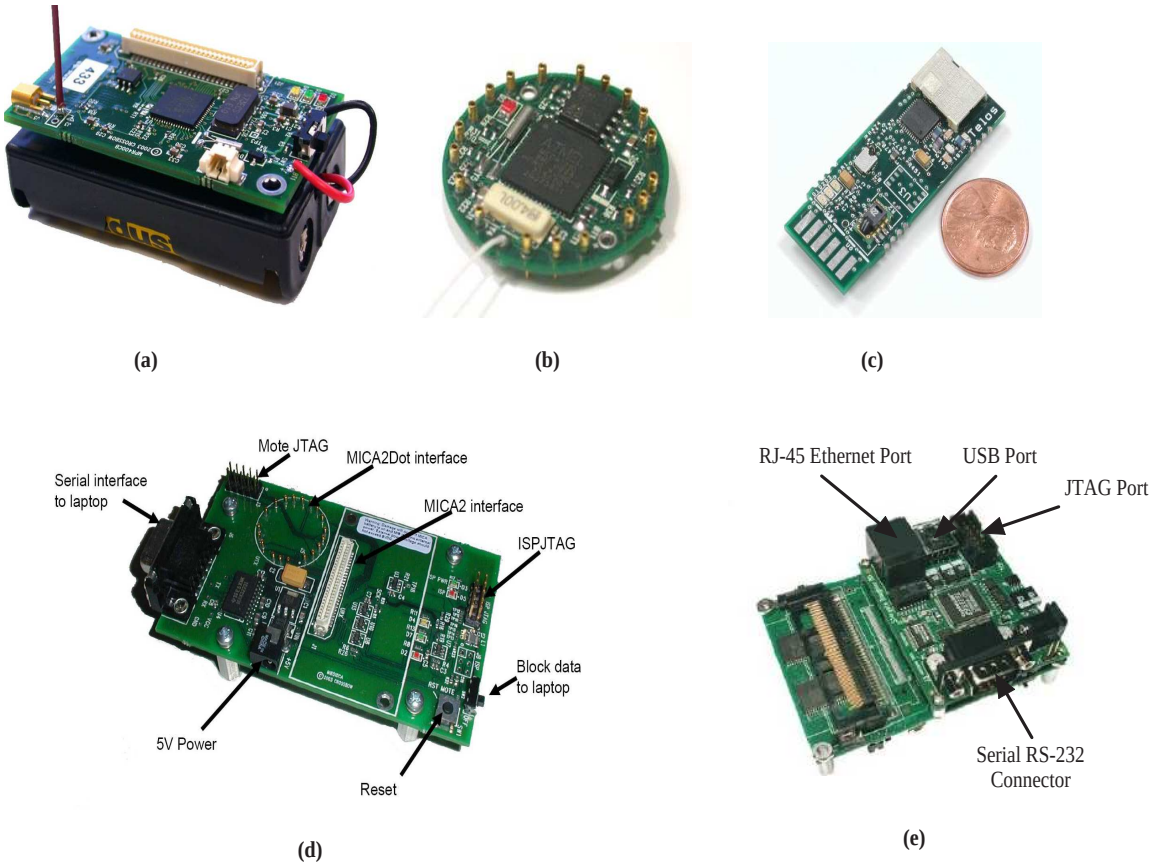

Figure 1.5: (a) Mica2 mote, (b) Mica2Dot, (c) ZigBee-ready, IEEE 802.15.4-compliant TelosB mote, (d) MIB510 Programming Board, (e) Extreme Scale Stargate (XSS)[b]

6. It allows only static memory allocation. In other words, no dynamic heap memory allocation (malloc) and no function pointers are allowed.

Figure 1.6 highlights the difference between the proposed wireless sensor network model (Bulusu, 2002) and the traditional OSI (Open Systems Interconnection) model. The physical layers of both the models are significantly different. In the OSI model the communicating entities do not have any restriction on power availability as they have a constant source of uninterrupted power supply. Even in cases where cables are not possible (for example, wireless communication), recharging or

replenishing power is very manageable. As discussed earlier, this is not possible for an ad hoc randomly deployed large scale sensor network. There are other physical hardware and software differences between a sensor node and a typical personal computer.

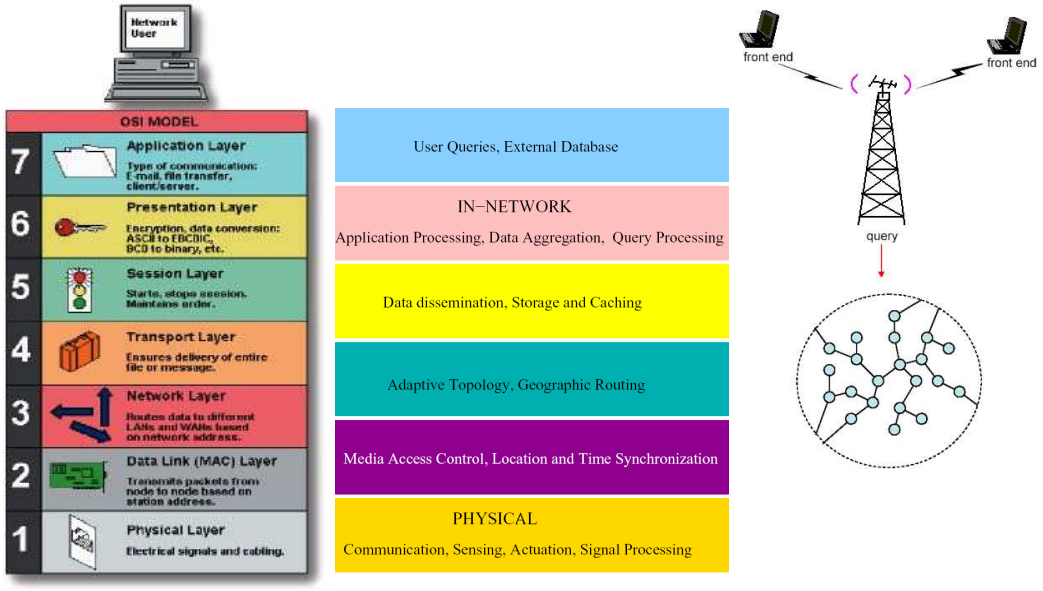

Figure 1.6: (a) Traditional OSI model, (b) Layered communications architecture for sensor networks[c], and (c) Typical sensor network infrastructure.

Figure 1.6(c) shows a sensor network deployed in a large unmanned area. The sensor nodes self-organize to form an ad hoc network to monitor (or sense) target events, gather various sensor readings, manipulate this information, coordinate among each other, and then disseminate the processed information to an interested

[c]The protocols in this stack are minimal and relate fundamentally to sensor networks (Bulusu, 2002). Zigbee (ZigBee) is the name of a specification that standardizes an elaborate protocol stack for distributed embedded devices forming a wireless personal area network (WPAN). The standard consists of a suite of high level communication protocols using small, low-power digital radios based on the IEEE 802.15.4 standard. ZigBee is targeted at RF applications that require a low data rate, long battery life, and secure networking.

data-sink or a remote basestation. This dissemination of information typically occurs over wireless links via other nodes using a multi-hop path (Estrin et al., 1999) (Akyildiz et al., 2002). This inherent distributed and collective communication paradigm for ad hoc wireless sensor networks is significantly different from the traditional point-to-point desktop to server communication model, which is quite centralized in nature. Thus, layers 2, 3, and 4 are modified to support this ad hoc collaborative communication model. These layers are also optimized for energy usage, computational, and space complexity. Additionally, further customization may be warranted by specific sensor network applications that wish to stream the data into a PC-based monitoring application. Layers 5, 6, and 7 of the OSI layer thus need to be modified (Raicu et al., 2002) to bridge this gap between the sensor network platform and the conventional PC (with its underlying 802.11 wireless Ethernet network or its wired 802.3 networks).

1.1.4 Constraints and Challenges

The characteristics of wireless sensor networks need to be understood in great detail in order to develop efficient sensor network protocols that support collaborative network monitoring with increased sensor network lifetime. These characteristics are listed below:

1. *Sensing application requirements*

   Biomedical applications (Schwiebert et al., 2001), for example, a glucose level monitor or a retina prosthesis have special requirements with respect to the sensor hardware and the operation capabilities. First, the sensors must be bio-compatible to avoid any tissue damage. Second, in addition to being fault-tolerant, energy efficient, and scalable, wireless networking solutions should

be ultra-safe and reliable. Also, in biomedical applications, sensors usually sense some biological event and trigger some nerves or tissues in response to control that event or generate another sequence of events. However, there is no explicit feedback between sensors and the external application except for certain periodic maintenance routines. These maintenance routines are essentially an effort to find any faults in the sensor networks implanted in-vivo. Thus, these requirements are application specific and are different for diverse applications.

2. *Security requirements*

   Sensor network applications, such as for biomedical, hazardous environment exploration and military tracking are typical mission critical systems that are highly security sensitive. Unfortunately, sensor networks are vulnerable to all kinds of attacks, such as eavesdrop, jamming, and trojan horses. With constrained available resources, it is impossible to deal with all possible security issues, however, some measures for expected attack must be provided.

3. *In-network processing*

   Wireless sensor networks typically consist of a large number of nodes randomly deployed to sense application specific physical phenomena. Due to bandwidth constraints and the high error rate of wireless links, energy efficient operation of sensor networks requires sensor nodes to relay in-network aggregated sensing events to a remote basestation. Without in-network processing of neighboring correlated sensing events, individual readings of each sensor would have to be sent to the basestation, which is highly impractical under the constraints discussed earlier. Also, by way of multi-hop communication, hop-by-hop reliability becomes more feasible as compared to traditional

end-to-end TCP-based reliability mechanisms.

4. *Data centric processing*

   Data centric processing is an intrinsic characteristic of sensor networks. Sensor data is no longer accessed by ID (or IP, as in the internet). It is more natural to address the data through content, location, or constraints. The IDs of the sensor nodes may not be of any interest to the application. The naming schemes in sensor networks are often data-oriented. For example, an environmental monitoring system requests the temperature readings through queries such as "collect temperature readings in the region bounded by the rectangle (x_1, y_1, x_2, y_2)", instead of queries such as "collect temperature readings from a set of nodes with the sensor net address x, y, and z".

5. *Network sensor platform (hardware and software)*

   Hardware capabilities determine the overall functionality of the sensor device. A sensor node equipped with a GPS can act as a position estimating beacon for other nodes without GPS. Similarly, nodes with higher processing capability and higher battery power can serve as data sinks for their neighboring underprivileged nodes. As mentioned earlier, the operating system determines the real-time capability of concurrently harnessing the sensor hardware to its full potential with low processing, low memory, and low energy.

6. *High unpredictability*

   Sensor network applications are driven by environmental events, such as earthquakes and fire, anywhere anytime following an unpredictable pattern. Sensor node failures are common due to these hostile environments. The radio media shared by densely deployed nodes is subject to heavy congestion and jam-

ming. High bit error rate, low bandwidth, and asymmetric channels make the communication highly unpredictable. Such unpredictability usually prevents off-line design of system parameters. Online monitoring and feedback control are required to provide a certain degree of QoS guarantee under such situations.

7. *Network makeup*

   (a) *Homogeneous or heterogeneous sensor devices*

   The sensor network may consist of specialized nodes having special hardware and software capabilities deployed randomly or deterministically with other low end sensor devices. This heterogenous deployment may be required by certain applications, where placement of the sensor nodes is practical. An example of such an application may be monitoring a high rise building for cracks and other critical hazards or faults.

   (b) *Random or controlled node placement*

   Biomedical sensor networks are examples of stationary wireless sensor networks. In such a network, the placement of sensor nodes is controlled and premeditated. A stationary sensor network normally has little or no mobility. One can also decide in advance the number of neighbors a node may have depending upon application requirements and the position of the sensor (border or internal node) within the deployment. In contrast, a tactical wireless sensor network deployed in a hostile area to track enemy movements in the battlefield is characteristically required to have a random deployment.

   (c) *Redundancy*

The highly unpredictable nature of sensor networks necessitates a high level of redundancy. Nodes are normally deployed with a high degree of connectivity. With such redundancy, the failure of a single node has a negligible impact on the overall capacity of the sensor network. High confidence in data can also be obtained through the aggregation of multiple sensor readings.

(d) *Indoor or outdoor environments*

Sensors deployed for building monitoring fall into the category of indoor environment, whereas sensors deployed to monitor a parking lot facility may be categorized as experiencing an outdoor radio environment. In outdoor environment, there are minimum obstructions, and so the radio signals do not experience as much loss in signal to noise ratio due to reflections and multipath fading. This is not the case for indoor environments, where walls contribute to a drastic reduction in signal strength.

8. *Sensor node or beacon density*

Depending on the application scale, tens of thousands of sensors may be deployed in a very large area. Examples of such an application would be deep space probing and habitat monitoring. Network protocols for collaborative monitoring need to adapt to the topology of the network, its density, and redundancy in order to achieve energy efficiency. Sparser networks may need special treatment to avoid a network partition due to several orphan nodes.

Initialization protocols (or neighbor discovery) for sensor networks usually have one basic operation for position or location estimation. Sensor nodes with pre-location information (for example, nodes with GPS capability) serve as anchors (or beacons) to other nodes that use localized triangulations or

multilaterations to estimate their positions. A higher density of such beacon nodes can significantly reduce localization errors (Bulusu, 2002) and hence the subsequent determination of network parameters that could affect decisions for other primary network operations.

9. *Node mobility*

   There is an important difference between a stationary wireless sensor network and a mobile ad hoc network. Network protocols for MANET are optimized for QoS by optimizing the three important tasks of organization, routing, and mobility management (ORM). For MANET, system performance is attributed mostly to random node mobility rather than to the energy depletions caused by the execution of various network protocols. However, for ad hoc sensor networks, energy depletion is the primary factor in the connectivity degradation and the overall operational lifetime of the network. Therefore, for WSN overall performance becomes highly dependent on the energy efficiency of the algorithm.

   Sensor node mobility makes wireless networking solutions extremely challenging. A mobile sensor network essentially becomes a special research challenge in the field of mobile ad hoc networks (MANETs).

10. *Time synchronization*

    The causality of detection of events is highly dependent upon the synchronization of the clock among the nodes within the sensor network reporting such an event. Also, most of the channel access schemes for providing collision free medium access use a TDMA-based design. This TDMA scheme needs efficient time synchronization among contending neighbors in order to avoid

collisions and hence save precious energy. Depending upon the application, one may need fine-grained synchronization or coarse-grained synchronization. Also, the granularity of the synchronization depends upon the scale of the network and its deployment. Energy efficient distributed protocols are thus needed for time synchronization.

11. *Target event characteristics*

    Sensor networks are deployed to detect and report application specified interesting events. In general, these target events may have spatial, temporal, spatio-temporal, or absolute characteristics. By spatial, we mean that events reported by sensors belonging to a common geographical region are similar and strengthen the fault tolerance level of the report. A moving target is temporal in nature and its detection may follow a predictable path. Depending upon the speed of the target, its characteristics may change from temporal to spatio-temporal. In some cases, the event may be a continuous event, for example detecting the presence of a chemical gas is continuous as it diffuses across the sensor network. On the other hand, intruder detection is a discrete event where the sensor network may detect several intruders or the same intruder at different places and at different times in the network. Thus, the target event characteristics fall broadly into these following categories:

    (a) *Event types*,

    (b) *Event speed*, and

    (c) *Event occurrence rate.*

12. *Sensor Network Models*

    Network simulation used for evaluating the performance of various sensor net-

work protocols may need to take into account various models for the following fundamental sensor parameters:

(a) *Radio propagation and energy models*,

(b) *Mobility model*

(c) *Event traffic model*,

(d) *Sensing coverage and exposure models*, and

(e) *Wireless bit error models*

1.2 Research Motivation

We will now discuss the research motivations based on generic observations on the solutions used to provide efficient sensing and routing services by wireless sensor networks. The following subsections discuss specific solution aspects that have influenced our design philosophy.

1.2.1 Hierarchy, Approximation, Aggregation, and Redundancy

It is very fascinating to realize that over these years with increasing complexity, systems have become hierarchically organized with an approximate aggregation of communication functionalities into several subsystems and components. For example, the internet has become a loosely hierarchical structure of core routers, DNS servers, gateways, and switches. Thus, the network functions via interaction among components that perform specialized tasks suited to their placement in the organization. In other words, the network intelligence is distributed in such a way that every component or subsystem globally provides a limited set of communica-

tion functionality with purely approximate local network information albeit with provisional reliability.

In such a distributed system, redundancy makes up for limited reliability by not only having a number of alternative providers for a certain function but also having an overlap among functions that providers can usually offer. For example, in the internet a layer three router can also act as a switch and viceversa. Besides the differences in physical links and associated protocol or messaging mechanisms, the main difference across communication networks lies usually in the way this network intelligence is distributed and maintained. For example, there could be a centralized system that does the partitioning and assignment of tasks to network entities and also tracks their performance. On the other hand, the network could do this almost instantaneously and on an ad hoc basis. Of course, the latter provides challenges and associated difficulties for maintaining guaranteed levels of quality of service (QoS) among distributed tasks in the face of changing network dynamics.

1.2.2 Task and Roles

An example of a distributed system with centralized control and static assignment of tasks is a hierarchically organized Radio-Access Network (RAN) that provides end-to-end voice and IP-based communication among cell phones in a CDMA based cellular network (see figure 1.7). A RAN consists of a Radio Network Controller (RNC) that communicates with a Radio Node (RN) that acts as a frontend and handles the mobility of the mobile phone. A Packet Data Serving Node (PDSN) acts as a backend access gateway providing simple IP and mobile IP services to mobile phones in an IP network. The PDSN also acts as a client for Authentication, Authorization, and Accounting (AAA) servers so that it can provide services only to

the subscribed mobiles and also differentiate among them based on their subscribed service level agreements (SLAs).

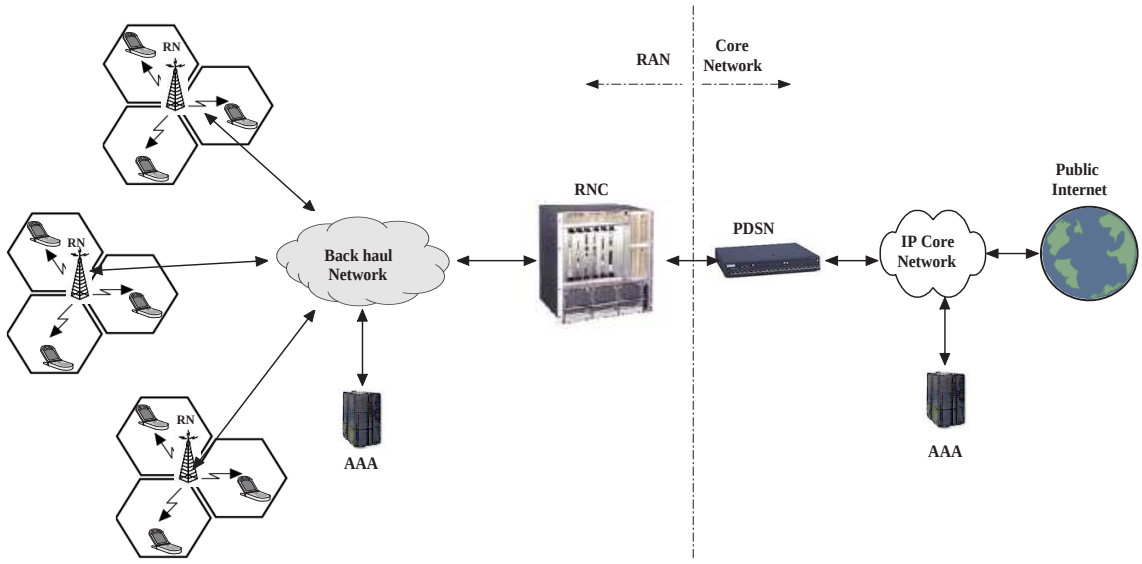

Figure 1.7: Hierarchically organized Radio Access Network (RAN) in CDMA2000

An example of a completely decentralized wireless network is an ad hoc wireless network (see figure 1.8) and almost analogous to the cellular networks is the Mobile Ad hoc Network (MANET). Another example is a Wireless ad hoc Sensor Network (WSN) which is complementary to MANETs with the addition of a sensor and/or an actuator along with a wireless radio, processor, memory, and a battery all integrated into individual nodes deployed to provide sensing and monitoring services. A WSN could be formed by tens to thousands of randomly deployed sensor nodes. The sensor nodes then self-organize into an ad hoc network to monitor (or sense) target events, gather various sensor readings, manipulate this information, coordinate with each other, and then disseminate the processed information to an interested data-sink or a remote base station. This dissemination of information typically occurs

over wireless links via other nodes using a multi-hop path.

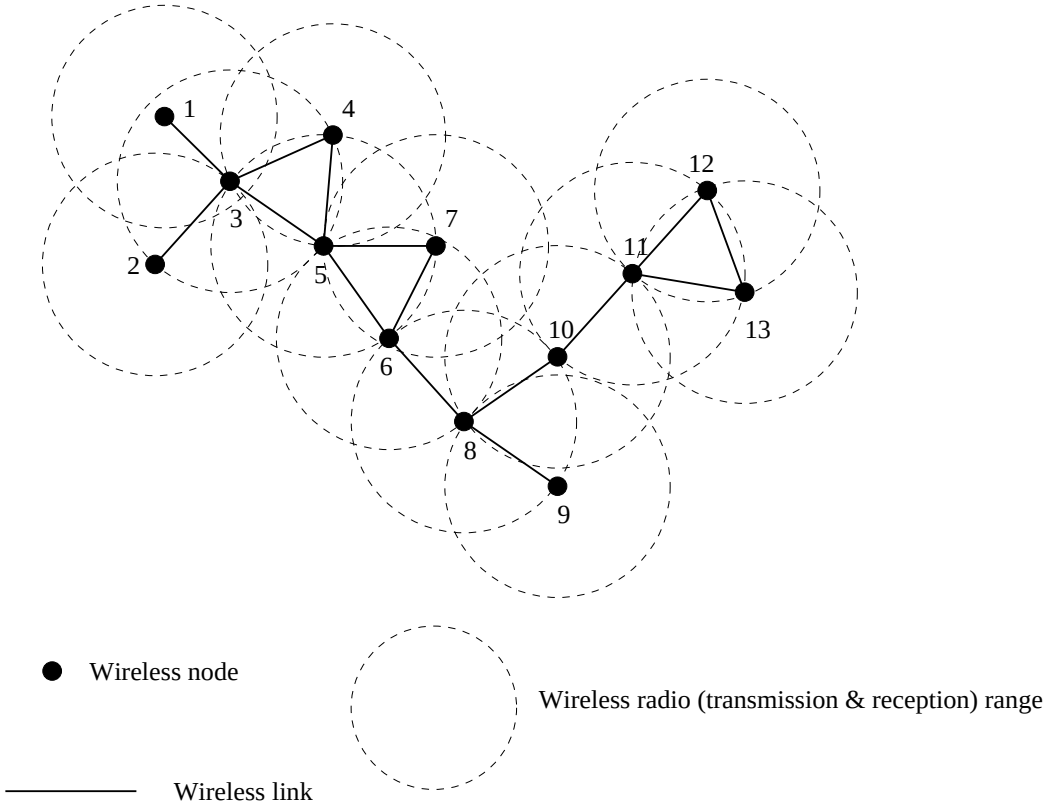

Figure 1.8: Ad hoc Network[d]

In contrast to a RAN, in an ad hoc network, the functions of event (or mobile) tracking and monitoring, handling node mobility, routing, security, performance measurement, and control are all handled collaboratively by participating communication entities or nodes. There is no static assignment of tasks to any particular node. Moreover, this assignment is dynamically taken upon by nodes individually based on the snapshot of the network context perceived locally in their neighborhood. In an ad hoc network, it is very likely that all nodes are equal in terms of their hardware and software capabilities. Thus, as opposed to a RAN, in an ad hoc network, heterogeneity does not play a major role in task assignment except

to break any tie among two equally competing nodes suitable for the same task. For example, if one node has more storage capability than the other and if they are competing for caching routes, then obviously with all parameters being equal, the one with larger spare memory should be selected to act as a router.

With reference to figure 1.8, nodes 4 and 5 can act as routers to route data from 1, 2, and 3 to the rest of the network. So, if both 4 and 5 have equal communication capabilities in terms of route quality to 3 and the rest of the network and they also have equal available battery power, then the only way to distinguish between them is by other related routing requirements, such as route storage. On the other hand, node 8 can cause network partition and eventual loss of communication among its neighbors and the subnetworks associated with them. In this case, we have no choice but to select 8 as the router to forward packets across these partitions. Of course, over time node 8 will die due to energy dissipated by its radio for forwarding packets destined to nodes other than itself[e]. However, fortunately with node mobility (if any) other nodes may come around the vicinity of 8 or 8 may move to some other place. In this scenario, routing will be offloaded from 8 to the best node among these new competing volunteers according to the criteria discussed above. The routing task behaves differently under different local network scenarios or context. These dynamics in the behavior of the task leads to a node and its neighbors acquiring different *roles* at different times in the network.

[d]The radio model assumed in this diagram is very simplistic. Wireless transmission and reception ranges are practically asymmetric and randomly unidirectional based on environmental vagaries, network conditions, and radio propagation characteristics.

[e]The ad hoc network communication paradigm assumes a cooperative model where nodes are not supposed to be selfish when it comes to forwarding packets for others. A form of cost-based incentive mechanism in terms of virtual money (Blazevic et al., 2001) is usually established where volunteers either buy or are rewarded for requested service by sellers or other obligated nodes, respectively.

1.2.3 Role Assignment

The routing service as described requires nodes to collaboratively identify the tasks or subtasks along the route where data gets forwarded. In other words, although node 8 may be aware of routes from several sources to different destinations, it needs other nodes along the path to relay the data to its next hop neighbor. Thus, the routing service needs some nodes to perform the task of route computation and storage whereas it needs others to simply forward along that computed route. For example, source node 1 wants to communicate some information to the destination node 12. There are several routes from node 1 to node 12 and the one that may be selected most importantly depends upon the network knowledge available at each node, and the quality of service (QoS) desired by and granted to source node 1 for routing its packet to destination node 12. Assuming that node 8 is aware of the route to node 12, a request to route from 1 is resolved by 8. Thus 8 acts as a *router*, 1 acts as a *source*, 12 acts as a *receiver* and other nodes along the chosen route from 1 to 12 act as *forwarders*. The task executed by a node within a network at a particular instant of time thus determines its behavior in terms of a "role" that it takes to collaborate among its peers to execute a certain service.

The mapping of tasks to nodes or in other words role-assignment may be based upon the QoS requested by node 1, say shortest path route to destination 12. Of course, it also depends upon the available network resources that nodes are willing to grant for this service. Assuming that the mission of the network is to satisfy QoS by selecting and using the shortest possible route from node 1 to node 12, then it is very likely that nodes along that path will not be available for long to sustain that route. This is due to energy dissipated by the radio. In other words, forwarders along the path will eventually die thus causing not only a network partition but

also a significantly reduced network lifetime. Thus, the assignment of roles has to be dynamic as opposed to being static. We will see how the underlying network organization architecture, by statically assigning roles, limits the flexibility of other protocols above it.

1.2.4 Network organization: a static assignment of roles

Self-organization protocols usually organize a set of randomly deployed sensor nodes into a logical controllable network infrastructure. Besides discovering the links for each node, a self-organization protocol characterizes not only the relative importance of links but also valuates the nodes responsible for managing those links.

A chain-based organization (see figure 1.11) logically organizes the network into a long communication chain. The criteria for a node to be a part of the chain could be high available energy or the shortest path route to a known destination. Nodes that do not satisfy this criteria are not part of the chain. However, they could join the chain if there comes a time when their neighboring chain node fails to pass the selection criteria as discussed earlier. In a tree-based organization, certain nodes are valued as parents whereas other's are valued as children. Links may be characterized as downstream/upstream to differentiate communication between a parent to its children or vice versa (See figure 1.9). Similarly, for a cluster-based organization, nodes with higher energy are valued as clusterhead whereas nodes in the close vicinity of a clusterhead are known as cluster members (See figure 1.10).

This network organization then forms the basis for other protocols to perform their optimizations. In other words, the design of the communication system is essentially from bottom to top, with the network organization architecture acting as an invariant framework upon which protocols dynamically optimize for desired

objectives and handle network variabilities. This bottom to top design that organizes the protocols into layers stacked above each other leads to rigidity where the invariant behavior of one protocol layer along with the invariants of layers below it cumulatively reduces the flexibility of an application to respond to several network scenarios while simultaneously meeting desired performance requirements.

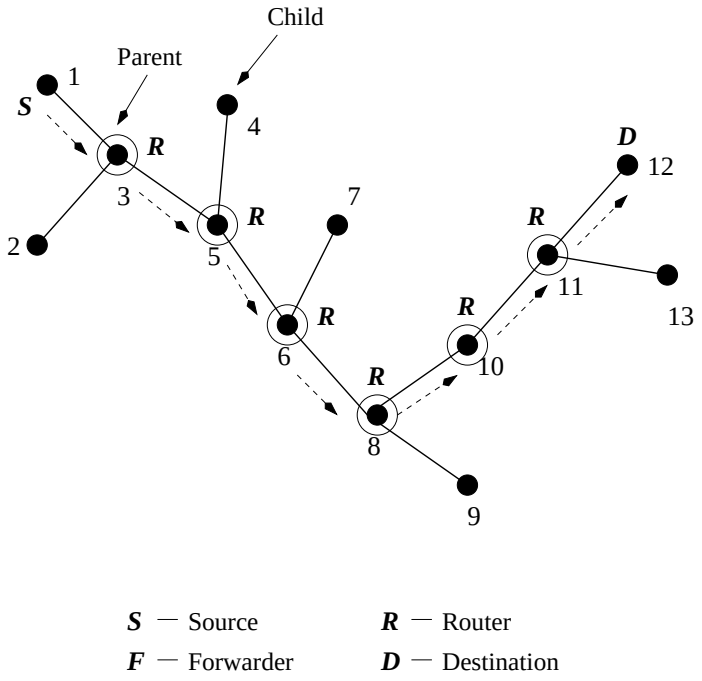

Figure 1.9: Tree organization limits the role assignment of every child to a forwarder and a parent to a router.

An example could be the competition between protocols that are developed for shortest path routing and the protocols that are designed to conserve energy and increase network lifetime. An energy-conservation protocol usually works by adaptively changing the duty-cycle of unused or idle local node resources to save energy. One way would be to identify sensors that are redundant with respect to the area of sensing coverage and then turn these OFF completely to increase the cumulative lifetime of the network. This obviously is inconsistent with protocols

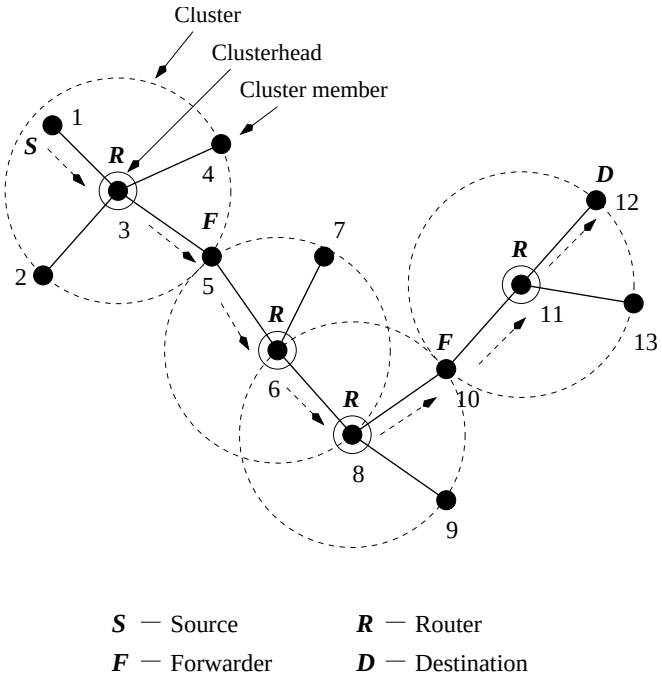

Figure 1.10: Cluster organization limits the role assignment of every cluster member to a forwarder and a clusterhead to a router.

that need these same nodes to be made available to meet specific service objectives such as a shortest route to a destination (as discussed earlier).

Another energy conservation approach would be turn those sensors OFF that are redundant with respect to the established shortest path route to the destination. However, an event tracking protocol that needs high sensing fidelity at various times and at different places within the network may be at odds with this energy saving decision. This is because some sensors that would have been able to detect an event with high sensing degrees have now been turned OFF because they were not assigned as forwarders along the already established shortest path route.

A self-organization protocol may similarly base its organizational decision to logically identify a certain node as a parent (or a clusterhead) and the other as a child (or a cluster member) according to a specific performance metric that conforms

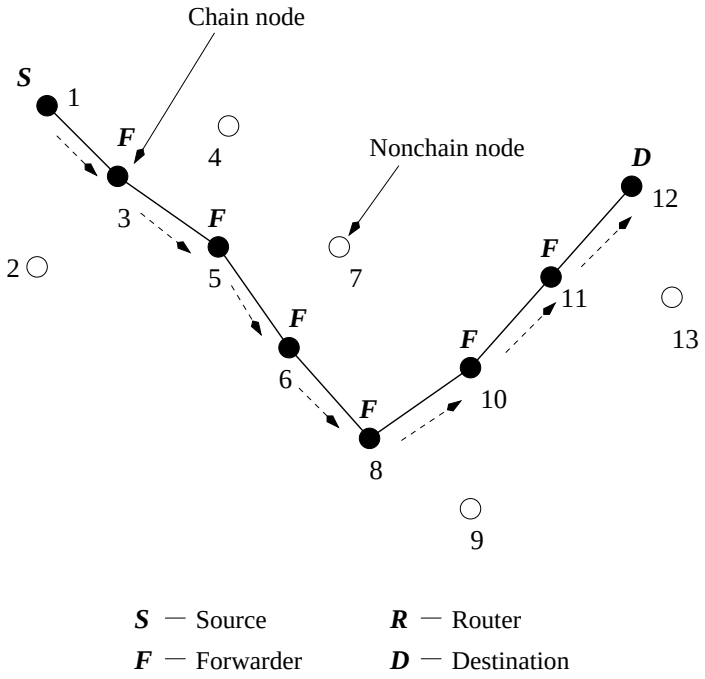

Figure 1.11: Chain organization limits the role assignment of every participating node to a forwarder.

neither to the shortest path routing protocol nor to the high fidelity event tracking protocol. And all of these are definitely at odds with the energy conservation protocol. It is thus obvious that simple logical abstractions such as cluster, tree, or chain lack detailed control flexibility across sensor network resources and protocols.

We therefore need a framework that can serve as a common foundation to express resources in terms of standard units across services, applications, and protocols. All the requirements both from the application in terms of desired QoS and from the network in terms of resource availability and fairness can then be bargained and arbitrated in terms of these units. The strategy that provides the maximum utility, if available, could then be implemented by coordination across protocols running on this common framework. This motivates the need for a resource-based coordination framework across all heterogeneous resources and applications.

1.2.5 Resource-based abstraction framework for coordination

Let us consider the hypothetical sensor network protocol stack shown in figure 1.12(a), where protocols at different layers provide layer-specific services. Also shown are the network functions such as energy management and security that need cross-layer optimization. We represent the problem of multi-protocol coordination for QoS and resource optimizations as a problem in 3 dimensions. As shown in figure 1.12(b), the point(s) where all these objectives are satisfied within the network in a localized and efficient manner is essentially signified as a dilemma representing the nature of the problem (usually NP-complete). However, it motivates the need for an adaptive control framework that promotes cross-layer protocol collaboration to map approximate solutions under established tradeoffs. With this recognition of tradeoffs a mutually agreeable objective could be satisfied if the framework allows this coordination to be modeled as an interplay of resources. This implies that the framework would fall short if it only allows diverging application requirements to converge to clear tradeoffs but makes it difficult to coordinate to achieve this common objective.

An efficient task-based abstraction is needed that allows network protocols to be modeled as a flexible and ordered placement of a collection of tasks. This needs to be complemented by a resource-based abstraction that permits the cost of the tasks to be evaluated in terms of the set of resources locally provisioned around a certain k-hops vicinity. Since use of any resource requires power from the local battery, energy consumption can serve as a common scale for evaluating tasks complexity. Similarly, the use of distributed resources requires imperative inter-task communications and this can also be taken into account by node energy dissipation. A union of these two abstractions results in a framework where nodes collaborate for a service by

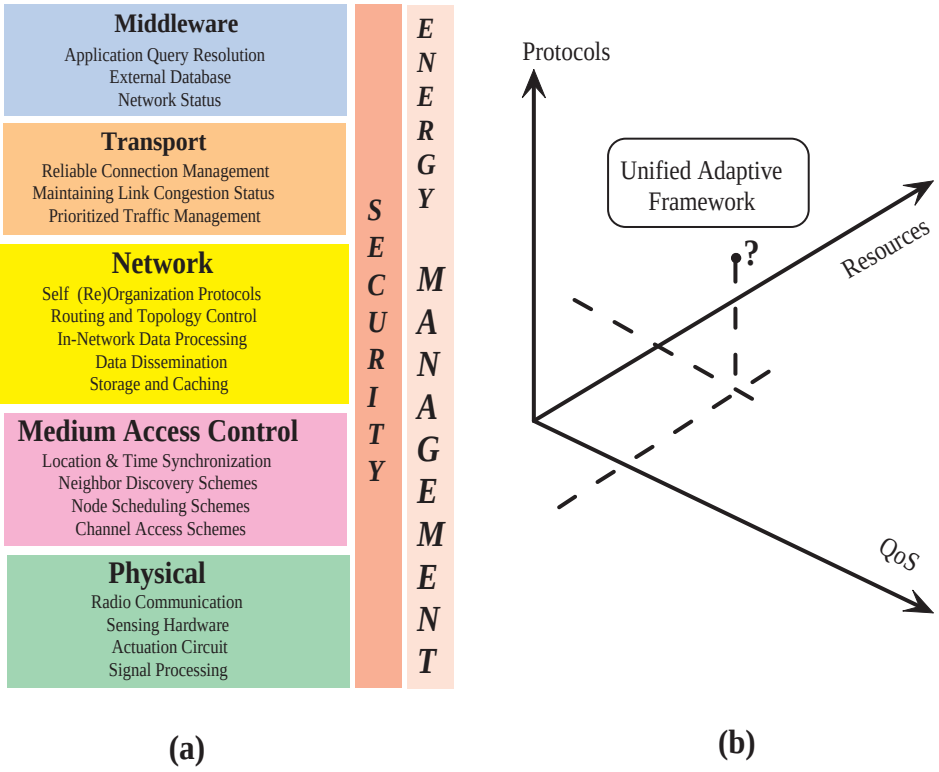

Figure 1.12: Case for a Unified Role Assignment Framework (a) Sensor network protocol stack and (b) Multi-objective {*Resource, QoS, Protocols*} optimization dilemma.

valuating the utility derived from satisfying application desired QoS. This utility is comparable across heterogeneous services, tasks, nodes, and resources. Nodes calculate their utilities based on the energy required to execute the subtasks or roles comprising the service. Depending upon application requirements and resource availability, tasks are (re)assigned to nodes that maximize this utility.

1.2.6 Rescue Mission: Uncertainty and Chaos

One of the interesting examples that illustrates most of the role mapping issues under cross layer design and multiple application scenarios is the deployment of

sensors in an industrial facility. Consider a heterogeneous deployment of sensors in and around a chemical plant that manufactures hazardous chemicals (see figure 1.13). The network of deployed sensors facilitates rescue missions during fire, chemical spill, and emergency medical scenarios. The rescuers have different missions in each of these scenarios and their applications may demand different services at varying QoS requirements. During disasters it may be impossible to have a centralized arbitration scheme to avoid contention for resources among applications through prioritized access to the sensor network resources.

In the example, we see several applications demanding services from the sensor network in real time. The events that are driving these applications are also very dynamic. Failure of nodes is expected in such a drastic and challenging rescue scenario. Although during the passive state sensors at various plant domains formed an initial organization, the current volatile situation demands changes in protocol functionality. Network and application dynamics dictate these changes in protocol behavior in real time. With the flexible abstraction features supported by the generic role-based framework, it becomes easy for applications to perform such adaptation and evolution for activities like organization, event tracking, collecting network status, deploying new nodes, etc. In fact the possibilities evolve as the role-based middleware unambiguously maps dynamic application configurations by way of elementary tasks and required node resource capabilities.

From this chemical spill example, we have identified the following fundamental role types: (1) Router or Backbone role; (2) Sensing coordinator role; (3) Sensing collaborator role; (4) Gateway or Bordercast role; (5) Beacon or Anchor role; (6) Shadow role; (7) Follower role; (8) Reserve role; (9) Source or Sink role; and (10) Sensing-region role. Some roles, as their names suggest, provide services that are

self-explanatory. However, roles like the *reserve* role, *shadow* role, and *follower* role need further explanation. Since energy efficiency and network lifetime are the main concerns in sensor networks, many energy-conservation protocols have been proposed that turn off sensors that are redundant from several aspects including network sensing and communication coverage. These *sleeping* sensor nodes act as *reserves* for future activities. These *reserve* roles become important as multiple applications simultaneously demand services from the sensor network. *Shadow* roles are required in situations where a single sensor or a relatively small group of sensors holds vital state information required for the stable functioning of sensing regions that are either static or migrate as events move around the sensor network. These roles are also important for data storage and caching as they act as replication storage facilities for critical state information. For events that migrate across the sensor network, e.g. a moving enemy tank, the sensing region that is formed for detecting, aggregating, and relaying sensor readings also needs to be mobile. Group sensing mobility differs from individual node mobility that is typical in mobile ad hoc networks (MANETs). Instead, sensing group (re)organization has to be repeated at every detection point of the migrating event so consistent real-time reporting of event characteristics is possible. Protocols that form such migrating sensing groups in-sync with mobile events belong to a special class of event detection and tracking protocols for wireless sensor networks. Typically these event tracking protocols work toward a pro-active selection of *follower* nodes that in the near future organize a sensing region per their prediction modeling of event dynamics (direction and speed).

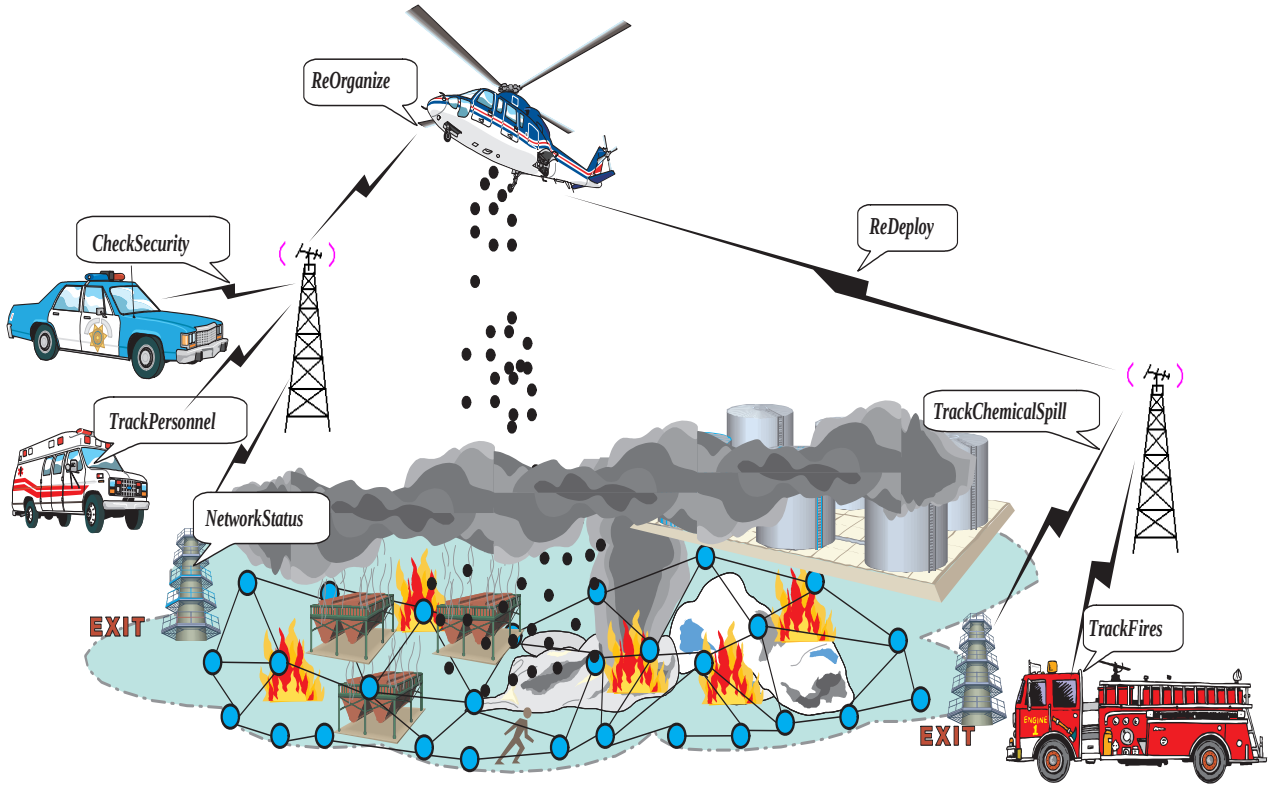

Figure 1.13: Example cross application service to role mapping scenario

1.3 Research Contributions

In this dissertation, our main goal is to generalize specific sensing and networking characteristics and use this to develop a unified resource abstraction mechanism for localized and collaborative resource arbitration, coordination, and control. We pursue this by first identifying common sensing and networking characteristics among competing application-specific protocol solutions. We abstract these performance characteristics into a set of networking and sensing metrics. To that effect, we have developed a set of sensing metrics that serve as a framework for our sensing Quality of Service (or $sQoS$). This agenda is then realized in the following components, which together form the research contribution of the thesis.

1.3.1 Role-based Hiearachical Self-Organization (RBHSO)

We use sensing and networking metrics to understand the flexibility of the underlying network organization to execute services as an allocation of specific tasks collaboratively executed as local network roles. To that end, we have developed a Role-based Hierarchical Self-Organization (RBSHO) protocol as an initial architecture for self-organization for wireless sensor networks (Kochhal et al., 2003). We study the flexibility of our organization with a popular cluster-based protocol (LEACH).

1.3.2 Unified Role-Abstraction Framework (URAF)

We extend our RBSHO algorithm to develop a Unified Role-Assignment Framework (URAF) to model application services as roles played by local in-network sensor nodes with sensor capabilities used as rules for role identification. The URAF design philosophy incorporates concepts of aggregation, hierarchy, approximation, and redundancy to develop generic roles for elementary tasks and use these to compose complex roles that abstract protocol specific interactions among nodes. We abstract these complex protocol task interactions in the spatial and temporal domain by way of role-coordination graphs that highlight this need for dependency and subsequent coordination among roles. Using an example domain-specific empirical model for energy consumption, we profile the use of various node resources such as radio, sensing, battery, memory, and computation for both elementary and complex roles. This role-service load profiling in terms of energy allows nodes to pursue application load balancing by way of an adaptive assignment and scheduling of roles to nodes. For example, with declining network resources roles become more adaptive as they evolve toward less energetic types. In this scenario, a node

attempts to extend its remaining lifetime by arbitrating services to a much larger pool of neighbors. This facilitates reconfiguration and the subsequent reassignment of complex roles to a number of simpler roles.

The URAF can be implemented as a Role-based Middleware (RBMW) that allows applications to quantify service requirements and tradeoffs either as a simple or as weighted sum of utilities and points in QoS space. The RBMW then maps these requirements in terms of a specific assignment of roles to nodes. To limit the problem space, we concentrate on energy-efficient role-assignment for data aggregation services in a heterogeneous sensor network. The RBMW incorporates load balancing protocols across roles, nodes, and services in terms of pairwise neighborhood role-exchange, role-mergers, and role-redirection. The RBMW behavior is dictated by its underlying role state machine that considers several scenarios for roles such as message arrival, sensing events, and neighboring role-context changes. The state machine also allows roles at a higher level to pursue execution scheduling of local and neighboring roles among nodes in the network. The URAF assumes the availability of protocols to provide and share cross-layer network information among nodes to pursue role assignment, scheduling, and load balancing.

1.4 Dissertation Outline

The rest of this dissertation is organized as follows:

Chapter 2 outlines the related work in terms of cross-layer optimizations achieved by protocols at various layers to meet desired QoS requirements. We will also discuss several network organization protocols along with our proposed role-based hierarchical self-organization protocol that forms the basis for our role-based framework. We understand the concept of tasks and roles from various perspectives including

sensor networks, multi-agent systems (MAS) and its challenging application toward robots playing soccer. We also analyze existing heuristics which formulate the role-assignment (RA) for data-aggregation as a capacity-based facility location problem (CFLP). We also analyze RA algorithms that recursively minimize for a set of metrics using the concept of local domination and elimination among a connected set of nodes. We also consider analogous algorithms based on micro-economic theories such as Game Theory and Mechanism design that have been applied to sensor networks where the utility is to minimize energy consumption. We also consider existing software abstraction frameworks and middleware for wireless sensor networks. Most of these abstractions are either application specific or service specific.

Chapter 3 focuses on the design philosophy of our Role Based Hierarchical Self-Organization (RBHSO) protocol that forms a hierarchical connected dominating set (CDS) network organization for wireless sensor networks. In this network hierarchy, we also assign specific roles (or tasks) to sensors based on their physical wireless connectivity and sensing characteristics. The chapter provides an in-depth analysis of the prominent characteristics of the sensing phenomena. These sensing concepts forms the basis of our proposed sensing based metrics, such as the sensing proximity value (SPV), the cumulative sensing proximity value (CSPV) and cumulative sensing degree (CSD). We have also discussed relevant connected dominating set (CDS) concepts. We use both the sensing and CDS-based network metrics to form a self-organized sensor network that establishes a network-wide infrastructure consisting of a hierarchy of backbone nodes, and sensing zones that include sensor coordinators and sensing collaborators (or sensing zone members). We demonstrate the effectiveness of our design through theoretical analysis and simulation.

Chapter 4 considers the design of our unified role-based assignment framework

(URAF) that serves as a common platform to map multiple applications and services across heterogeneous sensor network deployment. It implements the concepts of hierarchy, aggregation, redundancy, and approximation to abstract an application service in terms of a hierarchical organization of roles assigned to neighbors such that their collaborative execution meets the desired QoS. The QoS is modeled in terms of utilities that uses two domain specific models such as the role-energy model and the concave role-resource service utility model to measure the use of resources by roles over time in terms of energy. Using these two models, a specific configuration of roles could be considered to meet the desired energy and time requirements. If the application desired requirements cannot be met then the framework proposes to meet the QoS that is currently possible. With this, the framework allows the middleware to pursue two objectives, translate the desired requirements to the number of in-network roles needed and the energy expended by their execution for the desired service time. The number of roles that meets the agreed upon QoS between the application and the network is then used as input by the role-assignment algorithms (RA). We also discuss specific role properties such as role-coordination graph, role failures monitoring and repair, role-load balancing, role-state machine, and role-execution scheduling.

We wrap up our discussion in chapter 5 by giving an overview of the accomplishments of our work, and providing a glimpse of our future work and direction.

Chapter 2 – Related Work

In this chapter, we discuss several research solutions related to network self-organization and sensor network abstractions for ad hoc wireless sensor networks. These research solutions will be discussed considering the following prevalent sensing and wireless networking aspects:

- WSNs are ad hoc and nodes have heterogeneous capabilities,
- Most of the nodes are battery powered,
- Services are requested by an end user. These services usually relate to an event of interest being detected, monitored, tracked, and communicated by sensors within the network to the user.
- The QoS desired for a particular service includes not only efficient sensing requirements but also energy conservation such that the lifetime of the network is not deprecated significantly.

In regards to these aspects, several research challenges need to be addressed and the solutions incorporate optimization at a specific layer in the protocol stack. However, since energy conservation is also important, any layer-specific optimization needs to incorporate trade offs between competing performance requirements. Such tradeoffs usually involve maintaining, sharing, and adaptively controlling specific protocol parameters across layers in the protocol stack. This is referred to in the research literature as cross-layer optimization. This adaptive cross-layer opti-

mization is generalized into specific network abstractions such that they provide control interfaces for higher layers.

A further generalization of network abstractions is possible by way of a middleware that accepts service specifications and QoS requirements and translates it into appropriate control to the network abstraction at a lower level. Sensor programming languages provide generic programming constructs that allow flexible programming with network abstractions for application programmers to implement a generic middleware. In order to generalize across different sensor network platforms (both hardware and software), a middleware can also be replaced by a virtual machine. Figure 2.1 highlights these concepts at a high level in terms of sensor network middleware that incorporates specific network abstractions. These abstractions in turn incorporate shared cross layer information to map applications across heterogeneous sensor network platforms.

This chapter is organized as follows. The first section discusses several self-organization solutions with particular concentration on topics such as hierarchical self-organization, network metrics used to elicit such an organization, the message complexity of the algorithm, and the evaluation of the organized network in terms of several architectural metrics. We compare and contrast these solutions with our role-based hierarchical Self-Organization (RBHSO) approach. The next section discusses in brief several cross-layer techniques that consider both energy-efficiency and sensing performance together. This section will provide several generic insights into strategies used to operate an ad hoc wireless sensor network in a heterogeneous deployment and under energy constraints. The third section discusses specific network abstractions that use these generic strategies and provide a software framework. The fourth section discusses the generic application of roles in distributed

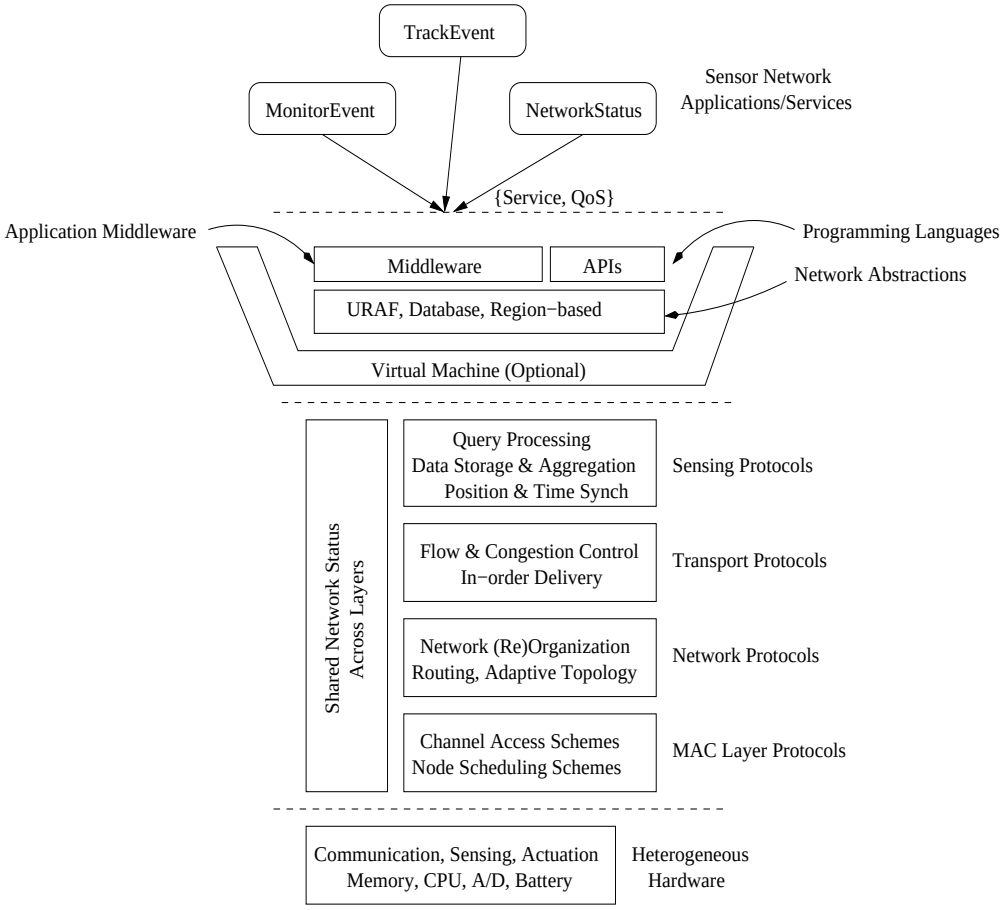

Figure 2.1: Middleware, Network Abstractions, and Protocol Layering.

systems with specific emphasis on role-assignment techniques for wireless sensor networks. We finally conclude by summarizing the existing work and highlighting future challenges in the area of sensor network self-organization, sensor network abstraction frameworks, and cross-layer approaches.

2.1 Sensor Network Organization Protocols

Self-organization in distributed systems is a natural outcome of specific interaction among network components on a microscopic level such that their emergent

global behavior at the macroscopic level yields desired application performance. Sometimes it is not possible to predict in advance any unintended emergent behaviors that may be undesirable. Similarly, sometimes changing network dynamics may make some previously desired behavior suboptimal and hence may dictate appropriate and suitable adaptations to a new required emergent behavior. To minimize or eliminate such undesired effects, interacting protocols or components may be guided by specific design strategies that map as local rules intended to yield desired global behaviors. The initial self-organization protocol that forms the basic organizational structure for WSNs may thus need further adaptation by the following network functions that run on top of this structure:

1. Communication scheduling,
2. Topology control for forming structures,
3. Time synchronization,
4. Data dissemination and aggregation,
5. Task organization and placement,
6. Software configuration and updates,
7. Energy conservation,
8. Fault detection and repair, and
9. Security and resistance against malicious attacks.

Some of these network functions are generic (e.g. routing), some are application dependent (e.g. data aggregation and query dissemination), some are context

dependent (e.g. energy-aware task assignment), and some are dependent upon the environment (e.g. event sensing and actuation). The dynamic nature of sensor networks makes it difficult to support a priori design of optimal behaviors to implement these network functions. Thus, there is a need to investigate self-organizing techniques that could enable a network to shape its own behaviors to adapt and evolve to changing network dynamics and application requirements.

From a layered perspective, it can be observed that these specific network functions also reflect a hierarchy of abstractions. The initial field deployment results in a fundamental abstraction in terms of available physical resources. The subsequent network self-organization then results in formation of a collective that allows these distributed resources to be managed and shared as common resources. This initial organization serves as the reference for other functions to shape this collective behavior to support additional levels of abstraction in terms of topology, energy, tasks, events or data, and faults.

The problem of self organization (or self configuration) has been a hot topic of research in wireless ad hoc networks including mobile and stationary sensor networks. Self organization involves abstracting the communicating entities into an easily controllable network infrastructure. Cluster or connected dominating set (CDS), tree, grid, or mesh based organizations are typical. An excellent discussion of various algorithms supporting cluster-based organizations is furnished in (Steenstrup et al., 2000). A much earlier survey of self-organization techniques in 1986 for wireless networks has been furnished by Robertazzi and Sarachik (Robertazzi and Sarachik, 1986). However, this survey considers solutions to resolve basic issues related to the physical characteristics and limitations of wireless networks instead of issues that arise from the pervasive and large scale ad hoc deployment of WSNs.

In mobile ad hoc networks, self organization essentially involves maintaining some form of network organization to support routing infrastructure in the presence of random uncontrollable node mobility. Some relevant research in this area include the ZRP protocol (Haas et al., 2002), and the terminodes protocol (Blazevic et al., 2001). For mobility management, ZRP uses zones that are similar to clusters whereas the terminodes protocol uses the concept of self organized virtual regions. Routing in both these approaches involves two different schemes, a proactive routing scheme for nodes within a local virtual-region or zone, and a reactive scheme for nodes located in remote virtual-regions or zones. Since in mobile-ad hoc networks the availability of the network is dependent on each user's discretion, an incentive for cooperation by way of virtual money called nuglets is employed in terminodes.

Sohrabi, Pottie, *et al.* (Sohrabi et al., 2000) (Sohrabi and Pottie, 1999) (Clare et al., 1999) have introduced in detail the problem of self organization in wireless sensor networks. They point out the differences in various related wireless network models (e.g.

MANET, Cellular networks, Bluetooth, and HomeRF) and the WSN with respect to the desired network performance objectives. (Clare et al., 1999) gives a detailed description of the top-level design components of a self organization protocol for WSN. (Sohrabi and Pottie, 1999) (Sohrabi et al., 2000) propose a self organization protocol for WSN forms a flat topology as opposed to a hierarchical organization. The self organizing algorithm includes a suite of protocols designed to meet the various phases of network self organization. There is one protocol (SMACS) that forms a joint TDMA-like schedule (similar to LCA (Baker and Ephremides, 1981)) for the initial neighbor-discovery phase and the channel-assignment phase. Other protocols (like EAR, SAR, SWE, and MWE) take care of mobility management, multi-hop

routing, and the necessary signaling and data-transfer tasks in local cooperative information processing.

Subramanian and Katz (Subramanian and Katz, 2000) propose a self configuration architecture that leads to a hierarchical network with address autoconfiguration and a number of other useful properties. Their self organizing algorithm lists four phases of operation. These are the discovery phase, organizational phase, maintenance phase, and self reorganization phase. Chevallay *et al.* (Chevallay et al., 2002) build on this architecture by proposing a hierarchical cluster-based organization of a network of wireless sensors. The clusterhead election is based primarily on the energy level and processing capability of each sensor node.

Mirkovic *et al.* (Mirkovic et al., 2001) organize a large-scale sensor network by maintaining a dynamic multicast tree-based forwarding hierarchy that allows multiple sinks to obtain data from a (sensor) source. Their algorithm does not need a globally unique ID for every participating sensor node. Thus address autoconfiguration is not one of their self organization objectives as it is for (Subramanian and Katz, 2000) and (Chevallay et al., 2002). The RBHSO algorithm assumes the existence of a globally unique ID for each sensor node.

Krishnan and Starobinski (Krishnan and Starobinski, 2003) present two algorithms that produce clusters of bounded size and low diameter by having nodes allocate local *growth budgets* to neighbors. Unlike the expanding ring approach (Ramamoorthy et al., 1987), their algorithms do not involve the initiator (or clusterhead) in each round and do not violate the specified upper bound on the cluster size at any time, thus having a low message overhead as compared to (Ramamoorthy et al., 1987). The RBHSO protocol uses localized communication among neighbors during the self organization phase. In order to limit the membership of the sensing

zones as well as the number of sensing zones, our algorithm uses two specified minimum and maximum sensing zone membership limits. However, in the final stages of the algorithm, orphan nodes will join any nearest neighboring sensor coordinator or a sensing zone member. This is done to cover the maximum possible number of nodes in the organized hierarchy.

Ni *et al.* (Ni et al., 1999) consider the clustered broadcast protocol where they emphasize the importance of highly uniform clustering with low overlap. In this protocol, the broadcast message is relayed from cluster-head to cluster-head, which then broadcast the message to their followers. In a clustering with few clusterheads and large cluster sizes, the clusters have minimal overlap and provide the best coverage of the network with the fewest clusters. Hence, the number of repeated broadcast transmissions over any area will be small, thus reducing the amount of transmission collisions and channel contention, allowing communications to become faster, more efficient and more reliable.

On the other hand, a poor clustering with much cluster overlap and many cluster-heads loses much of the benefits of clustering as transmissions will be repeated in areas of overlap with significant channel contention. This has repercussions for other efficient protocols that rely on having a network partitioned into clusters of uniform size. Some examples of these protocols include routing protocols (Krishna et al., 1997) (Thaler and Ravishankar, 1998), protocols for reliable broadcast (Ni et al., 1999) (Pagani and Rossi, 1997), data aggregation (Heinzelman et al., 2000) (Xu et al., 2003), and query processing (Estrin et al., 1999).

Chan and Perrig propose an emergent clustering algorithm known as ACE (Chan and Perrig, 2004), that results in highly uniform cluster formation that can achieve a packing efficiency close to hexagonal tiling. By using the self-organizing properties

of three rounds of feedback between nodes, the algorithm induces the emergent formation of clusters that are an efficient cover of the network, with significantly less overlap than the clusters formed by existing algorithms. The algorithm is scale-independent i.e. it completes in time proportional to the deployment density of the nodes regardless of the overall number of nodes in the network. ACE requires no knowledge of geographic location and requires only a small constant communication overhead. ACE is an example of the power and flexibility of emergent algorithms (defined in (Fisher and Lipson, 1999)) in large-scale distributed systems.

In the case of a random sensor deployment scenario, there is essentially a dichotomous scenario where on one hand, we need almost total independence between sensing zones, and on the other we also want to reliably track moving events among neighboring clusters of sensors. Our RBHSO algorithm is a localized algorithm that requires a constant number of three iterative rounds of message exchanges among neighbors. Our algorithm selects sensor coordinators or clusterheads deterministically, with shorter average distances between sensing zone or cluster members and the sensor coordinator or the clusterhead. Similar to ACE, our algorithm tries to minimize overlap between neighboring sensing zones. We feel that the overlap among neighboring sensing zones actually reflects the dependency for events occurring at the border of the sensing zones. This dependency among neighboring sensing zones can be effectively used to detect and track moving targets and communicate it effectively across clusters or sensing zones. However, the desired amount of sensing dependency is still an open issue and is application dependent. To minimize inter-cluster communication and avoid the clusterhead becoming a communication hotspot, nodes that are at the border of 1 or more clusters can be assigned the role of a gateway or bordercast sensor role where they gossip this information across

clusters with minimal overhead.

(Olariu et al., 2004) develop a lightweight clustering protocol that organizes a large number of sensor nodes into a multi-hop, collision free and adaptive communication infrastructure. After the infrastructure is constructed, the protocol pursues energy efficiency by letting sensors sleep and wake up randomly in the leader election phase. The sensors that wake up the earliest are elected to be cluster leaders. In this way, the leaders are allowed to be elected evenly in the area. Their network infrastructure isolates clusters to reduce power interference by assigning different frequency channels to neighboring clusters. The protocol is complemented by a collision resolution mechanism that avoids power interference between neighboring leaders by having them send neighbor discovery beacons at different times. In the cluster, the leader schedules the routine transmission and reception of events. Unpredictable events are handled by the wake up mechanism. The role-energy model that complements our RBHSO protocol supports energy management at the granularity of the tasks executed by roles and the resources it consumes. Our approach evens the energy consumption of the sensing zones by limiting its membership and also including only those sensors that contribute significantly to the sensing quality of any event in its zone or group. However, our RBHSO algorithm assumes the existence of a MAC protocol to resolve collisions among neighboring transmissions. Our algorithm does not necessarily warrant the use of separate frequency channels to roles as is needed in the clustering protocol. Thus RBHSO protocol avoids the need for complex radio hardware in sensors.

Self-organization algorithms usually converge to a desired structure or organization within a certain number of rounds of local communication among its neighbors. The convergence of most of the algorithms are bound by certain performance pa-

rameters such as node degree, sensing coverage or exposure, cluster size, number of cluster heads, average inter cluster and intra cluster distance, remaining energy or number of communication rounds among neighbors, etc. The best scenario is to have the self-organization algorithm converge to a critical equilibrium where it meets the desired requirements. Sometimes this equilibrium may be of a fragile nature and may itself need to be bounded by a upper and lower margin around the equilibrium point or the threshold.

Krishnamachari *et al.* (Krishnamachari et al., 2003) reported phase transitions in wireless networks, identifying a critical threshold of node density that leads to global connectivity. Below the threshold a network will not connect whereas above the threshold a network generates interference and wastes energy. Krishnamachari suggests that phase-transition analysis could help to select design parameters that enable a self-organizing wireless network to reach a desirable operating point. However, such a study needs to not only account for the conditions that existed during the initial organization but also incorporate changing conditions that can dynamically disturb equilibrium and induce periods of instability, or drive a system into oscillation or chaos. Self-reorganization and maintenance protocols need to be able to forecast and analyze these conditions so that complementary mechanisms could be developed that resist and deal with these conditions effectively.

Mills (Mills, 2007) provides a detailed survey of similar aspects of self-organization protocols for other higher layer protocols that optimize for energy while simultaneously: sharing processing and communication capacity in terms of data gathering (Xu et al., 2006) and query dissemination (Wang et al., 2004), synchronizing time (Werner-Allen et al., 2005) (Ganeriwal et al., 2003) (Hong et al., 2004); configuring software components; adapting behavior associated with routing (Braginsky

and Estrin, 2002), with disseminating (Intanagonwiwat et al., 2000) and querying for information, and resisting attacks (Boonthum et al., 2006) (Ye et al., 2004) (Yu and Liu, 2005) (Jamshaid and Schwiebert, 2004). The objective of these research proposals clearly fall outside the scope of our dissertation. However, the underlying network architecture optimized by these protocols for a specific service are the standard cluster-based designs or a tree based architecture. We discuss the relationship between these basic network architectures and our RBHSO protocol in the next chapter where we highlight its design philosophy.

Our Role-Based Hierarchical Self-Organization (RBHSO) protocol identifies the need for organizing a sensor network according to the tasks appropriate for each sensor node based on their initial deployment in the network. Past research in group-based (or hierarchical) sensor networks have ignored the possibility of utilizing both the physical communication and sensing characteristics to assign roles to sensor nodes. First, this may be partly due to the assumption that such a hierarchical organization may be too static (or rigid) to be reorganized with respect to the ultimate traffic pattern that may run on top of this self-organized network architecture. Second, concentrating specific responsibilities on specific nodes could result in such nodes becoming easy targets for faults, thus making such a hierarchical network inherently less fault tolerant. However, with sufficient network density, both of these problems can be resolved efficiently by systematically rotating roles among neighboring nodes in a localized manner without much overhead. We develop this role-rotation concept to pursue load balancing among neighbors. In this regard, we have developed several role assignment algorithms that are a part of our Unified Role Assignment Framework (URAF).

2.2 Cross Layer Approaches in Sensor Network Design

In this section, we discuss several existing application specific cross-layer protocol designs and optimizations that jointly optimize and trade off specific performance metrics over others in a balanced way by exploiting the collaborative nature of WSNs and its correlation characteristics across layers. In general, cross-layer approaches are motivated by the following three reasons (Melodia et al., 2005): (1) the stringent resource capabilities of sensor nodes, (2) the significant overhead of layered protocols, and (3) the need for application-aware coupling of specific low power optimizations at relevant layers with the event-centric communication protocols.

Recent research on cross-layer optimization techniques (Song and Hatzinakos, 2007) (Rowe et al., 2008) (Royo et al., 2007) (Fang and McDonald, 2004) (van Hoesel et al., 2004) (Vuran et al., 2005) have identified specific joint optimization across protocol layers that are especially important for an energy efficient and application specific design of sensing and communication protocols for WSNs. Thus, for example a cross-layer optimization to select a best possible node for a particular role (say, a sensing coordinator) in the network may include exploiting the following observations among neighboring sensors: spatiotemporal correlation of sensor readings, node heterogeneity at the hardware level, along with other environmental characteristics such as wireless channel conditions, link quality, sensing coverage, and network redundancy. In other words, the communication protocols devised for WSNs that focus on cross-layer design techniques usually result in either the complete fusion or replacement or modification of several specific network layers in the classical open system interconnection (OSI) network stack. With this there exists several possibilities of disruptions in the protocol layer abstractions, and hence precaution (Kawadia and Kumar, 2004) is advised with the cross-layer techniques as

it may decrease the level of software modularity across well defined interfaces and between software processes such as design and development. This causes difficulty in further design improvements and innovations as it increases the risk of instability caused by unintended functional dependencies, which are not easily visible in a non-layered architecture.

Since a wireless sensor network is a synthesis of several technologies across domains such as wireless, sensing, battery, and low power operations, it becomes necessary to design and validate models for all these technologies for practical applications in a heterogeneous deployment. Accordingly, the broadcast and asymmetric nature of the wireless channel for simple communication primitives such as flooding is investigated through testbed experiments in (Ganesan et al., 2002). It is observed that the performance predicted using unit disk graph models for flooding are not realistic as they are simplistic and do not account for the effect of wireless channels. Similarly, in (Zuniga and Krishnamachari, 2004), it is found that the radio models that assume a perfect reception within circular radio range coverage can be misleading in the performance evaluations of communication protocols as they ignore the existence of transitional regions in low power links. With this non conformity between theoretical communication models and practical observations, guidelines for physical-layer-driven protocol and algorithm design are investigated in (Shih et al., 2001). In general, research studies that identify such issues strongly advocate that the communication protocols for WSN be re-evaluated using practical models and through experiments to account for the wireless channel properties.

Another popular cross-layer design for efficient protocols considers interdependency across layers. For example, interdependency between local contention and end-to-end congestion call for adaptive cross layer mechanisms for efficient data

delivery in (Vuran et al., 2005). Similarly, the interdependency of sensing observations across time and space also known as spatiotemporal correlation is another significant characteristic of sensor networks that can be exploited for further energy savings in WSNs (Kochhal et al., 2004) (Vuran et al., 2004).

A simpler approach to cross-layering is to develop protocols that simply share information across layers for better decisions. In this regard, most of the energy-aware sensor network protocols (both for communication and sensing) assume the availability of contextual information maintained vertically across layers as shown in figure 2.1. X-lisa (Merlin and Heinzelman, 2006) proposes such an information sharing protocol architecture for sensor networks that can support existing protocols while simultaneously providing a platform for advanced cross-layer improvements. This architecture supports different services and data structures for providing information that can be shared among all layers of the protocol stack for increased network performance. As an example, the authors implement X-lisa, a network-aware adaptation of the channel probing MAC protocols that reduce idle listening by contextually matching the schedule used for probing to the current network conditions (Merlin and Heinzelman, 2007).

We now discuss the cross-layer principles in terms of interactions among physical (PHY), medium access control (MAC), network (both routing and organization), application sensing, and transport layers:

- MAC + PHY: In this joint optimization approach, the objective is usually energy savings (Haapola et al., 2005) by scheduling nodes ON and OFF as per specific communication and sensing correlation (Tian and Georganas, 2002) among nodes in the network. In general, the following techniques have been applied:

1. *Node scheduling as per redundant sensing coverage*: S-MAC (Ye et al., 2002b) is a MAC protocol that puts nodes into periodic sleep mode for energy conservation. Each node is free to choose its own listen/sleep schedules and broadcast the schedule to all its one-hop neighbors. Similar energy efficient MAC layer designs for general sensor networks can be found in T-MAC (Dam and Langendoen, 2003), SIFT (Jamieson et al., 2003), and CC-MAC (Vuran and Akyildiz, 2006).

2. *Node scheduling as per redundant connectivity*: Span (Chen et al., 2001) is a distributed randomized power-saving technique where nodes make local decisions on whether to sleep or to join a forwarding backbone as a coordinator. Each node bases its decision on an estimate of how many of its neighbors will benefit from it being awake and the amount of energy available to it. Span integrates with the IEEE 802.11 MAC protocol and tries to preserve both capacity and network connectivity while minimizing energy consumption. Similar energy efficient topology control and management optimizations have been proposed in (Ramanathan and Rosales-Hain, 2000) (Wattenhofer et al., 2001) (Schurgers et al., 2002) and are integrated with sensing redundancy by several hybrid MAC protocols (Chang and Chang, 2008) (Warrier et al., 2008) for sensor networks.

- MAC + Network: Because MAC level protocols have a very narrow view of the network, the main approach followed by such energy-efficient protocols has been to turn off radios that are not actively transmitting or receiving packets. Since there is a certain amount of time involved in turning radios back on when they are needed, MAC protocols typically trade off network

delay for energy conservation. Energy efficient MAC and routing protocols can be used together to increase energy conservation.

1. *Contention aware or receiver-based routing*: This is a very popular approach where the next hop is chosen as a function of the contention in the neighborhood. Receiver-based routing has been proposed to optimize the energy efficiency, latency, and multi hop performance of the routing algorithm (Skraba et al., 2004), (Zorzi and Rao, 2003). Similarly, in (Ferrara et al., 2005), the routing decision is a result of successive competitions at the MAC layer. In other words, the next hop is selected based on a weighted progress factor and the transmit power is increased successively until the most efficient node is found. Moreover, similar on-off schedules are utilized for energy efficiency as discussed earlier.

2. *Joint scheduling and routing scheme*: The general advantage of this approach is to turn OFF nodes that are not used for the routing service. Also, since WSNs are characterized by multiple flows from closely located nodes to a single sink, it becomes necessary for routing protocols to mitigate potential interfering routes in their route establishment. If the traffic is periodic as in the case of sensing applications that monitor periodically, the nodes can form distributed on-off schedules for each flow in the network while the routes are established such that the nodes are awake only when necessary. Since the traffic is periodic, the schedules are then maintained to favor maximum efficiency. Such a technique is proposed in (Sichitiu, 2004) for periodic traffic in WSNs. The usage of on-off schedules using a TDMA scheme and topological information in a cross-layer routing and MAC framework is also investigated (van Hoe-

sel et al., 2004). Another approach (Pantazis et al., 2009) that aims to conserve energy while minimizing end-to-end packet delay uses a similar TDMA-based scheduling scheme that balances energy-saving and end-to-end delay. This balance is achieved by an appropriate scheduling of the wake up intervals, to allow data packets to be delayed by only one sleep interval for the end-to-end transmission from the sensors to the gateway.

3. *Interference aware routing*: In (Fang and McDonald, 2004), the interference effect of the broadcast nature of MAC on routing is investigated. This MAC interference between routes is minimized by implementing the use of node codewords that indicate the interference level of nodes and each packet contains a route indicator for route establishment.

4. *Network organization and topology management*: Network organization and topology control algorithms use specific inter layer correlations and redundancies along with application level hints to support a customized topology for meeting application requirements. LEACH (Heinzelman et al., 2000) is an energy-aware cluster head selection mechanism for environmental monitoring sensor networks that is customized to support continuous and periodic monitoring. As discussed earlier, the topology of an ad hoc network plays a key role in the performance of networking services such as scheduling of transmissions, routing, flooding, and broadcasting. Xu (Xu et al., 2003) (Xu, 2002) proposes two topology control protocols (GAF and CEC) that extend the lifetime of dense ad hoc networks while preserving connectivity by turning off redundant nodes. Bao and Garcia-Luna-Aceves (Bao and Garcia-Luna-Aceves, 2003) propose

another topology management algorithm that constructs and maintains a backbone topology based on a minimal dominating set *(MDS)* of the network. Using only transmission control techniques for controlling the sensor network topology, both centralized (Tseng et al., 2003) and distributed (Kubisch et al., 2003) approaches have been proposed. In order to improve management efficiency across topology control, media access control, and routing, a unified cross-component power management architecture for wireless sensor network is proposed in (Xing et al., 2009).

- App + MAC: With this approach, the MAC layer is customized specifically to meet application objectives of efficient sensing, event monitoring, and target tracking. Such an application specific approach is limited to the domain to which it is applied and hence it is not universal. The Low Energy Self-Organizing Protocol (LESOP) (Liang and Dimitrios, 2007) for target tracking in dense wireless sensor networks is a two-layer Embedded Wireless Interconnect (EWI) architecture platform that is adopted for high protocol efficiency, where direct interactions between the application layer and the MAC layer are exploited. In this approach, the transport and network layers are excluded to simplify the protocol stack. A lightweight yet efficient target localization algorithm is then implemented, with a QoS knob employed in the application layer that controls the tradeoff between the target tracking error and network energy consumption.

- App + Network: Brooks *et al.* propose location centric CSP (Collaborative Signal Processing) approaches for target tracking sensor networks in (Brooks et al., 2003) and (Moore et al., 2003), where a selected region instead of an individual sensor node is activated. The approach uses local sharing of

robust statistics that summarize local events. Local collaboration extracts detection information such as time, velocity, position, heading and target type from the summary statistics. Groups of nodes used for local collaboration are determined dynamically at run time. Local collaboration information is compared with a list of tracks in the immediate vicinity. A variation on the nearest-neighbor algorithm associates detections to tracks. Zhang et al. propose an optimized tree reconfiguration scheme for target tracking (Zhang and Cao, 2004) adapted at the network layer but shaped by the tracking application requirements.

- Network + PHY: By solving the throughput optimization problem into two sub-problems: multi-hop flow routing at the network layer and power allocation at the physical layer, the authors (Yuan et al., 2005) propose a cross-layer optimization of network throughput for multi-hop wireless networks. The throughput is a function of the per-link data flow rates, which in turn depends on the link capacities and hence, the per-node radio transceiver power level. On the other hand, the power allocation problem is a function of the interference as well as the link rate. Based on this solution, a CDMA/OFDM based solution is provided such that the power control and the routing are performed in a distributed manner. In (Seada et al., 2004), new forwarding strategies for geographic routing are proposed based on the results in (Zuniga and Krishnamachari, 2004). The authors provide expressions for the optimal forwarding distance for networks with automatic repeat request (ARQ) and without ARQ. The forwarding algorithms require the determination of the packet reception rate of each neighbor in order to select the next hop and construct routes accordingly.

- Transport + PHY: In (Chiang, 2005), a cross-layer optimization solution for power control and congestion control is considered. Based on this framework, a cross-layer communication protocol based on CDMA is proposed, where the transmission power and the transmission rate is controlled. However, the proposed solutions apply only to CDMA-based wireless multi-hop networks, which may not apply to WSNs as CDMA technology is not currently feasible with battery powered sensor nodes.

- 3-Layer Solutions: In addition to the proposed protocols that focus on pairwise cross-layer interaction, more general cross-layer approaches among three protocol layers exist.

    1. TRANSPORT + MAC + PHY: In (Madan et al., 2005), the optimization of transmission power, transmission rate, and link schedule for TDMA-based WSNs is proposed. The optimization is performed to maximize the network lifetime, instead of minimizing the total average power consumption.

    2. NETWORK + MAC + PHY: In (Cui et al., 2005), joint routing, MAC, and link layer optimization is proposed. The authors consider a variable-length TDMA scheme and MQAM modulation. The optimization problem considers energy consumption that includes both transmission energy and circuit processing energy. Based on this analysis, it is shown that single-hop communication may be optimal in some cases where the circuit energy dominates the energy consumption instead of transmission energy.

    3. App + Network + MAC: In (Choe et al., 2009), an integrated and adap-

tive cross-layer data reporting scheme is proposed that supports information quality of service in terms of good throughput performance and stable data reporting at the end system. This is done by efficiently controlling data reporting functions in communication layers considering parameters from other layers. The main work of this paper focuses on the QoS-aware data reporting tree construction scheme, called QRT, and the QoS-aware node scheduling scheme, called QNS. QRT constructs a data reporting tree based on the conditions of the end-to-end delay and the traffic load to find data reporting paths from each cluster head, which has already collected data from its cluster members, to a sink. QNS schedules a certain number of nodes that are selected based on the QoS requirements in a cluster to report data to its cluster head in a collision-free manner.

Tian and Ekici (Tian and Ekici, 2007) propose an application-independent task mapping and scheduling solution in multi-hop homogeneous WSNs, Multi-hop Task Mapping and Scheduling (MTMS), that provides real-time guarantees. Using their proposed application model, the multi-hop channel model, and the communication scheduling algorithm, computation tasks and associated communication events are scheduled simultaneously. The Dynamic Voltage Scaling (DVS) algorithm is presented to further optimize energy consumption. A similar optimization known as EcoMapS (Energy-constrained Task Mapping and Scheduling) that incorporates channel modeling, concurrent task mapping, communication and computation scheduling, and sensor failure handling algorithms is proposed in (Tian et al., 2007).

The Unified Role Assignment Framework (URAF) uses the concept of roles which are tasks that are assigned to nodes in the network by way of rules. Rules are metrics that identify performance requirements for any node and/or its neighbor to be assigned a task (or a set of subtasks). The assignment of tasks to nodes and their subsequent scheduling and execution is done at a layer above the network layer. The metrics that are elicitepd from nodes for role-assignment may come from any layer below the URAF abstraction layer. So, the framework assumes the existence of a vertical cross-layer information data base that is maintained and updated by respective layers. In that sense, the URAF layer is not intrusive and does not involve special modification or customization of lower layers except addition of interfaces to share the available information at each layer. However, specific customizations that are expected by applications at other layers may be possible only by providing customized protocol stacks that interface with the unified role-assignment framework. In other words, we leave the application specific optimization of the behavior of roles and their execution details to the programmer. These application specific cross-layer protocol optimizations are abstracted as specialized roles (with their respective rules for control) in the unified role assignment framework.

2.3 Generic Sensor Network Protocol Abstractions

In this section, we will discuss existing research on generic sensor network protocol abstractions. As mentioned earlier, a generic protocol abstraction is essentially a framework that hides application specific optimizations of both individual layers as well as across layers and exposes them to the application layer by way of control interfaces provided as the middleware. In general, sensor network protocols are customized to exploit both the application and the domain specific correlations to gain

the maximum benefit. These benefits are usually in terms of energy savings, tracking efficiency in terms of delay and information quality, etc. Such customizations usually result in these protocols embodying very different assumptions about network stack composition and, as such, having limited inter-operability. It has been suggested (Culler et al., 2005) that, in principle, wireless sensor networks would benefit from a unifying abstraction and that if the architecture has a "narrow waist" (as does the Internet architecture), then it could effectively decouple many aspects of the application software from the underlying customizations at specific protocol layers and differing sensor network platforms (operating systems and hardware). Such a decoupling would be of great benefit given the rapid technological advances in the sensor networks arena, particularly in heterogeneous deployments.

This section is organized as follows. We first briefly discuss the motivations for a generic network abstractions for wireless sensor networks. These motivations broadly fall under the application development and deployment flexibilities made available to a domain expert to program a heterogeneous wireless sensor network across diverse applications and deployment environments. The second subsection discusses the generic requirements that can be minimally desired by any abstraction. In general, due to the application-specific utility of wireless sensor networks, the flexibilities required from an abstraction may differ widely. Some may require maximal control for the use and allocation of sensor network resources both at the node level and at the network level, and across services and tasks executed within the network. On the other hand, domain experts would still want all of this but with minimal programming language complexity. Such tradeoffs are still being explored by the research community and the variety of existing network abstractions proposed and still under active development offer testimony of the fact

that there are complex research issues still open in this area of active research. In the third subsection, we develop a taxonomy of network abstractions to better understand the classes of solutions at a high level. Using this taxonomy, we briefly discuss representative network abstractions for every class of solutions. We also identify to which class of network abstractions our unified role-assignment abstraction framework (URAF) belongs. In this process, we identify the features of URAF abstractions and its limitations.

A detailed discussion that surveys the recent programming abstractions as applicable to the field of wireless sensor networks and their application requirements is provided by Mottola and Picco (Mottola and Picco, 2010) and Sugihara and Gupta (Sugihara and Gupta, 2008).

2.3.1 Motivations

Software architectures for networked sensors are typically concurrent and event driven. However, event triggered programming models are not natural for programmers. This is because applications have to be written as explicit state machines, which are difficult to understand and maintain. Hence sensor networks are notoriously difficult to program, given that they encompass the complexities of both distributed and embedded systems. To address these problems, there is a need for application developers to access generic network abstraction interfaces in the form of programming language constructs. Besides programming flexibilities, we identify the following motivations for generic programming abstractions for wireless sensor networks:

- Programming incentives: Usually it is expected that the application developer for a wireless sensor network application is more conversant with the domain

knowledge than low level protocol details. Such domain experts need a general purpose programming language that provides similar goals to higher level programming for desktop and server regimes. These are usually: flexibility, ease of development and debugging, and portability. However, since sensor networks are resource constrained and highly customized across protocol layers, it becomes necessary that such programming languages be as natural a fit for sensor networks as possible while significantly improving software productivity, quality, and also the production of efficient, resource constrained code.

- Modularized and adaptable interfaces across heterogeneous systems and diverse applications: With the diversity of wireless sensor network applications and the facilities provided by network abstractions, it becomes necessary that network abstractions be modularized and act as building blocks with each other. In other words, no single abstraction may be completely generic such that it supports all the requirements expected by any application across diverse domains and heterogeneous platforms. In such scenarios, a coherent framework can be developed by programming requisite abstractions to collaborate with each other such that it supports the programming flexibilities desired by any application (Mottola and Picco, 2010).

- Cross layer optimization flexibility: As discussed in the earlier section on cross-layer optimization, sensor network applications can greatly benefit from the ability to manipulate cross-layer protocol functionalities in order to minimize resource usage while meeting another performance metric (Akyildiz et al., 2002). Sensor network abstractions provide low level hooks (or interfaces) to programmers along with specific usage templates for varying performances

across for all scenarios.

- Standardization: A generic network abstraction can provide standardized capabilities across all stages of sensor systems development and deployment. In general:

    1. Application conception and requirements elicitation: A generic network abstraction by way of its interfaces, its programming flexibility and also by its adaptation capabilities across other abstractions eases the way system requirements can be generated and mapped for a certain application. Standardization of network abstractions in terms of these requirements reduces the learning curve of developers and also reduces the time to deployment.

    2. Application development: With the availability of a generic set of standardized performance templates for every abstraction that hides optimization across layers, development and debugging become easier. Also a varying set of code-metrics including but not limited to the number of lines of code (LOCs) are available to identify the complexity of the development process and planning can be done accordingly for subsequent stages.

    3. Deployment and support: By supporting generic and standardized debugging and testing strategies across network abstractions, post deployment support becomes less cumbersome. In other words, defects can be clearly identified either in requirements or in the implementation. In either case, mapping and resolving both kinds of defects is easier when standardized sets of abstractions are used. Comparison and documen-

tation of application performance across domains and differing scenarios becomes concise and effective. This paves the way for developers to gather experience from prior deployments and reduce errors.

2.3.2 Generic Requirements

The requirements from a network abstraction need to be considered from different perspectives. These are application demands, architectural adaptability, level of programming abstraction, level of resource control in terms of cross layer solutions, and other generic distributed systems requirements such as fault tolerance and reliability. We discuss these briefly:

- Application demands: Sensing applications may demand the following services from the sensor network: sense or monitor a particular region, track an event across a region, or sense and react under certain conditions. The services can be executed periodically or they may be event triggered. Nodes may be heterogeneous and adorned with differing capabilities for sensing, communication, computation, storage, and energy (battery powered or uninterrupted). The network deployment may be static and regular or mobile with random placement. The communication pattern may be many-to-many or many-to-one. The former is the case for sense-and-react applications whereas the latter is for sense-only applications such as data collection. The quality of sensing desired may be based upon both the information quality and confidence measured in terms of both the number of samples and the number of sensors in agreement with these samples. In addition to these, network abstractions may need to consider the energy consumption for all activities needed to execute the service and optimize the use of resources such that application QoS

is satisfied without significant depreciation in network lifetime. All of these need to be made available as open interfaces for flexible control by application developers without significant development and debugging overhead.

- Architectural adaptability: A network abstraction that interfaces with protocols across layers and brings up control interfaces for tweaking by application developers needs to adapt to both its evolution as well to its collaborative usage with other protocol abstractions. The evolution of an abstraction is more related to its software architecture in terms of its interaction with other protocols across layers. A vertical abstraction that interfaces with every layer through simple and generic interfaces is considered to be the least intrusive approach. It gives both the protocol stack and the cross-layer abstraction to evolve independently of each other as long as the provider-subscriber interface between these two respective software subsystems are maintained. A horizontal abstraction between an application and the layer below either does not give much control or it achieves higher control flexibility only by intrusively interacting with all layers. In any case, as discussed earlier, cross layer optimizations, are usually not portable and a network abstraction ends up getting tied to a specific protocol stack and is not generic. Hence network abstractions need to tradeoff between the amount of control flexibility they want to open up for programming and its need to be portable and collaborative with other abstractions based on different protocol stacks.

- Level of programming abstraction: At one extreme, low level programming platforms and languages make programming cumbersome and error-prone. At the other extreme, declarative approaches greatly facilitate programming but restrict what can be done. In both cases, additional limitations include lack of

support for concurrency, difficulties in changing applications, and insufficient abstractions from low-level details. To address some or all of these limitations a virtual machine (VM), which is a platform-independent programming abstraction, is developed for heterogeneous sensor platforms and applications. Low level programming leaves the responsibility of message communication to the programmer whereas in the case of declarative approaches (as well as in VM abstractions), the programmer is either shielded from message handling or messages are higher abstractions for the final data that is collected globally or regionally across neighbors. Thus, data sharing in low-level languages needs to be managed at the node level whereas for other higher level abstractions it is addressed at the neighborhood, region, or network level. Similar trends apply for both the computation scope and the communication awareness. A programmer using a low-level abstraction can assign computation at the node level and is aware of link level communications. On the other hand at higher abstractions the computation is assigned to a group or region and the communication awareness is implicit such that the programmer is completely oblivious of links, neighbors, and even routing. At such a declarative level of abstraction, the sensor network is usually viewed as a database and the programming is more similar to SQL.

- Cross layer hooks for fine-grained resource control: Resource allocation and scheduling at node, neighbor, group or network level requires knowledge of information maintained at various layers in the protocol stack. With WSNs deploying nodes that are battery powered, preserving network lifetime gets introduced as one of the important criteria to be optimized along with application objectives. The lifetime of an ad hoc network is measured usually

in terms of the time when there is a significant hole or partition in the network such that a majority of the nodes are unreachable. Network abstractions provide specific open interfaces for metrics they are optimizing for a certain application. A low-level abstraction is basically an operating-systems level abstraction and the optimization is more related to the way basic services such as process or task scheduling, timers, memory, and device management are implemented. A declarative abstraction as mentioned earlier provides resource control at a very high level in terms of the number of samples for a query resolution, number of such queries initiated within an interval, and the periodicity or longevity of the queries. Thus, declarative approaches are not suitable for fine-grained resource control although specific underlying cross-layer optimizations may have been abstracted away from the application developer. Usually these hidden optimizations relate to the way the query is disseminated to nodes in the network, the way replies are collected and aggregated from neighbors and transmitted back to the sink.

- Generic distributed systems requirements: These relate to the general problems in distributed systems where continuous coordination for the use of shared resources is required to be achieved without any conflicts or deadlocks. These challenges are related to fault-tolerance. Specific examples of related problems include consensus problems, Byzantine fault tolerance, and self-stabilization. All these problems need to be efficiently resolved in terms of space, time, and communication complexity for sensor networks, and handled effectively by application developers controlling the use of a set of programming abstractions.

2.3.3 A high level taxonomy: Abstractions, Programming Languages, Virtual Machines, Middleware

For ease of understanding, we have classified existing sensor network protocol abstractions in a layered manner resembling standardized protocol stacks. As seen in figure 2.2, the taxonomy includes both generic as well as specific (or non-portable) sensor network abstractions. The specific abstractions are basically cross-layer protocol optimizations that are relevant only for a specific application and context that include a specific deployment environment, hardware, and software capabilities. As these are non-portable and application specific, the solutions fall under the category of cross-layer optimizations that we discussed in section 2.2.

The current trend is toward a thin, flexible, simple, and high level yet control-rich generic programming abstraction. The taxonomy shown in figure 2.2 represents a range of abstractions from the lowest OS/instruction level that are programmed by threads/functions or virtual machines (VMs) to the highest network and/or middleware programmed at a macro level. Depending upon the level of abstraction, programmers will have the ability to address the fundamental operations such as communication, computation, and data access (and sharing) within a specific programming paradigm supported by the abstraction. The programming constructs supported by several programming paradigms allow a programmer to represent the individual elements of a program such as variables, functions, or steps that compose a computation i.e. assignments, and iterations. A survey of the state of the art in network abstractions (Mottola and Picco, 2010) reveals three major programming paradigms:

1. Imperative: Here the programming constructs require the programmer to deal with state and event driven programming at node level. At another end, the

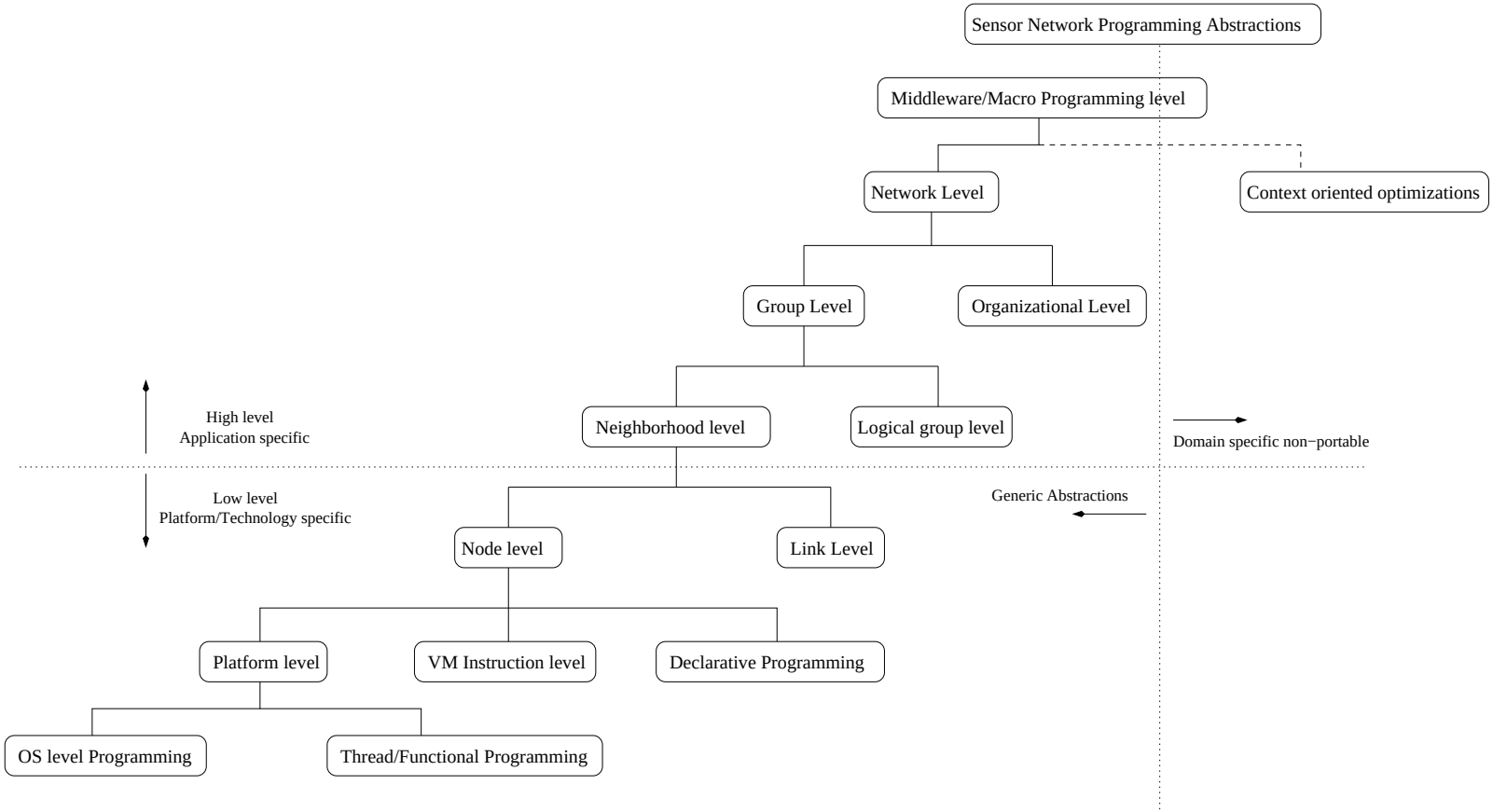

Figure 2.2: Sensor Network Abstractions Taxonomy

imperative approach has been considered to include sequential programming where the programmer instead views the system as a network and is not required to get into lower level event handling and state transitions.

2. Declarative: Here the programmer describes the application processing without actually specifying how it is accomplished. Declarative approaches include functional, rule-based, SQL-like, and other special purpose language mechanisms.

3. Hybrid: this is a combination of the above programming paradigms.

In the next subsection we will discuss representative examples of network abstractions for each category identified in the taxonomy shown in figure 2.2. In particular, we will discuss the flexibilities offered by the framework both in terms of ease of programming and the richness of the abstractions in providing appropriately grained high to low level resource control.

2.3.4 Survey of representative programming abstractions

Node level abstractions

The lowest level abstraction is essentially a node level programming model that is abstracted by an operating system such as TinyOS (Hill et al., 2000) or a node-level programming language above it. TinyOS is the most widely used operating system in wireless sensor network applications. It is written in nesC (Gay et al., 2003), which is a variant of C. It is a component based OS allowing modular programming. Using two types of components, modules and configurations, along with commands and event based interfaces, a programmer is given the flexibility to wire different components together. This is done by connecting interfaces used by

components to interfaces provided by others. In the former case, the component (known as the user) using the command interface of other components (known as providers) is required to implement the event handlers for the respective providers. In this way, nesC supports encapsulation between modules as programmers are freed from being aware of whether modules are implemented in hardware or software. However, as the level of abstraction is very low and since TinyOS excludes blocking operations, even simpler programs need to deal with the rigorously event-based programming approach. The communication primitives in nesC are provided by the set of Active Messages (AM) interfaces. In Active Messages, messages are tagged with an identifier that specifies which component must process them upon reception. Since nesC is non-blocking, programmers are forced to deal directly with message parsing, serialization, packet buffering, and even scheduling and handling unreliable 1-hop/broadcast transmissions and receptions. The mechanisms implementing AM interfaces are usually tied to a specific MAC layer (Ye et al., 2002b) and radio chip (Polastre et al., 2004). Although a lot of sensor APIs are provided for different types of sensor hardware, interfaces for actuators are still evolving. In summary, although a raw level of control is provided to the programmer, this level of abstractions entails higher application development and debugging complexity.

The complexity introduced at the node level programming by TinyOS/nesC are resolved by other abstractions in a number of ways. First, in order to overcome unreliable communication for both 1-hop unicast and broadcast, a suite of multi-hop protocols for data collection and dissemination (Intanagonwiwat et al., 2000) (Ye et al., 2002a) have been developed atop the Active Messages interface. In order to reduce or eliminate the complexity of state/event based asynchronous programming, additional software engineering by adding more layers for extending the event

model has been proposed. T2 (Levis et al., 2005b) is a new version of TinyOS that provides a new abstraction boundary that supports a telescoping abstraction for application programming. This is a hybrid approach consisting of horizontal decomposition for the lower level to support various types of hardware devices and vertical decomposition to support higher level platform independent functionality for various hardware platforms. Another prior approach is SNACK (Greenstein et al., 2004) that provides reusable service libraries for programmers to combine for building applications. Services in SNACK are similar to configurations in nesC but are more customizable through the availability of parameterizable configurations.

Thread based abstractions eliminate the asynchronous event-driven and non-blocking related complexity of nesC by supporting statically allocated threaded blocking execution contexts. The event driven model is more suitable for allowing the microcontroller to sleep as much as possible, thereby achieving energy efficiency. Although thread abstraction often simplifies the programs significantly, its major limitation is its need for static allocation of per-thread stack which is often too expensive in terms of memory space. A number of thread based programming abstractions have been reported in the research literature. Thread-based abstractions include Fiber (Welsh and Mainland, 2004), Mantis OS (Bhatti et al., 2005), TinyThread (Mccartney and Sridhar, 2006), Protothreads (Dunkels et al., 2006), and Y-Threads (Nitta et al., 2006). These differ in terms of the number of features provided such as lightweight concurrency support, preemptive time-sliced multithreading, cooperative multithreading, and efficient per-thread stack (or stackless) allocations.

Virtual machine based abstractions

Virtual machines based programming abstractions for sensor networks not only provide platform independence and isolation but also node/network reprogramming by way of dynamically injecting new codes into each on-site node. There are interpreter-based virtual machines that are customized for a specific application in order to reduce the number of instructions (or the size of the assembly code) transmitted to each node. This reduces the communication overhead and hence minimizes energy usage. Maté (Levis and Culler, 2002) and ASVM (Levis et al., 2005a) are stack-oriented application-specific virtual machines implemented on top of TinyOS. On the other hand Melete (Yu et al., 2006) extends Mat*é* by supporting multiple concurrent applications. Additionally, VMStar (Koshy and Pandey, 2005) is a virtual machine that supports the dynamic update of the system software such as the VM itself along with updates to the application code.

At a higher level of abstraction i.e. the middleware, a virtual machine can further reduce the code size in addition to realizing the benefits of specific application customizations. Examples of middleware level VMs include Impala (Liu and Martonsi, 2003) and SensorWare (Boulis et al., 2003). Impala is a middleware designed for the ZebraNet project (Juang et al., 2002) and its modular design supports easy and efficient on-the-fly reprogramming via a wireless channel. SensorWare uses Tcl-based control scripts as the reprogramming language and uses a high-end hardware platform as compared to Maté/ASVM.

A virtual machine provides platform-independent execution models for application development. TML (Token Machine Language) (Newton et al., 2005) is one such intermediate language that provides a distributed token machine (DTM) as the execution model where each node sends and receives tokens to/from other nodes

and the associated token handler is executed upon receiving a token. TML along with its DTM execution model semantically supports a much higher level programming constructs compared to the low-level constructs provided by nesC. TML thus seems to make programming much simpler for domain experts.

Group level abstractions

Group level abstractions relinquish node-level resource control to gain lower complexity in application development. Resource control at the node level requires the programmer to trade off specific node operations such as computation, communication, and storage for energy savings. As discussed earlier, the development complexity at the node level requires high technical expertise not only at the domain level but also at the platform level. Group level abstractions reduce all these operations to shared computation and data sharing. The programmer is now made completely agnostic of the communication complexities. The formation of the group and its maintenance is managed by a lower level group management protocol.

Group level abstractions can thus be classified based upon how groups are formed. In the survey by Mottola and Picco (Mottola and Picco, 2010) and Sugihara and Gupta (Sugihara and Gupta, 2008), group based abstractions in general have been classified into neighborhood based abstractions and logical based abstractions. The neighborhood abstraction is based on physical wireless proximity usually within 1-hop or within direct radio range. The logical abstraction is based upon some logical relationship among members in a group such as required node features in terms of types of sensors and/or actuators. However such a logical relationship among requisite sensors is also complemented by those sensor readings that are in certain conformance (agreement or otherwise) to an event that is being monitored

or tracked.

Physical proximity-based group abstraction is essentially a representation of the localized nature of distributed protocols that are typical of applications executed in an ad hoc wireless sensor network. It fits well with the broadcasting nature of wireless communication and enables efficient communication within the group. In essence, physical neighborhood represents an abstraction that is one level below the organized network architecture. With this abstraction, the programmer gets an extra flexibility to organize the network according to the requirements of the application. However, with this level of abstraction, group management in the presence of node and event mobility becomes challenging and entails higher logical complexity for the programmer. Maintenance of collective state and its eventual migration across groups in case of target tracking also adds to the complexity.

Logical group based abstractions on the other hand are dynamic and much broader than neighborhood based abstractions that are based on physical closeness. Instead, a logical group uses higher level logical properties that include the type of nodes, sensor inputs about the dynamic and ambient environment, and the volatility of group membership. Logical groups are addressed by way of indirections which refer to the event they are currently monitoring or tracking. In other words, events determine the creation, movement, and the persistence of a group. Node attributes that are relevant to the event at hand are logically exported across nodes (multi-hop) and are logically made available as group level attributes. Group formation is essentially a sort of predicate that is run across nodes in the vicinity of the event and these predicates specify required group properties or attributes. A dynamic and distributed group management protocol revolving around these logical attributes is supported by the programming abstraction and the programmer needs to simply

identify required logical group properties for an application desired event.

The unifying link abstraction for WSNs (Polastre et al., 2005) is one of the neighborhood based lower layer thin waist architectures that abstracts a broad range of link-layer technologies (B-MAC on mica2 and IEEE 802.15.4 on Telos) and supports a wide variety of network protocols and while avoiding significant loss of efficiency. This is done by exposing sets of packets, exerting simple reliability and urgency controls, adapting to congestion and loss after concerted effort, and by cooperating in neighbor management and schedule formation. Network protocol designs can thus be done at a fairly high level by application developers without concentrating on link specifics. Moreover, these protocol optimizations are agnostic on the underlying link layer technology. The authors provide experimental evidence of power-aware network protocols expressed in terms of their link layer abstractions being mapped efficiently to very different link-level power management mechanisms.

Abstract Regions (Welsh and Mainland, 2004) and Hood (Whitehouse et al., 2004) provide a higher level neighborhood based abstraction compared to the link based abstraction. The programming interface of Abstract Regions provides neighborhood discovery, variable sharing via a Linda-like (Gelernter, 1985) tuple space, and also MPI-like reduction operations. In Abstract Regions, groups are defined either topologically (e.g. n hops), geographically (e.g. k-nearest neighbor), or by their combinations. Hood defines a similar set of operations as Abstract Regions, but uses a one-hop neighbor as the sole option for group definition. Abstract Regions also provides a way to tune the tradeoff between resource consumption and accuracy in its runtime component.

Kairos (Gummadi et al., 2005) supports a centralized and imperative programming language as extensions to the Python language where constructs are provided

to iterate through the neighbors of a given node, and communication occurs by reading or writing shared variables at specific nodes. The application processing is expressed as pairwise interactions between neighboring nodes which in a way resembles the high-level localized behavior of generic sensor network algorithms. Another approach for sharing data is based on the popular tuple space abstraction known as Linda (Gelernter, 1985). A tuple space is a shared memory space where different processes read/write data in the form of tuples. TeenyLime (Costa et al., 2007) using the tuple-based abstraction, supports an asynchronous programming model similar to nesC atop TinyOS. This means that Teeny-Lime operations are non-blocking and return their results through a callback. Tuples are shared among nodes within radio range. In addition to Linda's operations to insert, read, and withdraw tuples, reactions allow for asynchronous notifications when data of interest appears in the shared tuple space. Similarly, capability-based tuples are also provided that enable on-demand sensing that reduces energy consumption by avoiding the need to keep sensed information up to date in the shared tuple space in the absence of data consumers. TeenyLime constructs have been used and extended to develop not only stand-alone applications but also system level services such as routing protocols. This has been demonstrated by the real-world deployment used to monitor heritage buildings by Ceriotti *et al.* (Ceriotti et al., 2009).

Snlog (Chu et al., 2007) is a rule-oriented declarative approach inspired by logical programming constructs such as predicates, tuples, facts, and rules. Similar to the format of records specified by tables in relational databases, predicates in SnLog specify schemas for data as ordered sequences of fields. Tuples represent the actual data which are similar to instantiated records in database tables. Facts are particular tuples that are instantiated at system start-up, whereas rules express the

actual processing. Distributed executions are described using a location specifier which maps to a node hosting a tuple and/or a rule. To support low-level resource control, nesC code can be linked to the rule engine. Similar to TeenyLime, Snlog has been used at different levels of the protocol stack, to implement services such as target tracking and routing.

EnviroTrack (Abdelzaher et al., 2004) is a logical group-based programming abstraction specifically for target-tracking applications. Its underlying object-based programming framework known as EnviroSuite (Luo et al., 2006) uses objects to represent physical entities in the environment. In other words, addresses are assigned to physical events and a one-to-one mapping between objects and physical entities are maintained as they move in the environment. A group is a collection of sensors that sense the event and are thus uniquely responsible for its corresponding object representation. A leader is elected among members of the group sensing the moving target and this leader is responsible for collecting data from group members, performing necessary computations, and also managing group management in the presence of event mobility. Similar to Abstract Regions and Hood, EnviroSuite also supports relevant data sharing and aggregation facilities. Additionally, it provides a sophisticated distributed group management protocol based on routing trees, to maintain objects bound to a fixed set of nodes and a mechanism to support objects mapped to moving entities.

Logical Neighborhoods (Mottola and Picco, 2006) is a programming abstraction that allows a programmer to define a node's neighborhood based on the logical properties of the nodes in the network instead of their physical position. In other words, a node is logically represented in terms of several node attributes it exports to its neighbors. These attributes include both static (e.g. node type) and dynamic

properties (e.g. sensor readings). A logical neighborhood is defined using predicates over node templates. Nodes periodically disseminate their logical profile, that is, their list of current attribute-value pairs. To avoid flooding the entire system, the underlying protocol exploits the redundancy among similar profiles to limit the spreading of information. The definition of Logical Neighborhoods is supported by a declarative programming language called Spidey. Spidey provides programmers with predicate based constructs to identify logical neighborhoods that are based on a certain set of attributes such as the number of hops and the desired communication costs. Spidey provides communication APIs within the logical neighborhood and also an efficient routing mechanism. Programmers interact with the nodes in a logical neighborhood that mimics the traditional broadcast-based communication. Instead of the nodes within radio range, however, the message recipients are the nodes matching a given neighborhood definition. Therefore, programmers still reason in terms of neighboring relations, but retain control over how these are established. Logical Neighborhoods is suited to the highly heterogeneous and decentralized scenarios typical of sense-and-react applications, where the processing often revolves around programmer-defined subsets of nodes.

ATaG (Bakshi et al., 2005) is a programming model that has its core programming notions based on abstract task and abstract data. A task is a logical entity encapsulating the processing of one or more data items, which represent the information. Different copies of the same task may run on different nodes. The flow of information between tasks is specified declaratively with abstract channels connecting a data item to the tasks that produce or consume it. The code in a task is written in an imperative language, and relies on a shared data pool for local communication, allowing tasks to output data or to be notified when some

data of interest becomes available. It thus features a hybrid programming interface wherein the communication among tasks executed on separate nodes is described in a declarative manner whereas the local node computation is expressed using an imperative language. ATaG enables programmers to express multi-stage incremental data-centric processing. In general, ATaG is suited to applications that employ both sensors and actuators. In such scenarios, complex operations are needed on the sensed data from various sources for the actuators to make a decision. The programming abstraction supports a compiler that processes input such as the description of tasks and channels, flow of data associated to them, and the location of the nodes. Using this information, the compiler outputs an optimal allocation of tasks to nodes. ATaG however is limited and requires the programmer to support a appropriate routing scheme for the specific application and target environment.

The Generic Role Assignment (GRA) (Frank and Romer, 2005) (Frank and Romer, 2006) abstraction supports a declarative programming language that allows roles to be identified among neighbors in the network based on a certain set of programmer specified logical attributes. It is a language specific extension of the Role Based Hierarchical Self Organization protocol (RBHSO) (Kochhal et al., 2003) and in essence supports dynamic self-configuration of WSN according to programmer-specified requirements. Similar to RBHSO, GRA considers a role specification as a list of role-rule pairs. For each role, the corresponding rule describes the conditions for the role to be assigned to the local node. Rules are expressed as boolean predicates referring to the properties of the node considered. To account for changing node properties and network dynamics, the role specifications are periodically re-evaluated and are also initiated dynamically whenever the role or properties of neighbors change. Similarly, distributed protocols are provided to trigger role re-

evaluation based on topology changes. GRA however leaves the concerns of lower-level data collection and dissemination to other complementary solutions. Hence GRA does not support complete application development and is expected to be used in conjunction with other approaches.

Finally, there are other related logical group based programming abstractions such as Regiments (Newton and Welsh, 2004) (Newton et al., 2007), RuleCaster (Bischoff and Kortuem, 2007), Spatial Programming (Borcea et al., 2004), snBench (Ocean et al., 2004), and Virtual Nodes (Ciciriello et al., 2006). The survey by Mottola and Picco (Mottola and Picco, 2010) discusses these abstractions in relation to both the programming language taxonomy and the programming abstraction taxonomy with particular emphasis to their applicability across several application domains and sensor network platforms.

Network/Macro level abstractions

Network level abstractions considers programming at a macro level that is higher than group, node, or link-level abstractions. Since applications on sensor networks are data-oriented, the database approach is an intuitive abstraction at the macro level. TinyDB (Madden et al., 2005), Cougar (Bonnet et al., 2000) (Yao and Gehrke, 2002), SINA (Shen et al., 2001), MiLAN (Heinzelman et al., 2004), and DSWare (Li et al., 2004) are the representative macro-level programming abstractions from the research literature that we will briefly discuss in this section.

TinyDB and Cougar are two of the earliest examples of high-level database oriented abstractions for sensor network programming. They allow users to issue queries in a declarative SQL-like language. A routing tree spanning all nodes in the network is maintained and is used to route queries injected from the base station.

The routes are then decorated with meta-data to provide information on the type and nature of data sensed by nodes in a specific portion of the tree. To achieve energy efficiency, Cougar pushes selection operations to the sensor nodes so that they can reduce the amount of data to be collected. TinyDB, on the other hand, interleaves data sampling and transmission scheduling at different levels of the tree to minimize power consumption without affecting the quality of the reported data.

SINA extends the SQL programming interface for the database abstraction by allowing users to explicitly embed tasks for nodes in the network. This is achieved by way of embedding scripts written in an imperative language called SQTL (Sensor Querying and Tasking Language) in the SQL query. This hybrid approach therefore supports greater programming flexibility by allowing more complex collaborative tasks.

MiLAN provides programmers of health monitoring applications the flexibility to trade system lifetime for data quality. In MiLAN, an application submits a query with a QoS requirement, where the QoS of a variable is a function of the specific sensors used to compute the variable's value. In other words, QoS is defined by the level of predefined reliability with which each sensor can measure some basic attributes. In order to accommodate changing application requirements over a period of deployment, MiLAN models these changes in terms of a state machine with different QoS requirements associated with different states. In response to an application query with the QoS based on the desired quality of data, MiLAN creates an execution plan that involves creating a routing tree rooted at the source and a feasible set of sensors. This subset of nodes is computed such that they collectively provide a QoS greater than or equal to the minimum acceptable by the application while simultaneously maximizing energy efficiency.

DSWare is a message passing QoS-aware middleware that is geared toward real-time applications for detection of sporadic events. Programmers are provided with a form of publish/subscribe (Eugster et al., 2003) SQL constructs that specify subscriptions expressing the characteristics of the phenomena of interest and register them with DSWare. The user is then notified upon the occurrence of an event by way of another higher level notion of events which allows programmers to infer the occurrence of a phenomenon from raw sensor observations. For example, an event can be defined as the composition of two physical sub-events occurring within a specific time interval. Similar to QoS support in MiLAN, DSWare has a notion of confidence about event detection that fine tunes the relationships among sub-events, that is, their relative importance or fitness to a pattern. DSWare also supports real-time semantics where users can specify the time constraint in terms of the latency until getting a notification after detecting an event. Subscriptions are propagated in the network and a routing tree is built as a consequence that connects the base station to the relevant sensor nodes. To minimize energy consumption, DSWare eliminates redundant transmissions for subscriptions to the same data by merging routing paths leading to different base stations. It also takes advantage of differing subscription rates and supports real-time delivery of event notifications by way of the earliest-deadline-first scheduling mechanism. An alternative, energy aware scheduling technique is also provided for events with relaxed delivery constraints.

Comparison of URAF with other abstractions

The Unified Role Assignment Framework (URAF) can support a GRA like declarative programming language for its role selection mechanism which is abstracted by way of rules. However, URAF gives the programmer flexibility to log-

ically address a region of roles as a complex role. This aspect is similar to Logical Neighborhoods where the programmer is agnostic of the node-level communications. In other words, the set of roles within a region that are abstracted as complex roles can span any number of hops and the hop-by-hop communication among nodes is hidden from the programmer. The task make up for any role is a lower level control flexibility that the programmer can exercise. This role-based task composition is similar to the ATaG compiler that requires the programmer to specify both the data and the tasks that operate upon it. In essence, the data-oriented nature of sensing applications imposes upon the programmer a greater complexity if there was a need to program the behavior of a role in terms of task and data. The level of resource control is limited only to the level of fine-grained resource usage estimation of any role both at the node and network level. Data sharing among nodes is accomplished through message passing. Role execution by way of tasks is non-blocking and is taken care of by a generic role-state machine that is hidden from the programmer. Services are scheduled in terms of roles where there are specific slots allocated for service requests, service execution by way of roles, and role feedback (or repair). Currently, a periodic schedule limits the use of the URAF framework to sense-only applications that require periodic monitoring. Moreover, a definitive domain dependent model is needed for both the role-execution time and its resource usage in order for URAF to be resource aware both in terms of time and energy.

2.4 Related role-based concepts in distributed systems

In this section, we discuss the concept of roles as jobs or tasks assigned to workers or agents in a distributed multi-agent systems (MAS). In general, roles represent the task assigned to an entity or agent in a MAS system. Specific MAS-oriented

examples include AI based role formulation and assignment for RoboCup (Stone and Veloso, 1999), P2P organization and structuring for content management and retrieval (Victor, 2004), and role organization and assignment in WSNs (Vinyals et al., 2010).

2.4.1 Role Abstraction

A role-based abstraction is a logical abstraction of a transient task assigned to an entity in a distributed system. Besides task assignment or placement, a role also captures the behavior of the task according to local network situation. In general, such a behavior depends upon a specific application domain. For example, in case of RoboCup where robots coordinate among each other to play soccer, the roles abstracted include defender, forwarder, attacker, blocker, etc.

The assignment and the transient behavior of the roles is dependent upon the optimality of an entity in the distributed system to play a certain role such that overall team performance is improved and the end goal of producing a win becomes feasible. A role allocation algorithm thus involves an optimal allocation of a set of roles to robots such that the team of players stand a greater chance to score a goal against the opponent and win the game.

Similar abstractions are modeled for WSNs depending upon the application. For example, in the case of static wireless sensor deployment, besides the remote sink role, other topology-based roles include the cluster head, cluster members, cluster gateway, and a hierarchical regional cluster (that acts as a router) for cluster heads in a specific region. For a specific service, additional application specific roles could be deployed. For example, for monitoring and data aggregation, roles such as sensing collaborator, sensing coordinator or data aggregator may also be assigned.

In case of node or event mobility, specific roles could be defined that include route manager and track manager, respectively.

2.4.2 Role Identification: Rules, Metrics, Utilities

The identification of roles can be based upon a specific rule that incorporates either an individual or a weighted set of performance metrics expected of that role. Since role formulations are in essence a priori knowledge based (both domain and application specific) requirements for a certain task, their criteria for selection are also implicit in the role expression. In other words, the conditions needed for a role assignment, reassignment (or repair), and removal are part of its role specification (Tambe, 1997). Usually, distributed systems use a set of rules where each rule specifies a certain criteria in terms of a desired performance metric. These rules are then ordered as per the relative importance for a specific role. At times all the rules can be grouped into a single rule representing a weighted set of metrics. In this case, the weights specify the importance of a certain performance criteria over another for the selection of a specific entity for a role.

Utility is one of the unifying concepts from economics, game theory, and operations research that allows an entity (across all application domains) to somehow internally estimate the value (or cost) of executing an action. It is variously called fitness, valuation, and cost. With multi-robot research, the calculation of utility can vary from sophisticated planner-based methods (Botelho and Alami, 1999) to simple situational sensing-based metrics (Gerkey and Matarić, 2002). In the RoboCup domain, it is common to compute utility as the weighted sum of several factors, such as distance to target position, distance from the ball, and whether the team is on offense or defense. A similar utility estimation based on appropriate application

domain specific metrics is also carried out in every task or role-allocation algorithms for wireless sensor networks (Byers and Nasser, 2000).

2.4.3 Role Assignment (RA) techniques

The assignment of roles benefits from global information, however, due to the real-time communication constraints, role assignment problems need to be solved in a distributed manner with local information. In general, solutions to classical coverage, clustering, and in-network data aggregation problems in wireless sensor networks are essentially distributed role assignment problems. The coverage problem in wireless sensor networks can be modeled as the "art gallery problem" (Rourke, 1987), which can be stated as determining the minimum number of guards required to cover the interior of an art gallery. Accordingly, it has been shown to be an NP-hard problem for wireless sensor networks as well (Slijepcevic and Potkonjak, 2001). Another well known problem related to sensor networks is the "facility location problem" (Drezner and Hamacher, 2004), where a set of facilities needs to be optimally placed in order to minimize transportation costs. This problem is also similar to covering an area in a WSN with the minimum set of sensor nodes for both sensing and forwarding.

The typical communication paradigm for wireless sensor networks considers a group of nodes sensing the environment, collaborating with each other to discard spurious events, and then forwarding and aggregating the genuine sensing data enroute in a multi-hop manner toward the remote sink or base station. This underlying sensor network paradigm entails nodes in the network playing different roles per local requirements. These requirements manifest from both the application and the network. Applications for wireless sensor networks require sensing, monitoring,

and tracking services to be optimized along with network lifetime. These sensing related services are resolved as role assignment for both optimal sensing coverage and data aggregation. The network related requirements include the application specific topological organization of the nodes in the system. This role-based structuring of the underlying network imposes specific requirements from routing and network maintenance. In general, energy efficiency is the single most important requirement for role assignment in an ad hoc wireless sensor network.

Role assignment assumes that nodes are aware of the generic role specification and the criteria for their selection. This is followed by an initial static role assignment for neighbor discovery. In this neighbor discovery process, nodes exchange their properties. These properties are part of the role specification discussed earlier. The neighbor discovery process is run periodically and also on-demand with nodes joining and leaving the network. This is then followed by a role assignment phase, which assigns roles for meeting a specific application requirement. Role assignments are re-evaluated proactively (that is periodically) or reactively in response to changes in node properties or sensing event dynamics. Both distributed and centralized solutions exist for role assignment.

The role-based hierarchical self-organization (RBHSO) (Kochhal et al., 2003) protocol employs the use of rules that elicit the list of metrics a node should recursively compete with in order for it to be locally dominating among its peers for a particular role. These rules are used to efficiently self-organize a network hierarchy with specific assignment of roles (or tasks) to sensors based on their physical wireless connectivity and sensing characteristics (Kochhal et al., 2004). It extends the hierarchical connected dominating set (CDS) construction algorithm to set up a hierarchical self-organization architecture that establishes a network-wide infras-

tructure consisting of a hierarchy of backbone nodes, and sensing zones that include sensor coordinators, and sensing collaborators (or sensing zone members).

TinyCubus (Marrón et al., 2005) incorporates a role specification algorithm defined by a generic specification language. A set of rules defines the necessary conditions for the assignment of roles. To assign roles to sensor nodes, it is necessary to take into account role specification and sensor node properties. For TinyCubus, role specification is a list of rule pairs. For each possible role, the associated rule specifies the conditions for assigning this role. All nodes in the network have a copy of the same role specification. An instance of the role assignment algorithm is executed in each node of the network, triggered by property and role changes on nodes in the neighborhood, the algorithm evaluates the rules contained in the role specification. If a rule evaluates to true, the associated role is assigned. Similar techniques to generalize the role-assignment process have been proposed in (Frank and Romer, 2005).

DFuse (Kumar et al., 2003) is a framework for distributed data fusion that considers the problem of maximizing network lifetime for data aggregation using role assignment. It uses a tree-based organization in which parent nodes with higher energy act as data collectors. The network is optimized periodically through role migration. The algorithm has three main phases: (1) naïve tree building where the root node urges to its neighbors to create sub-trees. This process is repeated recursively until tree build stops at the leaf nodes (data producer nodes), (2) optimization phase where every node hosting a fusion point role is responsible for either continuing to play that role or transferring the role to one of its neighbors. This decision is taken solely by the fusion node based upon local information, and (3) maintenance phase which repeats periodically and executes the optimization phase

to allow nodes to change their role in accordance with change in node or network properties.

The research presented in (Bhardwaj and Chandrakasan, 2002) generalizes bounds for data aggregation in sensor networks with specified topology and source movement. These bounds were derived by employing the formalism of feasible role assignments (FRAs). The idea is based upon the fact that there are only a finite set of assignments of roles to nodes that allow sensing in a non-redundant manner. A computationally intensive offline linear programming technique was used to finalize an FRA among all possible FRAs such that network lifetime is maximized. However their technique is based on a class of role assignment problems that permit a transformation to linear programs based on network flows that can be solved in polynomial time. It is therefore important to realize that not all role assignment (RA) problems can be similarly transformed. However it is applicable to several practical RA problems for pure routing, non-hierarchical and constrained hierarchical aggregation, multiple or moving sources, and sources with specified trajectories.

A similar centralized solution presented in (de Souza and Mateus, 2006) proposes solutions maximizing system lifetime for data aggregation in WSNs. This work presents an optimization model and a genetic algorithm for solving coverage and routing by way of role assignment. This joint optimization problem is modeled as a mixed-integer linear programming problem, that can be solved through optimization software available in the market (CPLEX). A heuristic technique based on genetic algorithms was also proposed as an alternative. In (Dasgupta et al., 2003), topology-aware role placement for maximizing system lifetime for monitoring applications has been proposed. The RA algorithm is based on a distributed implementation of the force-directed/potential-field based approaches in robotics/graph drawing (Battista

et al., 1999).

2.5 Summary

In summary, currently the research in wireless sensor networks is oriented toward a cross-layer programming abstraction that allows programmers to perform resource control for expressing energy-efficient network services. The concept of roles allows a programmer to deal with the assignment and scheduling of tasks to nodes or group of nodes in the network. We have proposed the Role-based Hierarchical Self-Organization as the underlying network architecture that allows programmers to construct an application specific architecture based on a newly developed sensing metric known as sQoS.

Programmers will find it difficult to trade off ease of programmability with the difficulty of using low-level cross-layer controls for achieving the desired performance. This is because of the multi-domain application of wireless sensor networks. Also, with the availability of a multitude of sensor network platforms (both hardware and software) and a variety of algorithms in the literature for efficient performance of several sensor network services, it is unmanageable even for a domain expert to achieve programming tractability. Under these conditions, it is expected that for every application domain and sensor platform, a set of templates can be standardized for a variety of network sizes and application performance levels. This should relieve programmers of the technical domain expertise, and will also give them a starting template of a recommended algorithm and parameters to tweak for a specific application. Future research efforts will be aimed at the standardization of service templates for heterogeneous sensor networks.

Chapter 3 – Role-based Hierarchical Self Organization for Wireless Sensor Networks

Recently research efforts in wireless sensor networks have focussed on ideas involving the possibility of coordinating the activities and reports of a large collection of tiny sensor devices. Efficiently self-organizing a network hierarchy with specific assignment of roles (or tasks) to sensors based on their topological wireless connectivity and sensing characteristics is an important and challenging problem. In this chapter, we extend the hierarchical connected dominating set (CDS) construction algorithm, proposed by Jie Wu, to develop our role-based hierarchical self organization (RBHSO) algorithm for wireless sensor networks. The resulting self-organized sensor network establishes a network-wide infrastructure consisting of a hierarchy of backbone nodes, and sensing zones that include sensor coordinators, and sensing collaborators (or sensing zone members). We demonstrate the effectiveness of our design through simulations.

3.1 Self-organization preliminaries

In this section, we discuss the elementary concepts of network self organization as applicable to the regime of wireless sensor networks. These concepts serve as the necessary foundation for understanding developments in the area of sensor network self organization. For ease of understanding, we have classified these basics into sensing and network organization concepts. We also formalize the necessary steps (or protocols) that fall under the unified umbrella of sensor network

self-organization.

The *sensing concepts* also known as the *sensing phenomenon* (Kochhal et al., 2004) are concerned with the characteristics of the sensors, the events to be detected, and their topological manifestations both in the spatial and the temporal domains. For example, it is obvious that sensors in close proximity to each other should have correlated readings. A temporal dual of this observation implies that sensor readings among neighboring sensors also have some correlation within some nearby time intervals. In addition to supporting the properties associated with the sensing phenomenon, it is also necessary to support hierarchical event processing in order to have an incremental comprehensive global view of an area of deployment at different levels of the self-organized network hierarchy.

As mentioned earlier, self-organization involves abstracting the communicating sensor nodes into an easily controlled network infrastructure. Cluster, connected dominating set (CDS), tree, grid, or mesh-based organizations are typical. We provide some insights into these organizations for use in wireless sensor networks.

3.1.1 Elementary networked sensing concepts

The sensing phenomena mentioned earlier relate to the natural property of sensors sensing events collaboratively as well as individually in a group. Figures 3.1, 3.2, 3.3, and 3.4 illustrate these sensing concepts of wireless sensor network organization for target detection and/or tracking. In the following discussion, we use the terms sensing groups and sensing zones interchangeably.

Figure 3.1 illustrates that the sensing capability of sensors sensing events collaboratively and/or individually in a group depends essentially on the sensitivity of the sensors with respect to the target event. The sensitivity of a sensor diminishes with

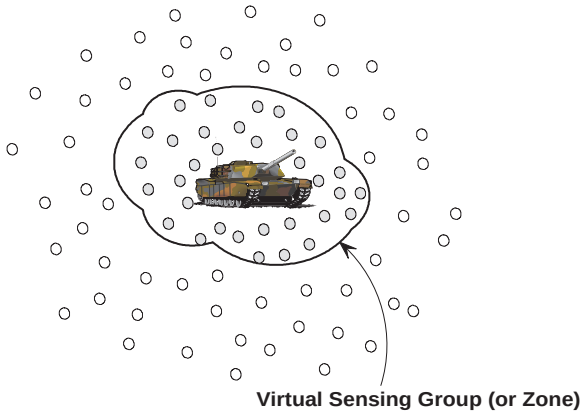

Figure 3.1: Spatial group sensing concept.

increasing distance of the sensor from the target. This sensitivity can be characterized theoretically by sensor models that are based on two concepts. One is that the sensing ability (*coverage*) diminishes with increasing distance[a]. Second is that noise bursts diminish the sensing ability but this effect of noise can be minimized by allowing sensors to sense over longer time periods (more *exposure*). Several algorithms based on the above sensitivity model have been developed that formulate the exposure and coverage properties of sensor networks. These algorithms use traditional computational geometry based structures like the Voronoi diagram and the Delaunay triangulation (Meguerdichian et al., 2001b) (Meguerdichian et al., 2001a) to compute sensing coverage and exposure. However, distributed versions of these algorithms are challenging and computationally intensive, and hence are impractical for use during the initial network organization phase.

In general, self-organization protocols usually employ the concept of *redundant sensing* in order to account for fault tolerant sensing in the presence of environmental vagaries. By redundant sensing, we mean that an observation of the presence of a nearby target event (i.e. a tank[a] in figure 3.1) should be supported not only

by one sensor but also by a group of neighboring sensors (Varshney, 1996) (Dyck, 2002). This requires selection of a group of neighboring sensors that can take sole responsibility of any event appearing within their region or group. The selection of sensors to form such a group requires quantifying relative proximity distances of each and every neighboring sensor. It also requires an intelligent discrimination between near and far sensors to avoid grouping sensors from distant locations.

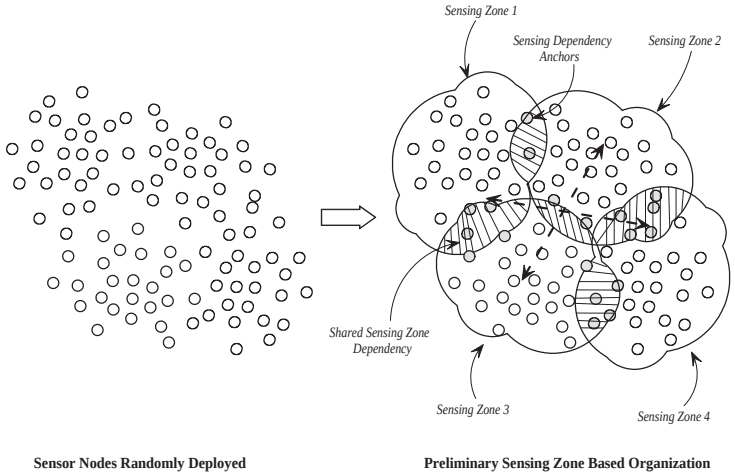

Figure 3.2: Sensing-group dependency concept.

Figures 3.2 and 3.3 illustrate the sensing zone dependency situation during tracking by sensing zones formed around a mobile enemy tank. Specifically, here we are discussing an initial network organization that statically forms sensing-zones in anticipation of the occurrence of any future event. In the case of a random sensor deployment scenario it is not possible to precisely control and place sensors so that

[a]The sensing range may depend upon the dimensions of the observed target, e.g. a seismic sensor can detect a tank at a greater distance than it can detect a soldier on foot. For ease of discussion, we assume the sensing range to be the same for targets of similar dimensions (Slijepcevic and Potkonjak, 2001). However, in general for an application specific sensor deployment, nodes are assumed to be pre-configured for desired targets in terms of their sensing signatures or readings. In the case of on-demand target detection and tracking, the application is free to provide respective target sensing signatures in its queries.

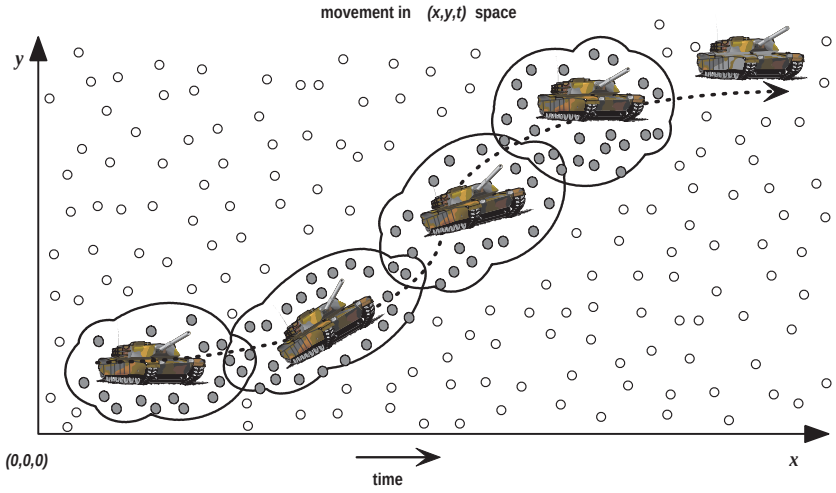

Figure 3.3: Tracking a mobile tank around neighboring sensing groups.

they end up in groups having no overlap with neighboring sensor groups. This means that although an attempt was made to form stand-alone sensing groups (or zones) that independently take responsibility for detecting and tracking events, there are some overlapping regions where collaboration among neighboring sensing groups may be needed. However, the boundary nodes in each region can also serve as anchors for tracking events moving from one neighboring region to another. This is essentially a dichotomous scenario because on one hand we need independence between neighboring sensing zones but on the other hand, we also want to efficiently track events moving across neighboring sensing zone boundaries. An event monitoring and tracking algorithm that runs on top of such a self organized network would have to analyze this dependency and utilize it to its best advantage. This can be done by either identifying neighboring dependent sensing zones and allowing collaboration among them for events moving around their neighborhood or tracking applications can dynamically specify an on-demand incremental reorganization of a new sensing zone around the moving event as it crosses the old sensing-zone

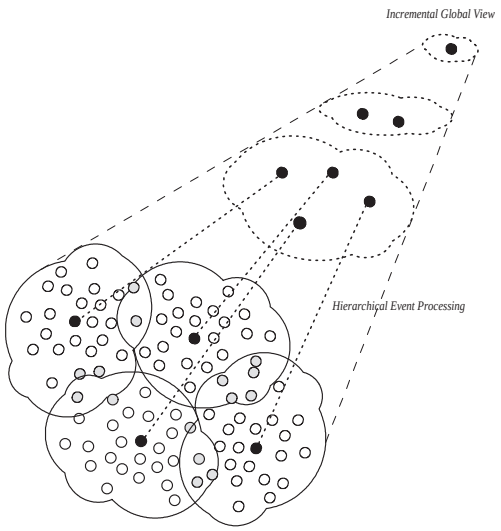

Figure 3.4: Hierarchical event processing for incremental global view.

boundaries. The *EnviroTrack* project (EnviroTrack: an Enviromental Programming Paradigm for Sensor Networks) (Abdelzaher et al., 2004) (Blum et al., 2003) that were pursued at University of Virginia as a proof-of-concept implementation that supports such application specified sensing-group network (re)organization for tracking in a physical environment.

Figure 3.4 illustrates event processing at various levels of a hierarchical sensor organization. It can be seen that as we go higher in the hierarchy fewer nodes are involved in event processing. However, as we go up the hierarchy, we also lose detail about the event(s). This is because wireless communication is an overhead in terms of draining energy. Also, due to the small form factor of the sensors, memory is also a crucial resource and hence a lot of information cannot be maintained by an individual sensor or a small group of sensors. If we assume that the sensors selected for the upper levels of the hierarchy are powerful in terms of both communication energy as well as memory, the problem is still not resolved, due to scalability issues.

However, any feedback from the sensing application about the granularity of monitoring would help in reducing overheads in information gathering and processing. In any case, hierarchical processing motivates the concept for distributed gathering, caching, and processing of sensing events where certain nodes in the hierarchy are assigned appropriate roles (Kochhal et al., 2004) (Kochhal et al., 2003) according to their capability in the current network organization.

3.1.2 Elementary network organization concepts

The primary objectives of this section are to categorize several elementary network organization architectures and discuss some relevant approximation algorithms that can be extended by self organization protocols.

Figure 3.5 shows a simple classification of various network architectures that can be employed by self-organization protocols. This classification is not complete, as there could be certain combinations of different network architectures. However, it provides the principal categories under which several current implementations could be studied and analyzed. Self-organization protocols could be either proactive or reactive. In other words, protocols could organize the sensor network statically in preparation for any future event or they could dynamically configure the network around any current event of interest. Additionally, self organizing protocols could pursue either a difficult to maintain hierarchical manifestation of the above network architectures or they could simply satisfy requests with their corresponding flat manifestations. Figure 3.6 provides a visual blueprint for elementary network architectures such as the chain, tree, spine, virtual grid, and role-based virtual regions. We will then discuss the typical algorithmic aspects of these network formations for general wireless ad hoc networks. This facilitates easier comprehension

and analysis of those sensor network organization protocols that extend or modify these algorithms in order to meet various sensing application requirements. However there are certain concepts that are common across all these network organizations. In all these organizations, nodes adopt certain performance metrics for selecting neighbor(s) in their local network formation heuristics. These performance metrics could be minimum distance, minimum energy, minimum transmission power, maximum/minimum node degree, delay, bandwidth, etc. Some of these metrics may be used collectively in some particular order (depending upon priority) to break ties among several eligible competitors. In order to have an optimally ideal neighbor selection scheme for self-organization, nodes may require complete global state information of the network. However, in an ad hoc network, nodes that execute distributed algorithms for localized self-organization do not have the luxury of gathering, maintaining, and using complete network knowledge. As mentioned earlier, this is because there are tradeoffs among storage capability, communication costs, computational capability, and time to completion. This effectively results in nodes maintaining network state information for only 2 to 3-hop neighbors. Using this information, nodes execute local decisions to select neighbors to form a global self-organized network.

The chain based organization is one of the simplest ways of organizing network communication, where nodes farther from the base station initiate chain formation with their nearest neighbor. The idea is to gather and fuse all the data from every node by forming a chain among them. A leader is then selected from the chain to transmit the fused data to the basestation. However, building a chain to minimize its total length is similar to the traveling salesman problem, which is known to be intractable. A greedy chain formation algorithm (S. Lindsey, 2002), when pursued

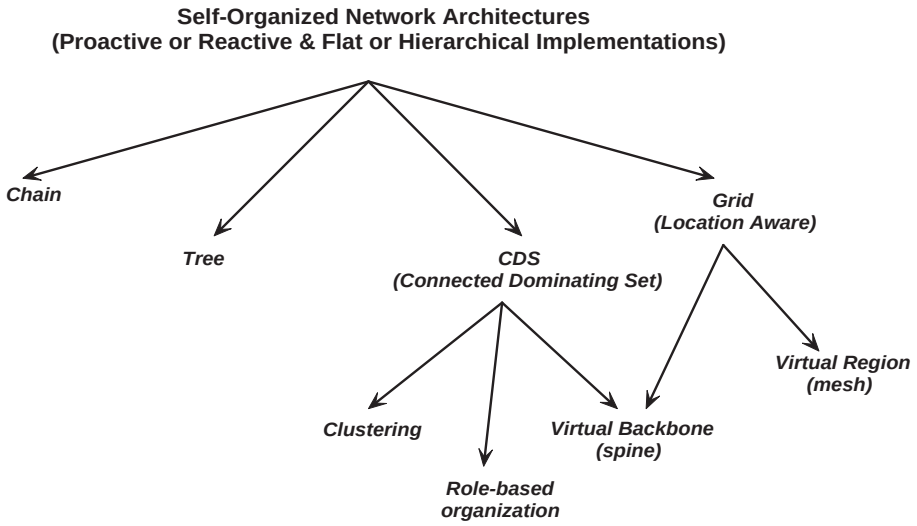

Figure 3.5: Self-organized network architectures.

recursively for every node, results in a data gathering chain oriented toward the base station. As we will discuss later, self-organization algorithms for sensor networks usually include certain sensing metrics too in order to form an optimal organization that is efficient from both the sensing (Kochhal et al., 2004) (Inanc et al., 2003) and the networking perspectives (Singh et al., 1998) (Salhieh and Schwiebert, 2002).

The tree type of network formation is similar to the chain and can be considered as an extension of the chain based mechanism. Tree type network organizations, utilize the multipoint connectivity nature of the wireless medium, where one source can be heard simultaneously by several nearby receivers that act as its children. If both the sender and receiver scheduling are made collision-free, then a tree-based network organization can support both *broadcast or multicast* (i.e. dissemination of information from a central node) and *convergecast* (i.e. gathering of information toward a central node) communication paradigms across all application domains (Annamalai et al., 2003). A considerable amount of research work is available for

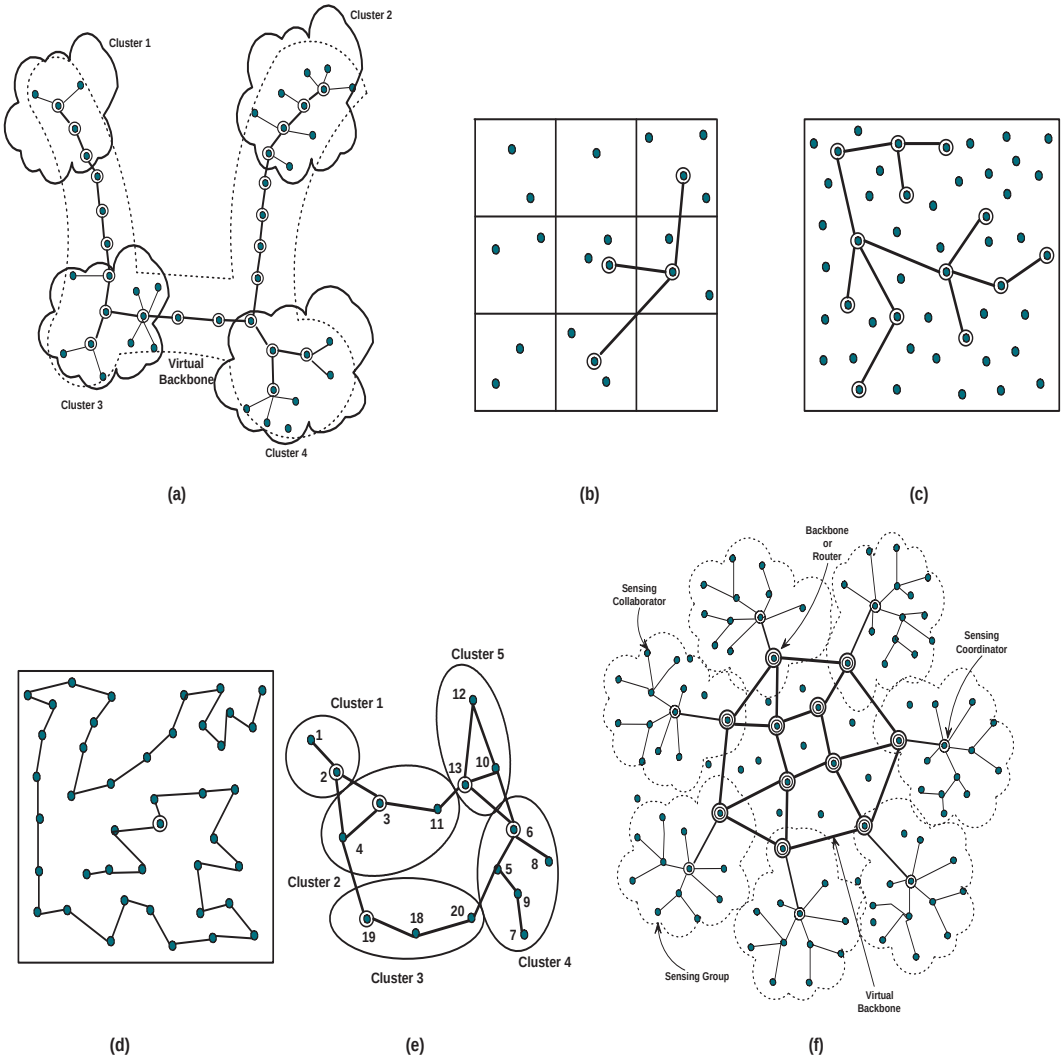

Figure 3.6: Example network organizations for (a) spine, (b) virtual grid, (c) tree, (d) chain, (e) clustering, and (f) role-based virtual zones.

constructing multicast trees (Ballardie et al., 1993) (Estrin et al., 1998) for dynamic wireless ad hoc networks (Adelstein et al., 1999) (Gupta and Srimani, 1999). Algorithms for generating multicast trees typically must balance the goodness of the generated tree, execution time, and the storage requirements. Several cost metrics such as delay, communication costs, etc. are used to generate cost-optimal multicast trees. However, this gives rise to the well-known *Steiner Tree problem*, which is NP-complete (Winter, 1987). Instead heuristics are used to generate "good" rather than optimal trees. This is still an active area of research.

Cluster-based organizations (refer to figure 3.5 (e)) partition the entire network into groups called clusters. Each cluster is formed by selecting some nodes based upon some quality metric such as connectivity or distance (Chen et al., 2002) (Amis et al., 2000) (Steenstrup et al., 2000) as cluster members and a group leader, known as the clusterhead, is also selected using some metric such as the node that has maximum energy to manage that cluster. These clusterheads, when connected, form a virtual backbone or spine as in figure 3.6(b) (Sivakumar et al., 1998) (Chen et al., 2001) or a connected dominating set (*CDS*) of nodes. Related to clustering is the problem of finding a minimum connected dominating set (*MCDS*) of the nodes, which is NP-complete. An *MCDS* satisfies two properties: (1) each node is either a backbone node or is one hop from a backbone node, and (2) the backbone nodes are connected. There are several approximation algorithms (Guha and Khuller, 1996) available in the literature that engineer virtual backbone based network configurations satisfying the *MCDS* properties. Of particular importance is Wu's (Wu and Li, 1999) (Wu, 2002) distributed and localized algorithms for constructing a hierarchical connected dominating set. This algorithm is inherently distributed and simple in nature. Ideally, it requires only local information and a constant number

of iterative rounds of message exchanges among neighboring hosts. The algorithm for CDS formation involves a dominating set reduction process and some elimination rules based on quality metrics that are executed by nodes locally to identify their dominating neighbors. The dominating set reduction process, when executed recursively by an already existing set of dominating nodes, produces a domination hierarchy. Dominating nodes at any level of the hierarchy can serve as local network coordinators for nodes in the next lower level of the hierarchy. Figure 3.6(f) shows a role-based hierarchical CDS organization of nodes, where the lower level of nodes form the cluster, whereas upper levels of dominating nodes can be used for routing or hierarchical information processing. Accordingly, nodes at every level of the domination hierarchy assume respective roles in the network depending upon the quality metrics used for role selection at that level (Kochhal et al., 2004) (Kochhal et al., 2003).

Finally, if nodes have location information (e.g., using GPS), then virtual grid based organizations (Xu et al., 2001) (Ye et al., 2002a) typically configure the network by partitioning the area of deployment into uniform grids also known as a mesh. From each grid, a dominating node is chosen using some selection rules. Dominating nodes from each grid, when connected, form a virtual backbone, which can then be used for gathering or forwarding information from one geographic region to another. The grid-based organization allows an implicit and simple naming system by having grids (regions) in the network be addressed by their relative geographic locations. Thus, it avoids the complex and non-scalable address generation mechanisms for individual nodes that are densely deployed in a very large area. However, the efficiency of such an organization depends critically on location accuracy and network partitioning schemes.

3.1.3 Steps to sensor network self organization

Self-organization or self-configuration is one of the basic initial steps toward an ad hoc deployment of wireless sensors. The network deployment as mentioned earlier may be done deterministically or randomly. In any case, the objective is to have nodes discover their neighbors, establish their positions, and form an easily manageable network architecture. All these self organization activities have to be performed in a localized and distributed manner with high energy efficiency and little or no communication overhead. Moreover, the self organized network infrastructure should be *adaptive* and *resilient* to be easily reorganized with respect to the ultimate traffic pattern that may run on top of it.

Following the self-organization steps as extended from (Subramanian and Katz, 2000) (Chevallay et al., 2002) forms the complete basis for any self-organization algorithm for ad hoc wireless sensor networks:

1. *Network Discovery or Initialization Phase*
    (a) Each node discovers its set of neighbors,
    (b) Depends on communication transmission radius (Tx_{max}), and
    (c) Random or deterministic initial channel scheduling for neighbor discovery.
2. *Coarse Grained Estimation Phase*
    (a) Location estimation and
    (b) Time synchronization.
3. *Organizational Phase*

(a) Formation of a hierarchical or flat network organization with the help of local group formations,

(b) Performing group reorganization if necessary,

(c) Generation of addresses for nodes,

(d) Generation of the routing table at every node,

(e) Generation of broadcast or multicast trees and graphs within a group,

(f) Merger of broadcast trees and graphs when groups are aggregated to form hierarchical networks,

(g) Establish medium access control schemes both for intra-group and inter-group communication, and

(h) Establish key setup schemes for secure communication.

4. *Maintenance Phase*

(a) Active or passive monitoring e.g., by "*I am Alive*" messaging,

(b) Network quality evaluation schemes such as connectivity and sensing coverage,

(c) Maintenance of routing tables,

(d) Maintenance of broadcast infrastructure,

(e) Fine-grained tuning of network parameters such as location and network time,

(f) Topology control schemes to maximize network throughput and spectral reuse or network capacity (Chen et al., 2001) (Schurgers et al., 2002) (Santi, 2003), and

(g) Energy conservation schemes for increased network longevity using dynamic node scheduling (Tian and Georganas, 2002) (Xu et al., 2001).

5. *Self-Reorganizing Phase*

(a) Redeployment leading to discovery of new node neighbors and

(b) Fault detection and recovery schemes under node or link failure and group partitions.

In general, the steps listed above can also be considered as services provided by self-organization protocols for wireless sensor networks. This means that some of these services may be optional whereas some are fundamental to any algorithm that self-organizes the sensor network. Thus, steps 1 and 3 are necessary. On the other hand, steps 2, 4, and 5 are optional and can be developed separately. Location estimation protocols are generally referred to as self-configuring localization protocols. Network time synchronization can be considered orthogonal and implemented separately without regard to any specific network design or architecture. Similarly, the maintenance phase can be implemented separately as a suite of network management protocols.

3.2 Design Philosophy

Wireless sensor network operations include data discovery, which is achieved by way of sensing application specified target events. Additionally, the sensor network needs to process this information in a distributed manner and then forward it to an interested data sink or a remote base station. These sensor network tasks can be managed individually by a sensor node or they may be collaborated upon by

several nodes simultaneously. An intuitive analysis of the sensor network activities leads to mapping tasks to roles as follows:

1. Sensing Collaborator
2. Sensing Coordinator
3. Routing or Backbone Nodes

Since all sensor nodes in the network are deployed to collaboratively sense target events, all nodes assume the role of a sensing collaborator. However, some of these nodes are also requested to assume the role of either a routing node or a sensing coordinator. The routing role, as the name suggests, supports a network-wide routing functionality for both application specific sensing queries and the sensing data gathered by the sensors. A sensing application may need to query for a target event in a certain interested region of sensor network deployment. On the other hand, target events sensed by some sensors in a certain region may need to be solicitepd by some other sensors acting as data sinks or sensor coordinators. These sensor coordinators not only take the responsibility of coordinating the sensing activities in their neighboring region (also known as a sensing zone) but also aggregate and forward the information to any remote data sink or the base station. The task of coordination is not a simple one and it is also not a short term job. In order to provide instantaneous sensing and reporting capability (dependent upon sensing applications) each sensor coordinator may need to systematically rotate its responsibilities transparently among neighboring nodes without much communication overhead. A hierarchical network organization would also be needed to provide scalability for a dense deployment of a large number of sensors.

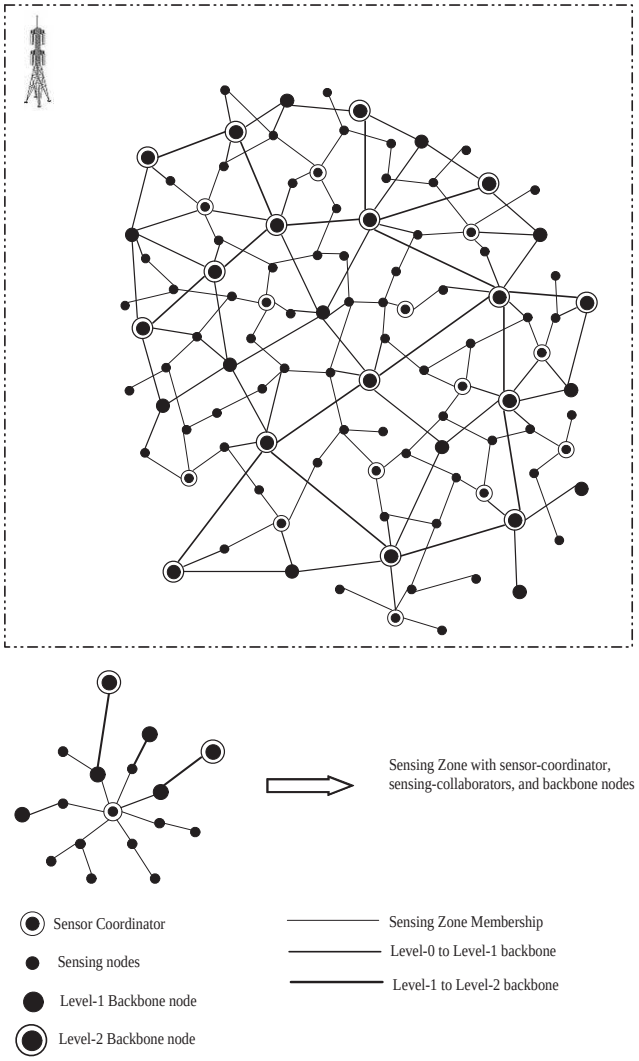

Figure 3.7: Role-based Hierarchical Self-Organization for WSNs

Figure 3.7 illustrates these design principles of our protocol. A two-level CDS hierarchy is shown to support routing infrastructures throughout the network. One advantage of having multiple levels of hierarchy is that as the hierarchy increases fewer nodes are involved in routing, which leads to paths with fewer hops within the network. Depending upon the requirements of the sensing application as well as the topology of the sensor network deployment, one may be able to provide certain levels of guarantees with respect to routing queries or routing data to the base station. Thus, it would be desirable that the reorganization phase of the self configuring algorithm preserve the lifetime of these higher level hierarchy nodes and hence preserve the capability of providing prompt delivery of services in the face of changing sensor network traffic patterns.

In the next section, we will be briefly outlining the CDS construction algorithm proposed in (Wu and Li, 1999) (Wu, 2002).

3.2.1 CDS based Network Organization

Cluster-based organizations partition the entire network into groups (or clusters). Each cluster is formed by selecting some nodes based upon some quality metric (say connectivity or distance) (Chen et al., 2002) (Amis et al., 2000) as cluster members and a group leader (known as the cluster-head) is also selected (using some metric, say maximum energy) to manage that cluster. These cluster-heads, when connected, form a virtual backbone (or spine) or a set of connected dominating nodes. Related to clustering is the problem of finding a minimum connected dominating set (*MCDS*) of the nodes. An *MCDS* satisfies three properties: (1) each node is either a backbone node or is connected (one hop) to a backbone node, (2) the backbone nodes are connected and (3) minimum set of nodes are involved

to form such a virtual backbone. Ephremides et al. (Baker and Ephremides, 1981) first tried to introduce a backbone-like structure in wireless networks. Guha and Khuller (Guha and Khuller, 1996) firstly used the MCDS problem in general graphs to model the problem of computing a minimum size virtual backbone in heterogeneous wireless networks. Since a smaller size virtual backbone is expected to have less control overhead in terms of messages and reduced interference, the size of the CDS is considered as one of the major quality criteria in the literature. Since computing the MCDS is a well-known NP-hard problem, all of the existing work propose approximation algorithms.

In our proposal, we will be using the distributed localized algorithm for constructing a hierarchical connected dominating set (CDS) presented in (Wu and Li, 1999) (Wu, 2002). Our main reason for selecting this algorithm is its inherent distributed and simple nature. Ideally, it requires only local information and a constant number of iterative rounds of message exchanges among neighboring hosts. The algorithm for CDS formation involves two processes, the marking process and the dominating set reduction process. We also assume the following network model.

Network Model

We represent the ad hoc wireless network by a simple graph $G = (V, E)$, where V represents a set of wireless nodes and E represents a set of edges. An edge between host pairs (v, u) indicates that both hosts v and u are within each others, wireless transmitter ranges and the wireless links are bidirectional. We assume that all the wireless nodes are homogeneous, i.e., their wireless transmitter ranges are the same. In other words, if there is a edge $e = (v, u)$ in E, it indicates that u is within v's range and v is within u's range. Thus, the corresponding graph is an

undirected graph also known as a unit graph, in which connections to hosts are determined by their geographical distances.

Marking Process

The marking process as described in (Wu and Li, 1999) (Wu, 2002) is a localized algorithm in which hosts interact with others only in a restricted vicinity. Each host performs exceedingly simple tasks such as maintaining and propagating information markers. Collectively, these hosts achieve a desired global objective, i.e., finding a small connected dominating set. The marking process marks every vertex in a given connected and simple graph, $G = (V, E)$. $m(v)$ is a marker for vertex $v \in V$, which is either T (marked) or F (unmarked). Initially, it is assumed that all the vertices are unmarked and that each vertex v has its open neighbor set as $N(v) = \{u \mid (v, u) \in E\}$. The marking process can thus be summarized as follows:

1. Initially, assign marker F to each v in V.

2. Each v exchanges its open neighbor set $N(v)$ with all its neighbors.

3. Each v assigns its marker $m(v)$ to T if there exist two unconnected neighbors.

In the example depicted in figure 3.8, $N(u) = \{v, y\}$, $N(v) = \{u, w, y\}$, $N(w) = \{v, x\}$, $N(y) = \{u, v\}$, and $N(x) = \{w\}$. After step 2 of the marking process, vertex u has $N(v)$ and $N(y)$; v has $N(u)$, $N(w)$, and $N(y)$; w has $N(v)$ and $N(x)$; y has $N(u)$ and $N(v)$; and x has $N(w)$. Based on step 3, only vertices v and w are marked T.

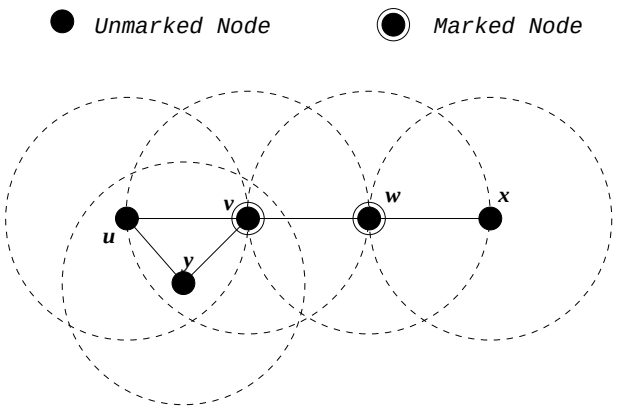

Figure 3.8: Initial Marking Process on a sample ad hoc wireless network[b]

Dominating Set Reduction Process

In order to reduce the connected dominating set (CDS) generated from the marking process, two rules are proposed. Assuming that each vertex v in G' is assigned a distinct ID, $id(v)$, it then calculates its closed neighbor set $N[v]$ as $N[v] = N(v) \bigcup v$.

Rule 1: Consider two vertices v and u in G'. If $N[v] \subseteq N[u]$ in G and $id(v) < id(u)$, change the marker of v to F if node v is marked. I.e. G' is changed to $G' - v$.

Rule 2: Assume that u and w are two marked neighbors of vertex v in G'. If $N(v) \subseteq N(u) \cup N(w)$ in G and $id(v) = \min\{id(v), id(u), id(w)\}$, then change the marker of v to F.

In Figure 3.9(a), since $N[v] < N[u]$, vertex v is removed from G' if $id(v) < id(u)$ and vertex u is the only dominating node in the graph. In 3.9(b), since $N[v] = N[u]$, either v or u can be removed from G'. To ensure one and only one node is removed, the node with the smallest ID is removed. Finally, in figure 3.9(c), $N(v) \subseteq N(u) \cup N(w)$ applies. If $id(v) = \min\{id(v), id(u), id(w)\}$, vertex v can be removed from G' based on Rule 2.

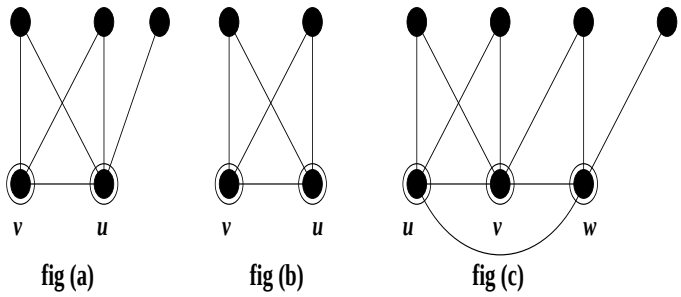

Figure 3.9: Three examples of dominating set reduction[b]

In (Wu and Li, 1999) (Wu, 2002), the above rules were extended to include a combination of metrics like energy level (EL) and node degree (ND) along with ID to break ties. In this chapter, we will be discussing our proposed sensing-based metrics, which can also be incorporated into the rules for dominating set reduction.

Hierarchical Dominating Sets

The dominating set reduction process can be reapplied on an already reduced dominating set of nodes to generate another set of dominating nodes. This process can be repeated until no further reductions are possible. (Wu and Li, 1999) (Wu, 2002) also introduce the concept of dominating ratio (DR), which is the ratio of the size of the resultant dominating set to the size of the original network. Clearly, $0 < DR \leq 1$. A small DR corresponds to a small dominating set. Unfortunately, the minimum dominating ratio is not known a priori. For our protocol, the sensing application can specify the desired DR and the self-organization algorithm will try in a best-effort manner to get a reduced hierarchy matching that DR, although it cannot be guaranteed, as it depends upon the network topology whose deployment cannot be controlled precisely.

[b]Example figure reproduced with permission from (Wu and Li, 1999) (Wu, 2002).

3.2.2 Sensing Attributes or Metrics

Sensing Model

A sensor is a device that produces a measurable response to a change in a physical condition, such as temperature, light, voice, or magnetic field. We assume the same sensing model as that of (Meguerdichian et al., 2001b) (Meguerdichian et al., 2001a). We also assume that the sensing region of a sensor is a circle with the sensing range[b] specified as S_R distance units.

Sensing Coverage Approximation

Figure 3.10 shows three sensors (say, seismic sensors) reporting the detection of a target event (say, an enemy tank) in a battlefield scenario. Since the tank is at a variable sensing proximity or distance (also denoted as SPV) from each of the sensors, the degree of fault tolerance sensing (denoted as CSD) for this event is proportional to the cumulative proximity of the three sensors to the target event[b]. In order to comprehend the maximum cumulative fault tolerant sensing capability of these three sensors, it may be necessary to calculate the amount of shared coverage overlap between these sensors. The sensing coverage is approximated as a circle with sensing range as its radius. Thus, the problem of finding the cumulative sensing coverage is transformed into finding the overlapping sectors between the neighboring sensors, which is a complicated approximation as discussed earlier. We simplify this approximation by dividing the circular sensing area of each sensor into square sensing cells. The dimension of the sensing cell determines the closeness of the coverage approximation. The sensing cell dimension (denoted as d) is also known as the *application specified sensing accuracy*. We assume that the three sensors, S_1, S_2, and S_3, know the positions of each other. Thus calculating combined sensing

coverage would amount to finding the common overlapping sensing cells among the neighbors and also subsequently updating these sensing cell's cumulative sensing proximity values (denoted as $CSPV$) by accounting for the relative distance of the neighboring sensors to the cell(s) in question. In other words, $CSD_{S_1,S_2,S_3} = K \times f(3, d, overlap)$, where K is some sensing constant for the sensors (in our case, $K = 1$), and f is the function that calculates the cumulative sensing degree (CSD) by accounting for the number of cooperating sensors, the sensing cell dimension (d), and the cumulative sensing proximity value ($CSPV$) of the overlapping sensing cells between them. Figures 3.11 and 3.12 illustrate the approximation used in our sensing coverage calculations. In the next sections, we will be defining in detail the SPV, $CSPV$, and CSD sensing parameters.

Sensing Proximity Value (SPV)

SPV for a sensing cell denotes how close that cell is to a particular sensor. The SPV may vary from the best value of 1 to some max value, say SPV_{max} (dependent upon sensing range S_R). *The lower the value of SPV for a cell, the better its sensing performance or sensitivity.* For calculating the SPV of a sensing cell i for a sensor node, say n, we need the location of the sensor n i.e. (x_n, y_n), the sensing range, S_R, and the application specified sensing accuracy, d. We calculate the minimum distance between the sensor n and the center of its closest sensing cell and denote it as d_{csmin}.

$d_{csmin} = d/\sqrt{2}$

[b]The sensing range may depend upon the dimensions of the observed target, e.g. a seismic sensor can detect a tank at a larger distance than it can detect a soldier on foot. For ease of discussion, we assume the sensing range to be same for both the tank and the soldier (Slijepcevic and Potkonjak, 2001). We can modify our self organizing algorithm based on the sensing range for a given application.

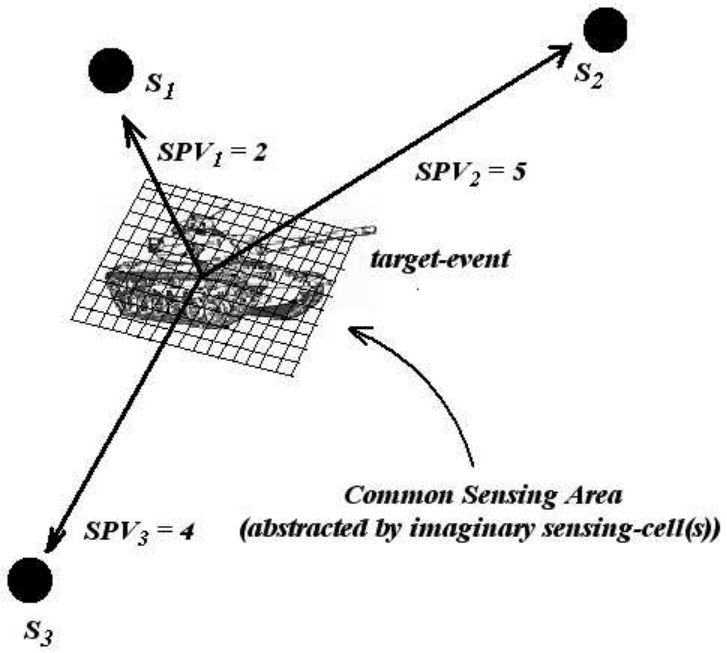

Figure 3.10: Sensing Proximity Concept with respect to a target

We also calculate the distance between the sensor node and the center of the square sensing cell i with coordinates as (x_{cell}, y_{cell}) and denote it as d_{cs}.

$d_{cs} = \sqrt{(x_n - x_{cell})^2 + (y_n - y_{cell})^2}$

Finally, SPV is calculated as the ratio of d_{cs} to d_{csmin} and is rounded to the nearest integer.

$SPV_i \approx \lceil d_{cs}/d_{csmin} \rceil$

In order to evaluate the cumulative sensing coverage of a shared region commonly monitored by neighboring sensors, we introduce two more sensing parameters, *cumulative sensing proximity value* (CSPV) of a sensing cell and the *cumulative sensing degree* (CSD) of a sensor node.

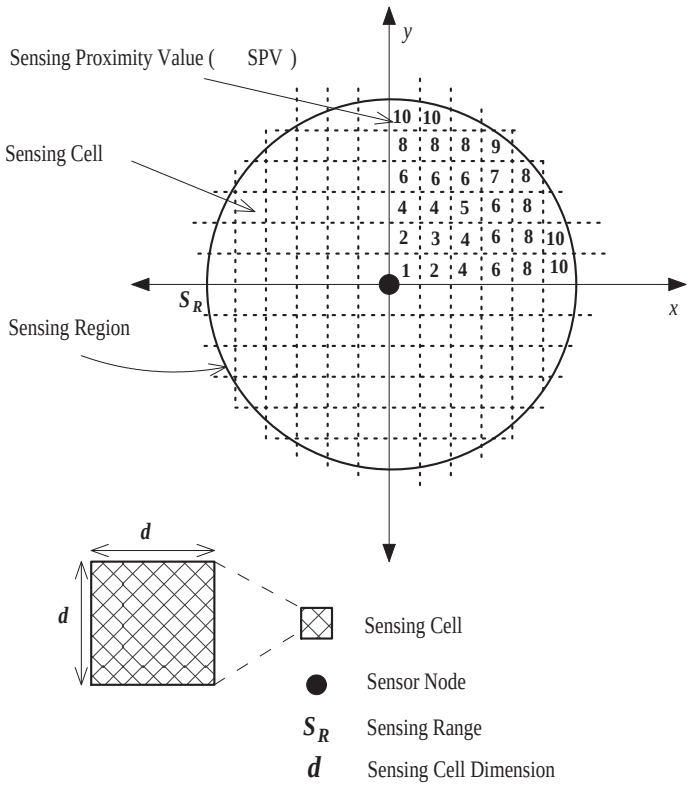

Figure 3.11: Sensing Coverage Approximation

Cumulative Sensing Proximity Value ($CSPV$)

$CSPV$ for a sensing cell is the cumulative SPV of all the overlapping sensing cells from the neighboring nodes covering that cell. Thus, if SPV_x is the SPV of sensing cell x and if n sensing cells having SPV values $SPV_1, SPV_2, SPV_3, ..., SPV_n$ overlap with cell x, then $CSPV_x$ is calculated using the reciprocal reduction technique which is formulated as

$$CSPV_x = SPV_x - \sum_{i=1}^{n} 1/SPV_i$$

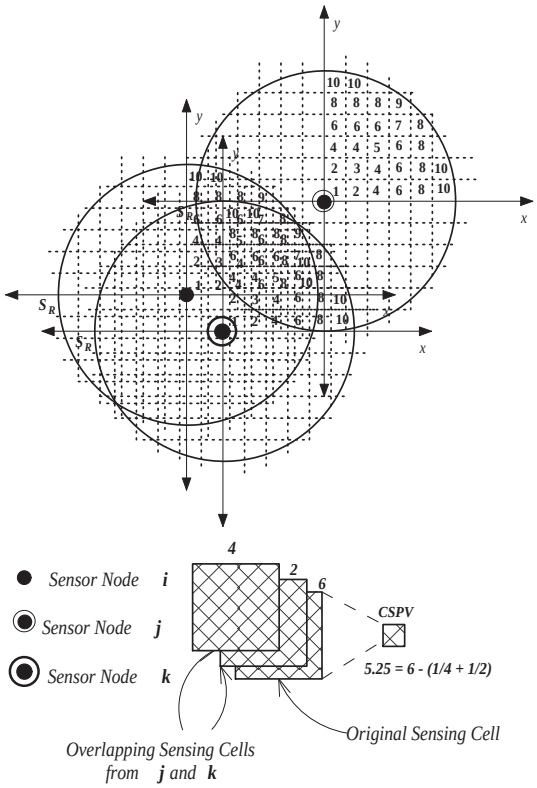

Figure 3.12: Calculating Cumulative Sensing Coverage

Thus, the more sensing cells overlap, the lower the final value of $CSPV$. Moreover, if cells having equal SPV values (say $spv = y$) overlap, then a total reduction of y will be adjusted toward the final $CSPV$ value only if y or more cells overlap. Finally, the $CSPV$ value is always adjusted to be within the range of 1 and SPV_{max}.

Cumulative Sensing Degree (CSD)

CSD describes the degree of cumulative fault tolerant sensing for a common area monitored collaboratively by some sensors. We calculate the sensing coverage of a sensor node as the average sum of the $CSPVs$ of all sensing cells covering its sensing area. Ideal sensing coverage would imply that all the $CSPV$ values for the

sensing cells of a sensor are 1 (i.e., each sensor node is covered by the maximum possible neighbors), whereas solitary sensing coverage would mean just the average of the sum of $CSPVs$ of a sensor node having no neighbors. Finally, CSD is calculated as percentage coverage and is given by the formula:

$$CSD_{sensor} = (1.0 - ((avg(\sum CSPVs) - ideal)/(solitary - ideal))) \times 100$$

3.2.3 Proposed self-organization Algorithm

We assume the existence of a neighbor discovery stage that precedes our self-organizing algorithm. In this stage, each sensor acquires knowledge of its neighbors and their positions. An example of 15 sensor nodes deployed in a hostile area to detect military tanks is shown in figure 3.13, which after neighbor discovery forms the network shown in figure 3.14. We construct a CDS hierarchy using the hierarchical CDS construction algorithm outlined in section 3.2.1. We also use the following metrics in order, along with the rules to break the ties. These metrics are energy level (EL), sensing-based metric known as CSD, connectivity-based metric or node degree (ND), and finally ID of the sensor node to break the tie. During the initial marking process, each sensor node exchanges one-hop neighbor information with its neighbors. This results in sensors gathering two-hop neighbor information, and also their corresponding location information. Figure 3.15 shows the CSD of the sensors and also the result of the initial marking process. The percentage CSD value calculated during the initial marking process is used in the subsequent hierarchical dominating set reduction processes. Figure 3.16 shows the 3-level CDS hierarchy formed after performing the dominating set reduction three times.

Our objective of having sensing zones is to have a self-sufficient collaborative group of sensor nodes that need as few sensing inputs as possible from sensors

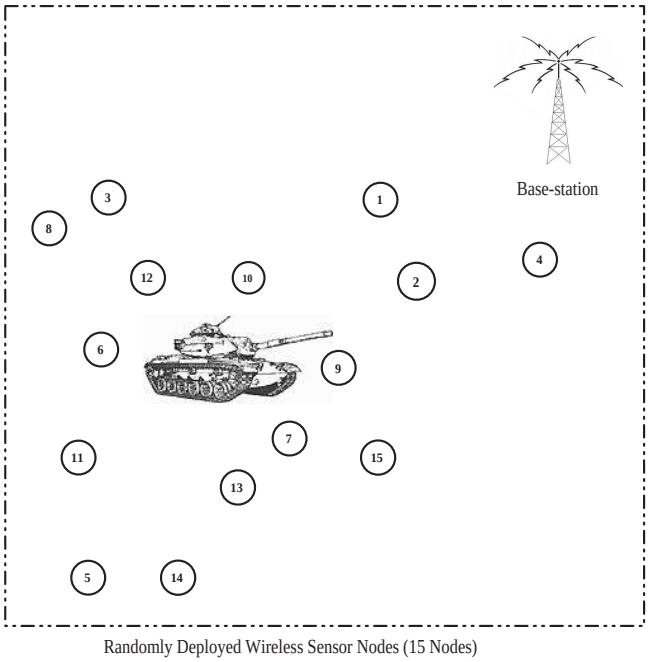

Figure 3.13: An example 15 sensor nodes random deployment for tracking an enemy tank

outside the group to reach consensus on any target events sensed. Such a group organization would need to be coordinated by a sensor coordinator. The sensor coordinator is that sensor node that has the maximum percentage coverage in the neighborhood. This implies that the chances of an event being detected by a sensing zone in its region coordinated by a node with higher CSD would be higher than any of its neighboring sensing zones. The sensor coordinator would then initiate a consensus among its sensor collaborators to rule out the possibility of a spurious event or noise. This leads us to another interesting *Sensing QoS metric* for a sensing zone that can be specified as the minimum percentage coverage or CSD of a certain region of WSN deployment. Figure 3.17 illustrates all these sensing zone concepts.

From figure 3.16, we can see that as we go up the CDS hierarchy the number of dominating nodes reduces. We can naively select the dominating nodes at any

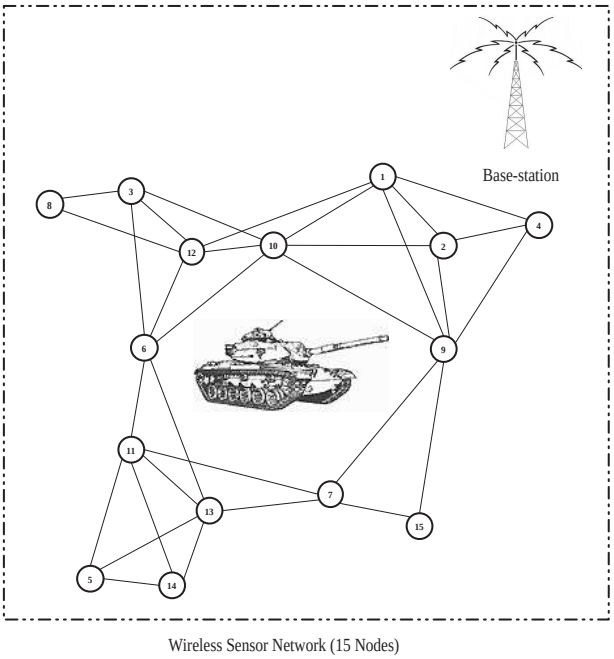

Figure 3.14: Wireless sensor network after neighbor discovery stage

higher level of the hierarchy to act as sensor coordinators. But as mentioned earlier, dominating nodes in the CDS hierarchy are essentially used as backbone nodes to route application specific sensing queries to the sensors and/or sensing data from the sensors to a data sink. An intuitive suggestion is to select sensor coordinators from the lowest level of the CDS hierarchy, level 0. We also know that the hierarchical dominating reduction process is a recursive process that uses marked nodes (or dominating nodes or backbone nodes) from the previous level to form the next level of the hierarchy. This means that our suggestion to use level-0 marked nodes as sensor coordinators has to be revised to include only those level-0 marked nodes that get removed during the dominating set reduction process to form level 1. In other words, our self-organization algorithm chooses sensor coordinators from level 0 marked nodes (but level 1 unmarked) as these nodes will not be acting as backbone

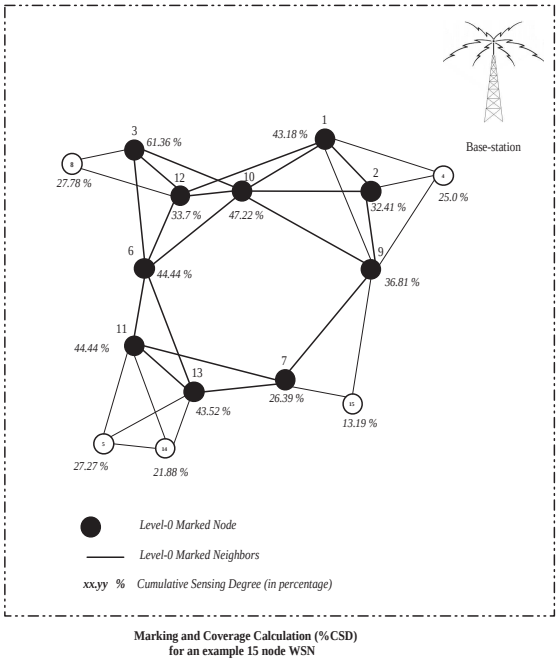

Figure 3.15: Wireless sensor network after the marking stage

nodes in upper levels of the CDS hierarchy. We note one more advantage of selecting sensor coordinators from level 0 marked nodes (and level 1 unmarked). They are the majority of available nodes in comparison to any other levels. Thus, we have a bigger pool of nodes from which we can select sensor coordinators. In order to have sensor coordinators at level 0, i.e., nodes with maximum percentage CSD, the algorithm uses an adaptive sensing-based metric. This means that during level 0 marking (or dominating set reduction) we eliminate those nodes (during tie breaker stage) that have the lower percentage CSD. This results in all level 0 dominating (or marked) nodes as nodes that have maximum percentage CSD within their one-hop neighborhood. Finally, from level 1 onwards, we eliminate those nodes that have the higher percentage CSD, which again leaves higher percentage CSD marked nodes at level 0 (but level 1 unmarked). The overall effect is that we make

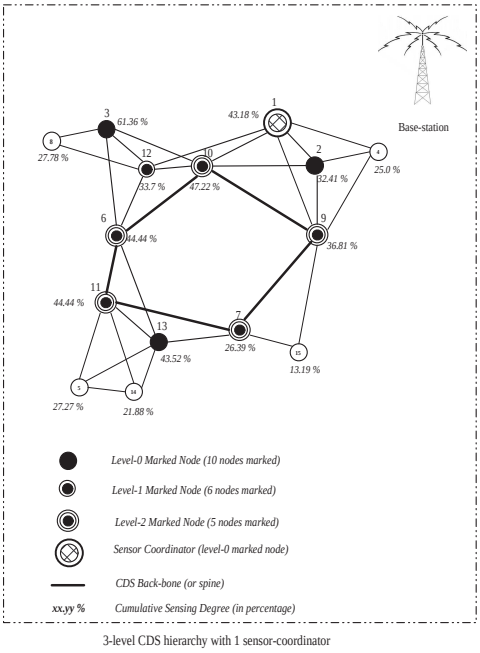

Figure 3.16: CDS hierarchy with sensor coordinator

sure that during both level 0 and level 1 marking, all the unmarked nodes at level 1 (but level 0 marked) have the maximum possible percentage CSD. This leaves a larger crowd of eligible sensor coordinators as level 0 nodes who are not dominating at any other higher level. Figure 3.18 shows the selected sensor coordinators.

In order to reduce from a list of probable sensor coordinators, we select only those nodes that have a higher percentage CSD than their level 0 marked (and level 1 unmarked) neighbors. If there is a tie, then we break it by the number of marked level 1 neighbors an eligible sensor coordinator may have. If there is still a tie, then we use sensor node ID as the final tie-breaker. An eligible sensor coordinator that passes the above three selection criteria would then advertise to its neighbors with its maximum percent CSD value. Sensors hearing this advertisement join the nearest soliciting sensor coordinator. However, in order to limit the

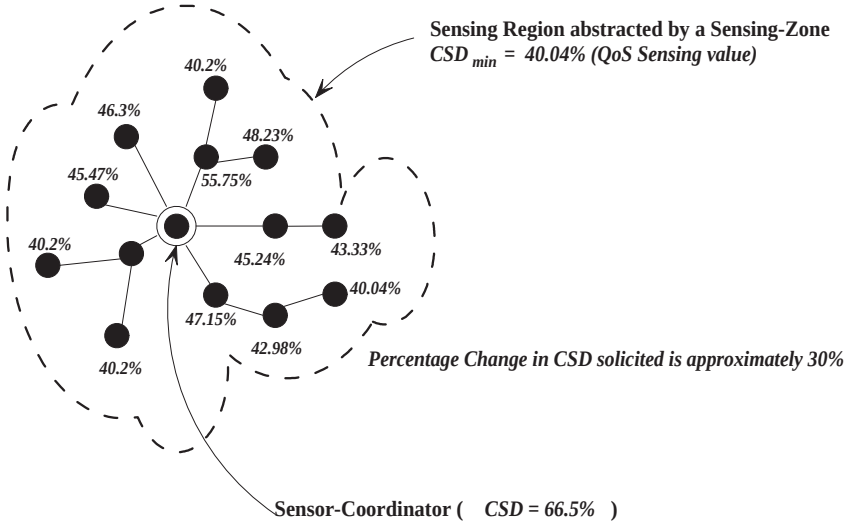

Figure 3.17: Sensing Zone Formation

overhead of sensing zone coordination and maintenance, we limit the group membership within a certain minimum and maximum number of sensor collaborators (or sensing zone members). Sensing zones with less than the specified minimum sensing zone membership will *merge* with neighboring *accepting sensing zones*. The reverse case applies for zones having membership larger than the maximum. In this case, sensor coordinators of these crowded sensing zones will ask distant members to find another neighboring sensing zone. These dismissed sensor nodes will join their nearest neighboring accepting sensing coordinator or sensing zone member. A sensor coordinator would *accept* a node as its zone member only if it has some space left to accommodate that node. Finally, all those nodes that were refused zone membership by their respective neighboring sensor coordinators due to zone size problems are considered as *orphan nodes*. Similarly, all those nodes who could not find any neighboring sensor coordinators due to a limited number of neighbors (or with sparse connectivity) will also consider themselves as *orphan nodes*. These

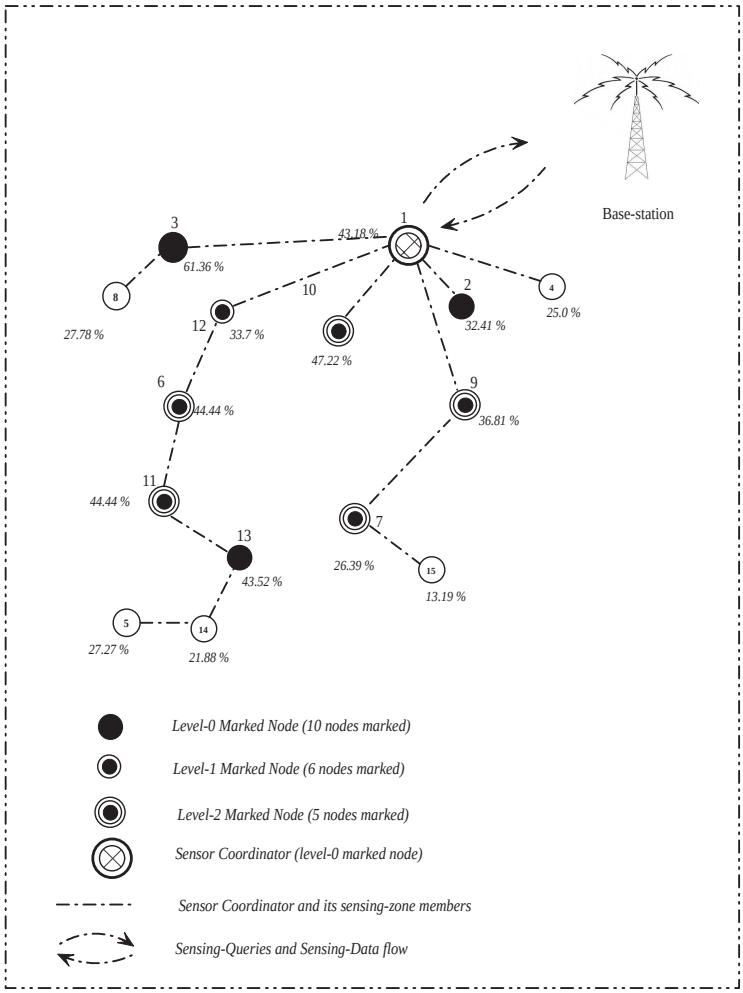

Figure 3.18: Sensing Zone Organization

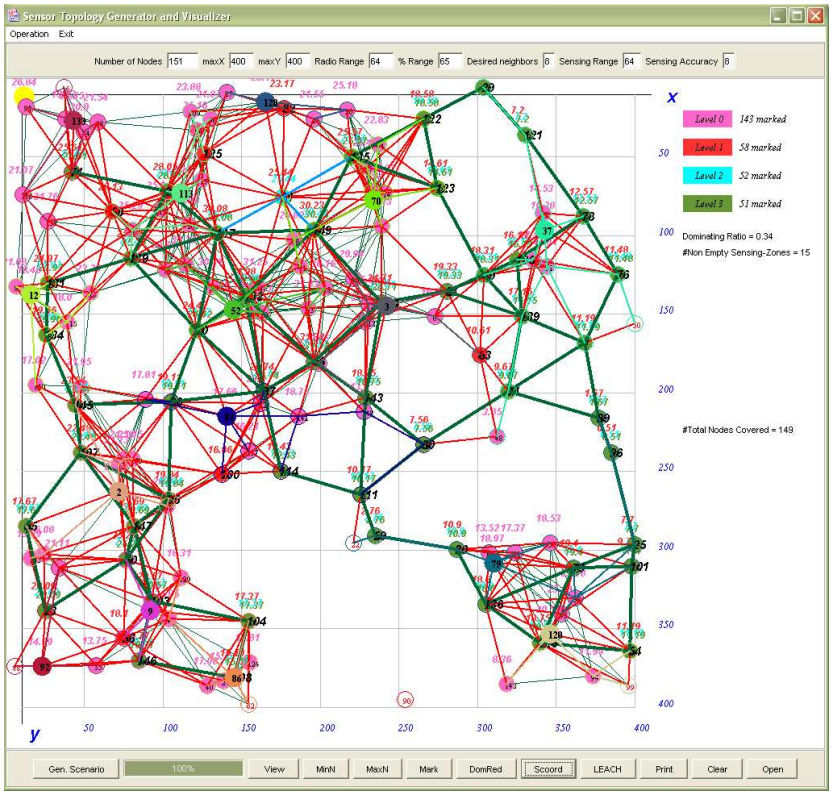

Figure 3.19: Our self organized infrastructure

orphan nodes will finally join any closest sensor coordinator or sensing zone member who would ultimately acquiesce to their join demands. The pseudocode of the self-organizing algorithm is outlined in appendix A.1.

3.3 Simulation

The role-based hierarchical self-organization protocol has been simulated using Java (JDK 1.3). The simulator can also be used to view the topology generated by the initial self-organization algorithm. A comparison between Leach and our approach is possible if we have the same number of clusters or sensing zones. To achieve this, the simulator takes the number of sensing zones generated from our

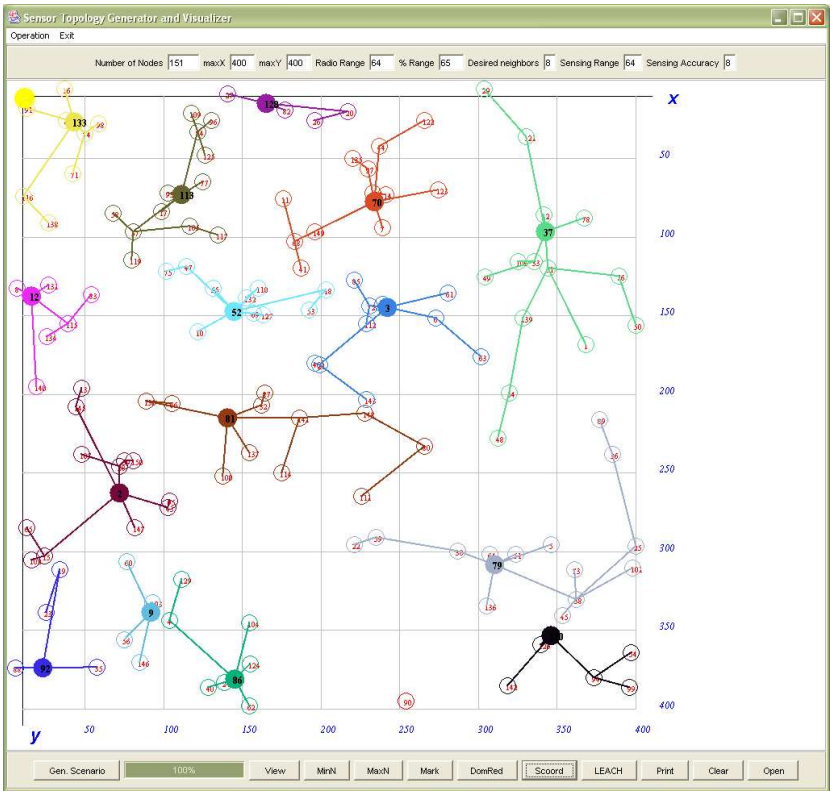

Figure 3.20: 150 nodes with 15 sensing coordinators

Table 3.1: Average group leader-member distances in Leach Protocol($d = 8$)

Network size	Leach					
	Mean			Standard deviation		
	MaxDist	MinDist	AvgDist	MaxDist	MinDist	AvgDist
100 (300x300)	80.81	17.02	47.11	9.73	3.05	4.49
300 (600x600)	90.70	18.50	52.14	6.93	2.46	3.16
600 (1400x1400)	228.24	27.98	121.28	21.57	2.85	9.08
1000 (2200x2200)	443.24	32.54	219.53	58.27	3.43	23.89

Table 3.2: Average group leader-member distances ($d = 8$) for our proposal

Network size	Our Protocol					
	Mean			Standard deviation		
	MaxDist	MinDist	AvgDist	MaxDist	MinDist	AvgDist
100 (300x300)	74.64	12.02	41.36	6.67	1.93	2.72
300 (600x600)	80.65	12.53	43.80	6.61	1.42	2.48
600 (1400x1400)	144.81	17.52	73.94	10.65	1.89	4.42
1000 (2200x2200)	150.16	18.52	76.83	11.75	1.73	5.17

protocol as input to the cluster based protocol. The simulator assumes no packet collisions. It also assumes that there are no packet errors during transmission and reception. In other words, we assume a perfect wireless channel. Figures 3.19 and 3.20 show the results of an example simulation run with the following simulation parameters:

1. Number of nodes = 150.
2. Maximum X, Y boundary coordinates of region of WSN deployment = 400 meters.
3. Maximum wireless radio range and sensing range = 64 meters.
4. Application specified sensing accuracy (d) = 8 meters.

We have performed 100 simulation runs on four different topologies:

1. 100 nodes in an area of 300 $\times$ 300 meters.

Table 3.3: Average static sensor CSD

Network size	d	Average static node *CSDs*					
		Mean			Standard deviation		
		MaxCSD	MinCSD	AvgCSD	MaxCSD	MinCSD	AvgCSD
100 (300x300)	8	29.27	14.08	22.54	2.19	1.59	1.46
	12	59.85	32.39	47.64	3.02	2.80	2.44
	16	80.09	47.59	66.43	2.87	3.65	2.56
300 (600x600)	8	26.47	12.96	20.46	1.37	1.08	1.03
	12	55.81	29.97	44.48	2.66	2.04	2.17
	16	75.93	44.11	62.34	2.01	2.42	2.02
600 (1400x1400)	8	12.91	1.52	7.20	0.51	0.34	0.38
	12	29.70	4.89	18.21	1.13	0.88	0.80
	16	45.16	9.14	28.93	1.36	1.26	0.96
1000 (2200x2200)	8	11.27	1.08	5.96	0.53	0.25	0.34
	12	26.59	3.52	15.57	1.15	0.75	0.79
	16	41.42	7.26	25.67	1.21	1.10	1.06

2. 300 nodes in an area of 600 $\times$ 600 meters.

3. 600 nodes in an area of 1400 $\times$ 1400 meters.

4. 1000 nodes in an area of 2200 $\times$ 2200 meters.

For all topologies, we have set the radio range and the sensing range to 64 meters. The minimum and maximum sensing zone (or cluster) membership size is set to 4 and 12, respectively. Finally, the application specified sensing accuracy or the sensing cell dimension (d) is set to values 8, 12, and 16 for the above simulation scenarios. Tables 3.1, 3.2, 3.3, 3.4, 3.5, 3.6 and 3.7 compare our protocol with the Leach-based protocol.

During the analysis of the simulation results, we will be using the terms *clusters* or *sensing zones* or *groups* interchangeably. From tables 3.1 and 3.2, it can be seen that our self organizing protocol organizes sensors into sensing zones with less distance variation compared to Leach. Moreover this distance variation becomes more pronounced as the topology becomes more sparse with an increasing number of nodes deployed in a larger area. Since the Leach protocol selects the clusterheads

Table 3.4: Average leader $CSDs$

Network size	d	Average Leader CSD			
		Leach		Our Protocol	
		Mean	Std dev	Mean	Std dev
100 (300x300)	8	12.77	1.91	16.53	1.49
	12	28.45	4.47	35.79	2.64
	16	41.52	5.00	51.21	3.86
300 (600x600)	8	11.50	1.06	15.07	0.89
	12	26.46	1.95	33.83	1.67
	16	39.07	2.91	49.14	1.91
600 (1400x1400)	8	4.95	0.68	10.46	0.54
	12	13.17	1.41	24.79	1.04
	16	21.49	2.05	38.05	1.26
1000 (2200x2200)	8	3.13	0.59	9.51	0.48
	12	9.80	1.49	23.06	1.14
	16	16.52	2.32	35.90	1.29

Table 3.5: Current organized average sensor CSD for Leach protocol

Network size	d	Leach					
		Mean			Standard deviation		
		MaxCSD	MinCSD	AvgCSD	MaxCSD	MinCSD	AvgCSD
100 (300x300)	8	21.02	4.14	9.98	2.34	1.12	1.29
	12	44.22	11.02	22.89	3.78	2.66	2.92
	16	60.93	17.10	33.88	5.08	3.59	3.68
300 (600x600)	8	18.77	3.09	8.42	1.71	0.63	0.66
	12	40.20	8.36	20.16	2.68	1.33	1.20
	16	57.87	13.31	30.32	3.91	1.84	1.75
600 (1400x1400)	8	9.66	0.16	3.66	0.72	0.11	0.29
	12	23.43	0.47	10.13	1.39	0.29	0.62
	16	36.36	0.95	16.79	1.71	0.48	0.85
1000 (2200x2200)	8	9.21	0.01	2.53	0.85	0.01	0.23
	12	23.02	0.08	7.48	1.21	0.05	0.44
	16	36.30	0.08	13.18	1.86	0.13	0.72

Table 3.6: Current organized average sensor CSD for our proposal

Network size	d	Our Protocol					
		Mean			Standard deviation		
		MaxCSD	MinCSD	AvgCSD	MaxCSD	MinCSD	AvgCSD
100 (300x300)	8	21.85	5.07	11.60	1.94	0.95	1.14
	12	46.45	12.98	26.48	3.57	1.96	2.24
	16	63.98	20.41	38.95	4.37	2.64	2.72
300 (600x600)	8	20.29	4.01	10.08	1.04	0.57	0.58
	12	43.98	11.10	23.90	2.21	1.17	1.30
	16	61.26	17.87	35.86	2.52	1.60	1.38
600 (1400x1400)	8	11.40	0.78	5.17	0.52	0.18	0.35
	12	26.62	2.69	13.91	1.01	0.51	0.70
	16	40.87	5.25	22.82	1.23	0.90	0.87
1000 (2200x2200)	8	10.08	0.64	4.40	0.48	0.14	0.29
	12	24.43	2.25	12.40	1.04	0.46	0.72
	16	37.93	4.60	20.98	1.13	0.93	0.95

Table 3.7: Average group membership sizes ($d = 8$)

Network size	Average Membership			
	Leach		Our Protocol	
	Mean	Std dev	Mean	Std dev
100 (300x300)	10.66	1.11	10.82	1.07
300 (600x600)	10.93	0.78	10.72	0.65
600 (1400x1400)	21.18	2.74	13.35	1.01
1000 (2200x2200)	46.28	9.23	16.77	3.86

randomly rather than deterministically, most of the times this results in suboptimal selection of clusterheads. This in turn results in situations where sensors having distant soliciting clusterheads will extend their radio range in order to join any nearby less crowded clusters. It should be noted that the objective of any self organizing algorithm is to abstract the random topology into an easily controllable network infrastructure. Thus, any group based self organizing algorithm will try in a best effort manner to include each sensor node in at least one group. In pursuing such a goal, Leach ends up having larger group memberships than our approach. This can be clearly observed from table 3.7, where the average group size for our approach remains within 20 members whereas for Leach it may be up to 55 members as network size increases from 100 to 1000 sensors.

Tables 3.3, 3.4, 3.5, and 3.6 analyze the effectiveness of our organization and Leach with regard to the cumulative sensing degree (CSD) metric. Table 3.3 shows the mean of the maximum, minimum, and average CSD of a sensor node assuming that it has all its neighbors in its group. We compare these *static* CSD values with the *current* CSD values obtained after the groups have been formed by the self organizing algorithm. It can be seen from tables 3.5 and 3.6 that our protocol always results in sensor nodes retaining most of their static CSD values, whereas Leach results in an appreciable loss in node CSD due to suboptimal selection of

clusterheads. However this difference is negligible because Leach has orphan nodes select any nearby distant suboptimal clusterhead in order to be registered in some group. Table 3.4 shows the mean of the average CSD values of the clusterheads or the sensor coordinators (also referred to here as *leaders*). It can be clearly seen that due to deterministic leader selection our self organizing protocol has higher average leader CSD values than Leach. One interesting result in all these tables is the dependency of our CSD approximation on the sensing cell dimension (or application specified sensing accuracy) d. In general, it is observed that lower values of d yield a better CSD approximation.

3.4 Summary

In this chapter, we present a role-based hierarchical self organization algorithm for wireless sensor networks. The algorithm groups sensors into sensing zones that are coordinated by a sensor coordinator. We also propose a sensing based metric CSD (known as *Cumulative Sensing Degree*) to form sensing zones. In order to form a hierarchy of back bone nodes we extend the CDS formation algorithm proposed in (Wu and Li, 1999) (Wu, 2002). The resulting self organized network consists of sensing zones that are connected to each other by a hierarchy of backbone nodes.

The simulation results show how a randomized cluster based organization performs worse as network size increases. Since our algorithm selects sensor coordinators deterministically, we have shorter distances between sensing zone members and the sensor coordinator. From figure 3.20 it can be seen that there is still some overlap between neighboring sensing zones. The amount of overlap among neighboring sensing zones reflects the need for collaboration for events occurring at the border of the sensing zones. For targets or events that migrate across sensing zones,

this overlap of sensing area among sensing collaborators reporting to two or more sensing coordinators is very useful and necessary. However, the size of the membership in this area of overlap among sensing zones needs to be controlled adaptively with moving events and changing network dynamics, if one needs to minimize radio communication overhead in these areas.

Our algorithm is a first step toward network organization after neighbor discovery. In the next chapter we generalize our role-based organization technique to implement a role-assignment framework (URAF) that unifies both the maintenance and the reorganization phase (similar to (Subramanian and Katz, 2000)) of a complete self-organization algorithm for wireless sensor networks. In most of the previous research literature it is assumed that a hierarchical organization is too static (or rigid) to be reorganized with respect to the ultimate traffic pattern that may run on top of this self organized network architecture. It is also believed that concentrating specific responsibilities on specific nodes will result in such nodes becoming likely points of failure, thus making such a hierarchical network inherently less fault tolerant. However with sufficient network density, the unified role-assignment framework (URAF) can support efficient solutions that systematically rotate or (re)assign roles among neighboring nodes in a localized manner without much overhead.

Chapter 4 – Unified Role-Assignment Framework for Wireless Sensor Networks

In this chapter, we conceptualize a generic role-based assignment framework. This framework is an extension of the Connected Dominating Set (CDS) based reduction technique introduced in chapter 3. We have used this technique to identify specific roles in a role-based hierarchical network organization (RBHSO). The technique employs the use of rules that elicit the list of metrics a node should recursively compete with in order for it to be locally dominating among its peers for a particular role.

The use of a particular metric for the selection of a specific role is design dependent and is most importantly influenced by the application requirements expected of that role and the environmental characteristics within which the role is supposed to deliver. In other words, not every instantiation of a specific role using the same set of rules may be enough for all applications.

With changing environmental and network dynamics, it becomes difficult for an end-user or the network operator to envision and implement clear-cut strategies incorporating trade-offs among diverging (if not converging) application requirements. This problem is not only limited to application solutions that execute as roles in the network. It could manifest for any higher layer protocol solution that is based upon certain cross-layer optimizations for protocols at the lower layers. The Unified Role-Assignment Framework (URAF) is a generic and flexible platform that supports implementation of custom strategies such as topological role assignment

for task mapping, role scheduling for dynamic resource management, and role reassignment for node load balancing. In case of ad hoc wireless sensor networks, conserving energy while meeting delay requirements simultaneously are the set of diverging parameters that applications usually optimize.

The scope of this chapter is limited to the features provided by the framework. The framework gives the flexibility and the responsibility to the end-user to use these features to come up with a specific solution in terms of roles and rules. The framework is not a solution unto itself. However, two different protocol solutions based upon the same implementation of the role-based framework can be compared and analyzed.

The chapter is organized as follows: We first provide the motivation for a role-based framework. The next section discusses the design aspects of the unified role-based assignment framework (URAF). It provides an introduction to the basic controls or interfaces provided by the URAF. These serve as tuning knobs for applications to embed specific service requirements for further customizations. In particular, we discuss the following:

1. Hierarchical decomposition of an application service to tasks and then to roles. We abstract inter-task coordination in terms of a hierarchical task-graph. These tasks (or sub-tasks), when executed distributively by nodes in the network, require hierarchical coordination and management. This is represented in terms of two way handshake based coordination among manager roles and worker roles. A role-coordination graph represents these coordinations in terms of message exchanges among roles.

2. We use domain specific models such as the role-energy model (R–ΔE) and the role-execution time model (R–ΔT) to abstract the measurement of energy and

time-based performance metrics for individual roles.

3. We propose rules that are used as role-controls for role assignment, failure monitoring, feedback, and role-reassignment for load balancing and/or repair.

4. We elicit a network model for unifying the use of node resources in terms of a common metric, that is, energy. We use this model to profile the load imposed upon a node by a role in terms of the energy usage per unit time.

5. We assume the existence of a concave role service utility (RS–ΔU) model for the domain for which sensor network services are being optimized. This model is an abstraction of a general observation from real life sensor networks. In general, it is observed that the marginal benefit of assigning additional resources (in terms of roles) beyond a certain point does not result in any meaningful improvement in service quality. This model is needed to understand the initial number of roles with which the role assignment algorithms begins in order to achieve the desired performance in the network.

6. We also assume the existence of basic protocol layer functionalities such as channel access or wireless media scheduling schemes for communication, flow control, transport level reliability schemes, security, position estimation, time synchronization, sensing, actuation, processing using A/D converters and the on-chip CPU, data storage and caching. In other words, we assume the existence of open network interfaces up to layer four along with a repository that shares interesting cross layer information.

The third section provides a high level architecture of the framework. This section correlates several features of the framework into specific components. This

serves as a reference to unite all the features that the framework supports. A high level algorithm lists the basic steps for any application service to be mapped in terms of roles and executed in-network.

In the fourth section, we consider a generic role-based formulation of the problem of minimum resource usage and minimum performance delay for any sensor application. With this, we investigate role-assignment (RA) requirements for energy minimization of individual and multiple services running among the same set of sensor network nodes. Specifically, we consider algorithms for Minimum Total Energy RA (MTERA) and Multi-Service Minimum Energy Role Assignment (MSMERA) requirements. These problems when resolved in terms of an optimal assignment of roles to nodes is essentially equivalent to computing the MCDS (Guha and Khuller, 1996) which is a well-known NP-hard problem. A combinatorial scheme that solves both the MTERA and MSMERA problems distributively is difficult and an optimal solution to this problem is NP-complete. We propose a number of techniques that fine-tune our generic rule-based RA algorithms (based on the CDS technique). We propose a novel technique that is a hybrid approach known as the Cooperative Redundant Coalitional Role-Assignment (CRC-RA) with iterative pruning. We conclude this section by evaluating the generic and flexible nature of the framework (URAF) toward implementing other practical sensor application requirements.

Finally, we conclude the chapter by summarizing the features and limitations of the Unified Role Assignment Framework.

4.1 Motivation

The design of the framework builds upon fundamental aspects of large-scale distributed systems such as *organization*, *hierarchy*, *aggregation*, *redundancy*, and

approximation. In the following discussion, we use the terms functions, tasks, or roles interchangeably. The framework uses these concepts in the following broad manner:

1. Organization: The network organizational architecture provides a framework for decomposing a global optimization problem into a set of local optimization problems with controlled interactions among them. The organizational architecture or structure acts as an underlying overlay such that it not only determines the patterns of network communication activity but also controls relationships among neighbors participating to perform a service. This overlay exploits locality by structuring the deployed network into manageable network partitions. Nodes belonging to a particular sector or partition thus limit both their information exchange and the resulting load experienced to within that partition.

   For example, in chapter 3, the role-based hierarchical organization (RBHSO) provides the basic infrastructure to support localized services, resource sharing, and collaboration. It also promotes scalability in the presence of a large number of sensor nodes and target events.

2. Hierarchy: Nodes are organized into a hierarchical network such that the control is done by nodes at higher levels for nodes below them that execute specific protocol tasks. This service control responsibility can be assigned incrementally and recursively to nodes at higher levels of the hierarchy. This hierarchical coordination mechanism is contextually dependent upon node density and the number of target events present in the network. In chapter 3, the RBHSO algorithm is evaluated against a randomized cluster based orga-

nization for different network sizes, the number of sensing zones (or clusters), and the overlap among them.

For example, if the monitoring area is small, and events are infrequent, only a few sensing nodes may need to be made available to monitor and track an event. In this case, a hierarchical organization where sensing nodes at the lower level of hierarchy report to some sensing manager at the higher level would be overkill both in terms of communication overhead and monitoring requirements. For this scenario, a peer-to-peer single level coordination architecture may be sufficient. On the other hand, if the monitoring area is large and the events are many and frequent, and the sensing resources are scarce, a multi-level hierarchy may be more appropriate.

3. Aggregation: Collaboration among nodes in the neighborhood pursuing a specific mission signifies an aggregate or a group (of members) that acts as a single entity to execute an application desired service. In other words, if a service or a task can be expressed as an aggregation of specific functions, a single node or a set of multiple nodes in the vicinity can undertake any combination of these functions such that their collaborative execution accomplishes the service or task. This collaborative function assignment to nodes can change over time. The optimal functional decomposition of any service is highly dependent upon the domain and the underlying network topology. Moreover, in order to reduce hot spots or functional disparity among nodes, it is necessary that the load be balanced among nodes executing specific task functions. It is also necessary that any functional load and the aggregation of these be empirically expressed in terms of a common metric, say energy used per unit time. Such a metric can serve as common yardstick for load

balancing protocols to balance the average load among nodes.

4. Redundancy: Fault tolerant network protocols usually require redundancy both in terms of functions and their assignment to nodes. In other words, critical functions should be replicated in advance. Critical functional state updates should also be cached redundantly across nodes. Redundant nodes should be made available in the vicinity of critical roles (tasks or functions) such that the network can dynamically anticipate a failure and also repair itself. This allows nodes to quickly detect neighbor failures instead of waiting for a status response to a query, thus supporting dynamic and transparent adaptation to repairs.

5. Approximation: It is difficult to gather and disseminate complete network information to all nodes in the network. This is not feasible due to the large communication overhead, storage, processing, and energy constraints. Moreover, network state changes are very dynamic and contextual. Most of local state changes may not be useful to remote nodes in the network in comparison to the overhead involved in communicating these. Moreover, by the time, the information is received and processed at any node in the network, the information has already become stale and irrelevant. Thus, approximate state information along with locality should be considered in the reasoning necessary to make effective coordination decisions among nodes collaborating to perform a service.

   The concept of approximation is most valuable for characterizing the general performance characteristics of a global service that has been partitioned to a set of local tasks and is assigned to different nodes in the network. In this

case, for additive or multiplicative metrics such as delay, energy, etc. the global performance is approximately equivalent to some aggregate function (such as addition or multiplication) of performance expected of nodes participating to perform that service. This property is also useful to fault-tolerant and resource adaptive protocols that use approximation to predict task performance and failures.

4.2 Role Abstraction Concepts

In this section, we discuss the following abstraction ideas conceptually:

- A service is composed of several tasks or subtasks that are cumulatively executed by nodes in the network,
- A role is a topological abstraction of tasks (and/or subtasks) and its subsequent node-level execution,
- A role-coordination graph (RCG) represents the topological organization and coordination among roles for a certain service,
- Rules abstract specific role requirements expected of any node executing that role, and
- Role assignment (RA) is the process by which nodes collaboratively and locally assign roles among themselves.

4.2.1 Services

Wireless sensor networks require several network services such as neighbor discovery, localization, time synchronization, network self-organization, data aggregation, storage, event detection, monitoring, tracking, and reporting. These network

services are usually executed locally and in a distributed fashion among a group of neighboring nodes each one collaboratively executing a task or a set of tasks for that service.

Service provisioning in networking terms usually requires the provisioning of networking elements for specific tasks within the network such that their collaborative participation over time cumulatively delivers the desired service. The networking elements here refer to the use of various node and network resources such as sensors, radio, memory, cpu, and storage. In addition to distributed provisioning, there is need for accounting, management, and load balancing of network resources among services. A hierarchical network organization allows both control and service functions to be managed and executed in a scalable, localized, and distributed manner.

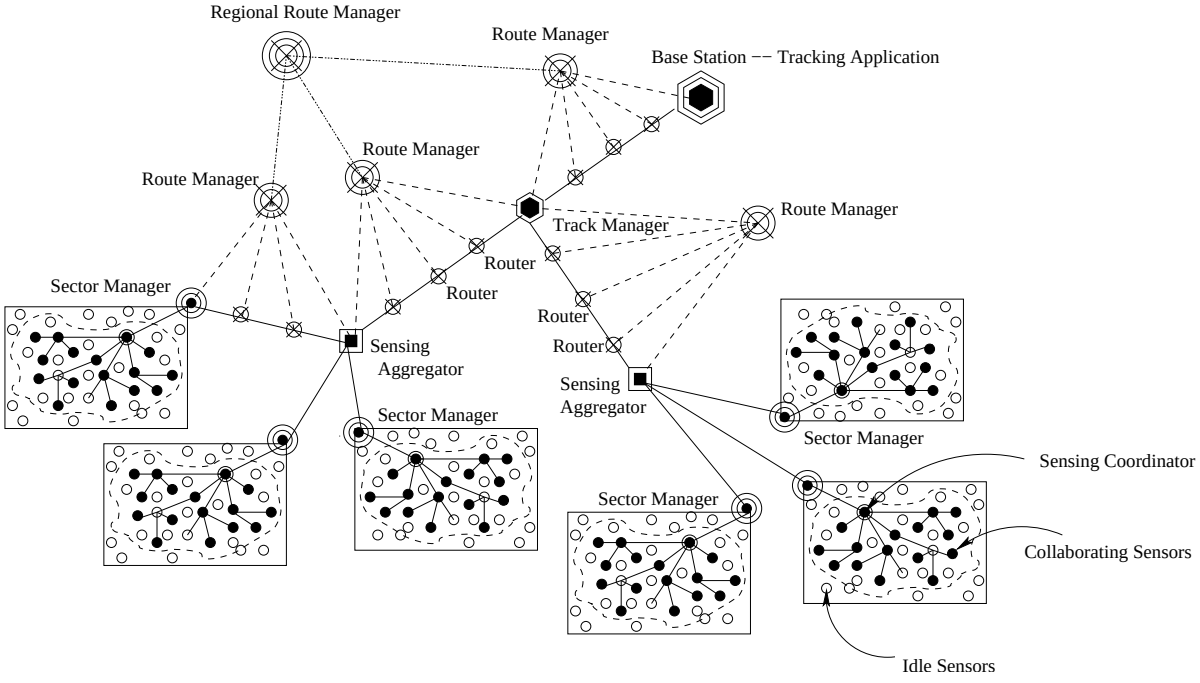

Figure 4.1: Provisioning of Network Resources for Tracking Application in WSNs

Figure 4.1 shows a tracking application executed by select nodes in the wireless

sensor network (WSN). At the network organization level, the sensor nodes are grouped into sectors that are managed by the sector manager. Every sector manager is hierarchically connected to nodes that act as sensing aggregators and/or routers. In other words, depending upon the proximity of a node to a region where sensing events are being monitored, a node that was acting as router can assume a data-centric role such as a sensing aggregator. The routers in turn are managed hierarchically by route managers. At a regional level, there could be a regional route manager (RRM) that manages several neighboring route managers. Since in WSNs the sensing event data is location-centric, it aids mapping of tasks such as sensing coordinators, sensing collaborators, and sensing aggregators in the vicinity of an event. A sensing coordinator manages the sensors within its sector and passes information to the sector manager or a sensing aggregator. Sensors that participate in sensing and monitoring an application-desired event act as sensing collaborators and they dynamically elect among themselves a sensing coordinator for coordination and management of the sensing activity or task. A sensing aggregator can collaborate among several sector managers to monitor and track an event moving across neighboring sectors. Sensing aggregators for different regions in turn report to a higher level node known as the track manager in this hierarchical organization of the tracking service. A remote base station could query this track manager at desired intervals and this data is eventually consumed by the tracking application for further processing. Nodes that do not participate in providing this service are idle and redundant. Such idle nodes can be managed effectively to repair failures at different levels in the hierarchy by a sensing coordinator or sector manager or route manager or track manager.

The Unified Role Assignment Framework (URAF) supports specific abstractions

that are based on the above-mentioned concepts of network organization, hierarchy, approximation, aggregation, and redundancy to manage the topological partition and mapping of several functions (or tasks) for any application service. These abstractions are flexible to capture and unite both the low level resource requirement of any elementary task and the high level topological organization and coordination of such resources for their effective management across services and nodes in the network. This section highlights the network level topological abstraction of the tracking service. In the next section, we generalize the high-level hierarchical decomposition of services to tasks to form an intermediate and low-level abstraction known as task graphs.

4.2.2 Task and Task Graph (TG)

The task-based abstraction is a low-level resource based approximation. In other words, the task abstraction requires that a service be hierarchically decomposed to its elementary tasks such that these tasks at the lowest level approximate the principal resource required to execute it. This service to task decomposition is then organized and managed by a hierarchical organization of worker–manager tasks. Figure 4.2 shows a generic organization of tasks and their management. At the lowest level, the subtasks, say $T_{a1,2,3}$, interrelated to a task say T_a, approximate the use of resources, 1, 2, and 3. These subtasks are then managed by its next level manager, say task $T_{amanager}$. This is recursively continued for management of unrelated tasks at the next level, say T_a and T_b, to get the $T_{abmanager}$ until we get the $T_{ab...zmanager}$. This hierarchical decomposition known as a *task graph*, represents a centralized organization that spans hierarchically from a parent to its children. In practical situations, such a clear-cut service decomposition and organization is

not possible and is also not recommended for ad hoc networks. This is because of the overhead involved in constructing and maintaining such a multi-level task hierarchy. At best, a two to three level hierarchy such as the one organized in the RBHSO architecture described in chapter 3 is recommended.

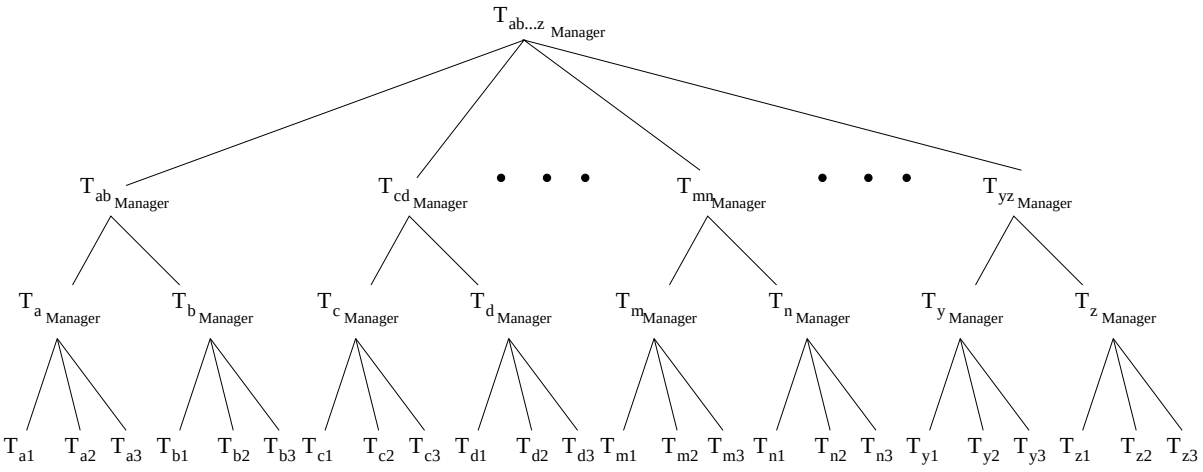

Figure 4.2: Task Graph: Hierarchical organization as worker–manager tasks.

In addition to expressing the worker–manager relationship between the various sub tasks and their associated parent task for a certain service, a task graph should also account for the various dependencies and redundancies among neighboring tasks. Tambe (Tambe, 1997) in his research on flexible coordination framework among group members has identified three primitive inter-task relationships such as (1) AND-combination, (2) OR-combination, and (3) task dependency.

For example, symbolically a $[SERVICE]$ could be expressed in terms of m different tasks $\{T_1, T_2, T_3, \ldots, T_i, \ldots, T_m\}$ along with their specific inter-relationships. Among these tasks, some tasks may be redundant and some may be necessary, whereas some tasks are dependent on each other temporally, spatially, or topologically. Thus, if all the tasks were necessary, the service could be expressed

as $[SERVICE] \Leftrightarrow \bigwedge_{i=1}^{n} T_i$. On the other hand, we could have $[SERVICE] \Leftrightarrow \bigvee_{i=1}^{n} T_i$. We could have $T_i \rightarrow T_j$ which means T_j is directly dependent on T_i. This directed dependency relationship, when perceived temporally, would imply that T_i needs to be executed first so that T_j can execute next. From the spatial or topological perspective, it implies that the nodes executing both T_i and T_j can either be the same node or they should be in the neighborhood. When both T_i and T_j are dependent on each other in a way that there is no need to maintain specific causal ordering for scheduling their execution, then they are said to be reflectively dependent, $T_i \rightleftharpoons T_j$.

In summary, a task graph represents the following:

1. Service decomposition in terms of tasks and subtasks,

2. Specific temporal, spatial, and topological inter-dependencies among tasks, and

3. Inter-task coordination abstraction in terms of hierarchical worker–manager organization.

Such a hierarchical task graph allows a seamless and transparent mapping of any service to any hierarchical network organization such as a tree or a cluster. For battery powered sensor nodes in a wireless sensor network, this results in energy savings as the state maintained for these network organizations can be iteratively reused for any necessary task assignments and their subsequent adaptations depending upon application requirements or available energy. In the next subsection, we consider an example partitioning and task graph organization of a data aggregation service. We discuss the ability of the framework to embed a domain specific application requirement of a data aggregation service on such a task graph.

Example task graph for a data aggregation service

In figure 4.3(a), we see that the data aggregation service is hierarchically partitioned into tasks and subtasks until no further decomposition is possible. For example, at the topmost level of the hierarchy the data aggregation service consists of the coordination among the following tasks such as sensing coordination, routing, and data collection by node(s) acting as sink(s). Similarly, each of these tasks are again partitioned into subtasks, with the task at the immediate higher level acting as a coordinator for lower level subtasks. For example, the sensing coordination task at the higher level is essentially a coordination among sensing, fusion, and data forwarding subtasks at the lower level of the hierarchy. Eventually, such a recursive partitioning of tasks into a hierarchy continues until the task(s)/subtask(s) at the lowest level cannot be further subdivided.

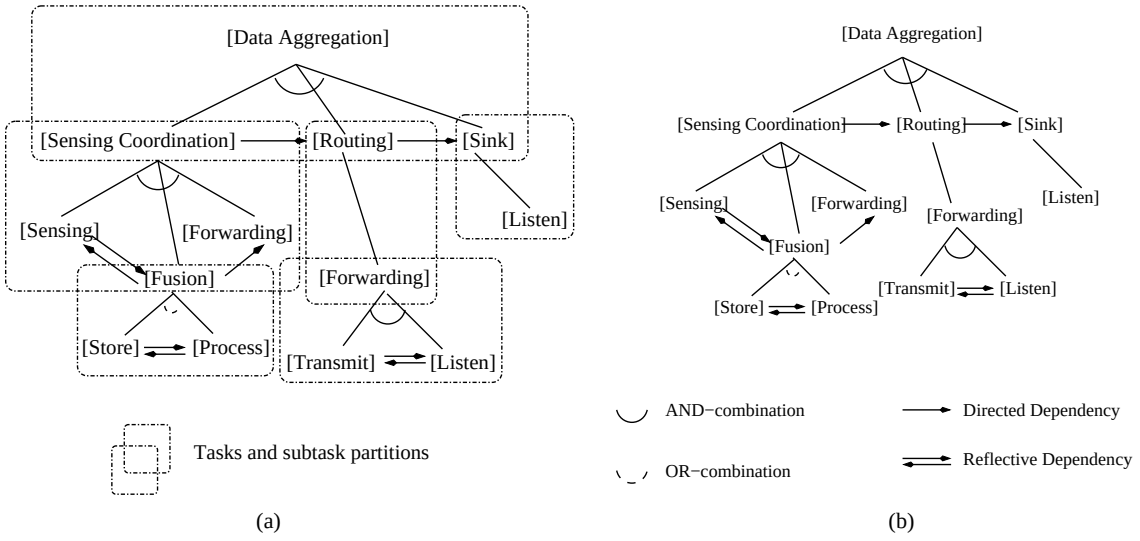

Figure 4.3: Data Aggregation: (a) Hierarchical decomposition and (b) Task Graph

In figure 4.3(b), we see observe the following:

- $[DataAggregation] \Leftrightarrow [SensingCoordination] \bigwedge [Routing] \bigwedge [Sink]$ which means

that the service needs all the 3 tasks of sensing coordination, routing, and a sink.

- $[SensingCoordination] \rightarrow [Routing] \rightarrow [Sink]$ implies that the order of execution is sensing coordination, then routing, and then finally sink.

- The $[Fusion]$ subtask consists of tasks $[Store]$ and $[Process]$ in an OR relationship, which means that the raw sensor readings may not need any processing and can directly be transmitted or they need to be stored and processed further before being forwarded to the next hop toward the sink.

- Both $[Store]$ and $[Process]$ subtask(s) do not seem to have a stringent causality-based scheduling requirements; They can be executed in any order. We will discuss role scheduling later.

A task graph representation makes it feasible to understand different ways by which a network can be tweaked to embed domain-specific objectives of a certain service. This is usually done by identifying and targeting only the relevant tasks or subtasks in the task graph that are concerned with delivering this specific service requirement. For example, it is easy for an application to specify the sensing fault-tolerance of a data aggregation service by specifying that it needs a minimum of, say, 5 sensors to cumulatively agree on genuine sensing events and/or discard spurious ones. From the perspective of a hierarchical partitioning of tasks into subtasks, such an objective can be taken care of by a regional sensing coordinator that verifies that it has at least 5 sensors reporting about any event.

Similar analogies apply to other tasks such as $[Fusion]$ and $[Routing]$ that can be tweaked for energy optimizations implemented by way of data compression and the sensing and monitoring periodicity expected by the service. Any application

information specified in this way allows the wireless sensor network to pursue increased energy savings, as it can then map these task partitions to appropriately sized groups within the vicinity of a sensing event. The URAF assumes the existence of domain-specific models that account for the tweaking of specific parameters with regard to general performance metrics such as energy and delay. We will discuss such models in a later section in this chapter.

4.2.3 Roles

A role is an abstract specification of the set of activities an individual sensor or a subgroup of sensors undertakes in the service of the group's overall activity. Thus, for example if the data aggregation service as shown in figure 4.3 were mapped to a region consisting of some number of nearby sensors, then the task(s) or subtask(s) executed by a node individually or together with its neighbors would determine their interaction in terms of their respective role(s).

In other words, there could be some set of nodes pursuing sensing coordination activity whereas a neighboring region would perform the routing task to route data from the sensing coordination region to the sink node(s). Within the sensing coordination region, one of the node may act as a sensing leader or coordinator for that region, whereas others would act as sensing collaborator(s) executing tasks such as $[Sensing]$ and $[Forwarding]$. The sensing coordinator may perform the tasks of $[Fusion]$ to fuse data from the sensing collaborators and $[Forwarding]$ to relay the processed information with hop-by-hop routing toward the sink. Thus, a role constrains a group member, say v_i (or a subgroup V' of deployed network V) to some subtask T_{v_i} of the overall service $[T]$ a group of sensors may execute. In other words, role-abstraction is the concept of a time-extended "role" representing the

behavior of a transient task. This behavioral interaction among roles is controlled by the role state machine described later.

4.2.4 Elementary and Complex Roles

We have identified the following elementary roles that represent fundamental tasks executed by a node within a sensor network. These fundamental tasks also represent the principal resources used by them during execution. These are:

1. ON: Node turned *ON* and is idle
2. OFF: Node completely turned *OFF* to save energy
3. Sense: Sensor role, S
4. Process: Processor role, P
5. Store: Node storing data in its memory, M
6. Transmit: Transmitter role, T
7. Listen: Role for listening to packets, L

These elementary task-based role abstractions allow for more complex roles compositions. For example, the forwarder role is basically composed of a listener and a transmitter role. Similarly, we have also formulated other complex roles:

1. Forwarder: This role consists of a transmitter and a listener i.e. $F \Leftrightarrow T \bigwedge L$
2. Router: This role consists of a series of forwarders i.e. $R \Leftrightarrow \bigvee_{i=1}^{n} F^i$
3. Aggregator: This role needs both the memory and the processor roles i.e. $A \Leftrightarrow M \bigwedge P$

4. Sensing Collaborator: Role similar to tree children/leaves or a cluster member, denoted by S_m i.e. $S_m \Leftrightarrow (S \bigwedge P) \bigwedge (T \rightarrow L)$. This role transmits the sensor readings and it listens for coordination.

5. Sensor Coordinator: Role similar to a tree root/parent or a cluster head, denoted by S_h i.e. $S_h \Leftrightarrow (\bigvee_{i=1}^{n} S_m^i) \bigvee_{j=1}^{m} F^j$

6. Sensing Region: Region represented by the whole cluster or tree, denoted by S_r i.e. $S_r \Leftrightarrow ((\bigvee_{i=1}^{n} S_m^i) \bigwedge S_h) \bigvee R$ or $S_r \Leftrightarrow ((\bigvee_{i=1}^{n} S_m^i) \bigwedge S_h) \bigvee_{j=1}^{m} F^j$. Alternatively, if there are several sub-clusters or subtrees, then the sensing region could also consist of several sub regions that could be defined by the expression, $S_r \Leftrightarrow \bigvee_{k=1}^{l}((\bigvee_{i=1}^{n} S_m^i) \bigwedge S_h^k) \bigvee_{j=1}^{m} F^j)$.

7. Target Tracking: Roles include track manager, sector manager, and sector, where the track manager is hierarchically at a higher level than sector managers. Thus, track managers keep track of event(s) moving through neighboring sectors with the respective sector managers collaborating via the track manager.

4.2.5 Role Coordination Graph (RCG)

Figure 4.4 shows the role-coordination graph for the data aggregation service. The coordination graph shows the roles organized in a hierarchy with the sensing collaborators (S_m) and the forwarders (F) being at the lowest level of the hierarchy. The sensing coordinator (S_h) is hierarchically dominating as it manages the sensing collaborators to perform data aggregation. Similarly, the router role (R) is also hierarchically dominating over the forwarders that relay the data to the sink. The use of hierarchy in a role-coordination graph (as shown in figure 4.4), allows roles

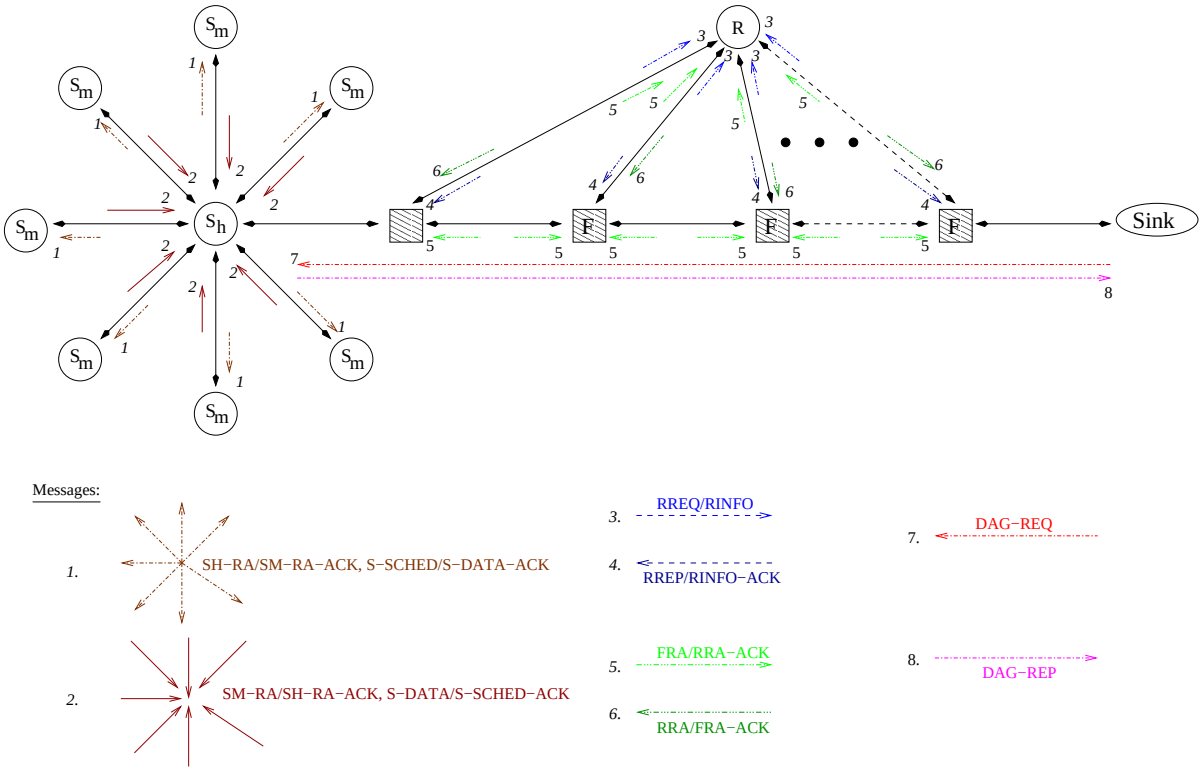

Figure 4.4: Role coordination graph for a data aggregation service

at a higher level in the hierarchy to provide coordination information to roles at a lower level in the hierarchy. This coordination information is communicated among roles by way of two way handshake message mechanisms.

For example, within the sensing region consisting of a sensing coordinator and several sensing collaborators, bidirectional communication capability among them is necessary. This is required in order to exchange both the sensing data ($S-DATA/S-DATA-ACK$) and coordination control information ($S-SCHED/S-SCHED-ACK$). The control messages, that is $S-SCHED/S-SCHED-ACK$, usually include the coordination scheduling scheme for sampling and gathering sensing information. The $S-DATA/S-DATA-ACK$ message exchanges between S_m and S_h eliminate spurious sensing events and reinforce genuine sensing events.

The role assignment control messages propagate a node's capability for a particular role by way of the $role - RA$ message and an assignment to it by its neighbors in the form of the $role - RA - ACK$ message. Figure 4.4 shows such an assignment for all the roles in the hierarchy. The query for a route to a destination is done among forwarders and routers by way of the Route–Request ($RREQ$) and Route–Reply ($RREP$) handshake. Similarly information about new routes is forwarded between the dominating router and the forwarders by way of the Route–Info ($R - INFO$) and the Route–Info–Ack ($R - INFO - ACK$) handshake. The sink or the base station queries is responded to by the sensing coordinator (S_h) by way of the Data-Aggregation-Request/Reply ($DAG - REQ/DAG - REP$) handshake.

The role-coordination graph (RCG) is thus the message level abstraction of the coordination mechanism among worker and manager roles organized in a hierarchy. This message-based coordination detail provides insights into the ramifications of a particular topological organization of roles in the network. With energy being the most important constraint in ad hoc wireless sensor networks, an empirical evaluation of the control overhead of mapping different role-graph configurations is possible. A prior analysis using this technique will serve to identify practical application requirements vis-a-vis the actual node deployment in terms of network size and capabilities. In the next section, we analyze generic rules for several roles.

4.2.6 Rules

The topological assignment of roles to a single node or a group of nodes is done by iteratively identifying the best node(s) among competing nodes in terms of resources needed to execute a role. We abstract the assignment or selection of roles by way of *rules*. We denote $rule_{role}^{[1...l]}$ as the set of l rules used to assign a *role*. Rule

sets select nodes per specified role-resource requirements. Thus rules involve quantitatively qualifying requisite resource capabilities in terms of node connectivity, energy, sensors and sensing redundancy for a specific role. The assignment of roles by way of rules is respectively defined as an explicit ordered list of desired resource requirements. We propose the following rules for the elementary roles discussed earlier:

- Sensor: We select the best node(s) for the sensor role using the following rules: (1) Sensor type, (2) Desired event detection tolerance (that is the number of sensors reporting the event), (3) Maximum energy available, (4) Nominal node degree (1 or more), and (5) Higher node ID.

- Forwarder: We select the best node(s) for the forwarder role using the following rules: (1) Non-redundant connectivity to disjoint nodes as neighbors, (2) Minimum distance, (3) Good link quality, (4) Maximum energy available, and (5) Higher node ID.

- Router: Among forwarders that connect two disjoint neighbors, we select the one that has connectivity to the maximum number of forwarders for the router role using the following rules: (1) Is a forwarder, (2) Maximum connectivity to disjoint forwarders, (3) Minimum hop connectivity, (4) Maximum energy available, and (5) Higher node ID.

- Processor: Among certain nodes that detect or sense the desired event and their k-hop neighbors ($k \leq 3$) that have free processing resources, we select the best nodes for the processor role using the following rules: (1) Available processor, (2) Higher processing speed, (3) Maximum energy available, (4) Requisite k-hop connectivity with roles like $\{S, C\}$, and (5) Higher node ID.

- Cacher: Among certain nodes that sense or detect the desired event and their k-hop neighbors ($k \leq 3$) that have free memory storage, we select the best nodes for the processor role using the following rules: (1) Available storage space, (2) Storage type, (3) Maximum energy available, (4) Requisite k-hop connectivity with roles like $\{S, C\}$, and (5) Higher node ID.

- Transmitter/Listener: Among nodes that assume roles like S, C, P, we need to select roles like T, L for scheduled communication among other roles using the following rules: (1) Is assigned a sensor, cacher, or processor role, (2) Maximum energy available, (3) Requisite k-hop connectivity with roles like $\{F, R\}$, and (4) Higher node ID.

In practical role configurations, we usually formulate complex roles and assign these to nodes instead of the elementary roles. This is because the elementary roles when assigned to nodes would require additional manager roles for coordination and management. Such an approach is overkill and is only used to elicit resource requirements for complex roles. It is also used to calculate the load imposed by a complex role on a node. This is accounted for as energy dissipated for respective combination of elementary roles for the time the role was assigned to the node.

These rules have to be applied recursively to remove redundancy and to achieve a minimum possible number of dominators that have the best resources to assume appropriate roles. Nodes that are not assigned any role in this selection process sleep to save energy.

4.2.7 Role Assignment (RA)

We extend the CDS based domination and elimination technique to have a generic scheme for recursive role assignment. This technique was discussed in chap-

ter 3 and was used to architect a role-based hierarchical organization (RBHSO). Following are the essential steps of the role-assignment process:

1. The role-assignment process starts by having sensors interact with others only in their local vicinity. This step is the neighbor discovery process and is used to identify the 1-hop neighbors for any node.

2. Each node performs very simple tasks such as maintaining and propagating information about relevant resource qualities. In this step, a node announces its initial capability for a certain role or roles by evaluating their respective rules and it awaits similar role announcements from its neighbors.

3. A greedy and recursive selection of the dominating nodes for a certain role with corresponding elimination of redundant nodes using the same rules results in the mapping of appropriate resource-specific roles at each iteration.

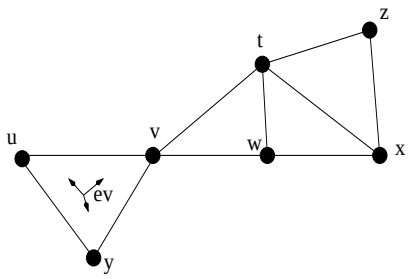

Figure 4.5: Data Aggregation: Example network for role assignment.

For example, in order to map a data aggregation service to a network of 7 sensor nodes in the vicinity of an event '*ev*' with node z as the *sink* shown in figure 4.5, the first step results in nodes exchanging their resource capabilities. With this exchange, nodes come to know of their 1-hop neighbors and their resource capabilities. They also come to know who is in proximity to the event *ev* and/or the

sink. In figure 4.6, we consider snapshots of the step-by-step mapping of elementary roles in the following order: $F \rightarrow S \rightarrow (C, P) \rightarrow (B, L)$. With the knowledge of complex role formulation in terms of elementary roles, the final step would involve assigning S_m, S_h, F, and R.

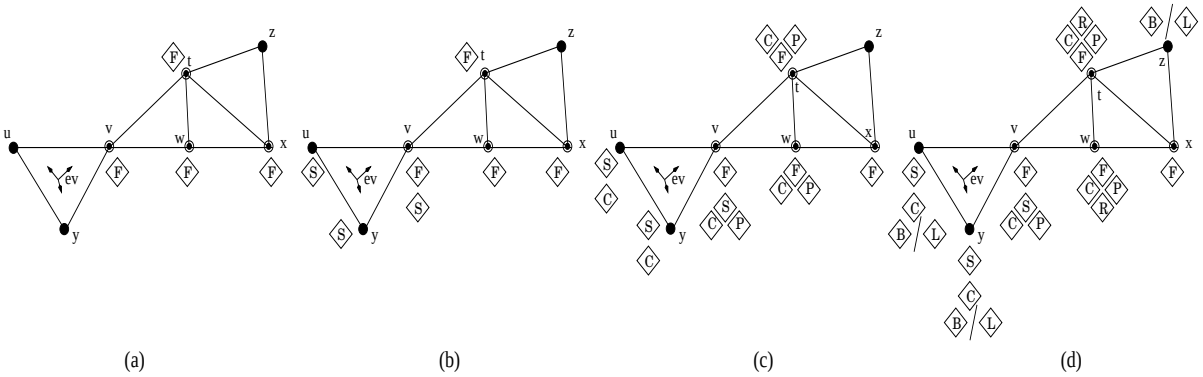

Figure 4.6: Role assignment snapshots: (a) Forwarder, (b) Sensor, (c) Cacher, Processor, (d) Beaconer, Listener, and Router.

Depending upon which role is selected and/or eliminated, it can be seen from figure 4.7 that the underlying topological organization adapts to changes in roles assumed by nodes. Thus for a tree-based organization shown in figure 4.7(a), nodes v, t, and z perform additional tasks compared to nodes u, y, w, and x. In other words, although nodes u, v, and y detect an event 'ev', only v takes an additional responsibility of forwarding the combined sensing data to its parent t and then eventually to sink z. Another case could be where all the nodes (except node u and sink node z) share equal responsibilities of aggregating and forwarding sensing data resulting in a chain-based organization shown in figure 4.7(b). Node u could also rotate responsibilities with y to perform network load balancing. This would result in the reorganization of the chain as $y \rightarrow u \rightarrow v \rightarrow \cdots \rightarrow z$. In a cluster (or CDS) based organization shown in figure 4.7(c), nodes v and t perform additional tasks of

cluster coordination and inter-cluster routing. Since the link between v and t serves as a critical backbone for communication among nodes 2-hop apart, load balancing is limited to controlling cluster membership and coordination.

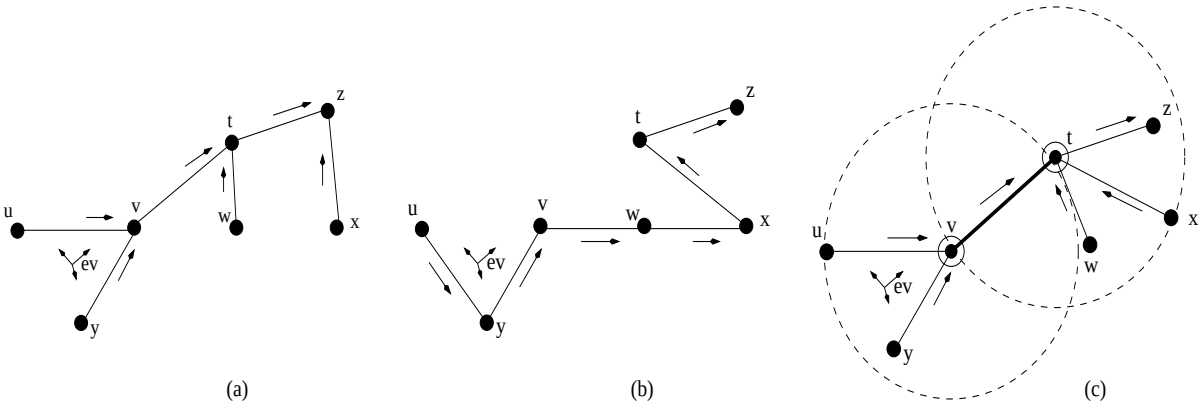

Figure 4.7: Sensor network organizations: (a) Tree, (b) Chain, and (c) Cluster (or CDS)

4.3 Domain specific models

To accommodate generic as well as specific domain dependencies, we propose models that incorporate these observations and simplify empirical evaluations of several role compositions for a particular service. We have identified three such models in the subsequent subsections.

4.3.1 Concave Role Service Utility (RS–ΔU) Model

The capability of both the network and the domain in which it is deployed to meet requirements specified by sensor network applications is a hard problem to solve. In other words, it is technically not possible for any ad hoc network to keep itself current with the available resources at any point in time. As discussed earlier

in section 4.1, it is not recommended to pursue an approach toward collecting, maintaining, and forwarding such global network information to the remote base station where services are usually requested from. This limitation complicates the problem of role assignment wherein the initial seed to the algorithm, that is the number of roles needed to execute the service, is missing. With the initial input to the role assignment algorithm missing, the convergence of the heuristics that optimize to meet an application requirement is not limited. The role-assignment heuristic thus needs to play with an artificial limit on the number of iterations. At times such an artificial limit may result in suboptimal assignment of roles to nodes in the network. It also makes the end-to-end performance evaluation of service delivery unreliable. With a change in the deployment domain, the solution becomes domain dependent and experimental. We propose a domain dependent model that approximates the resource requirements for meeting a specific service metric in terms of roles. With any change in domains, the URAF framework needs to update this model and the remaining components can remain unaltered.

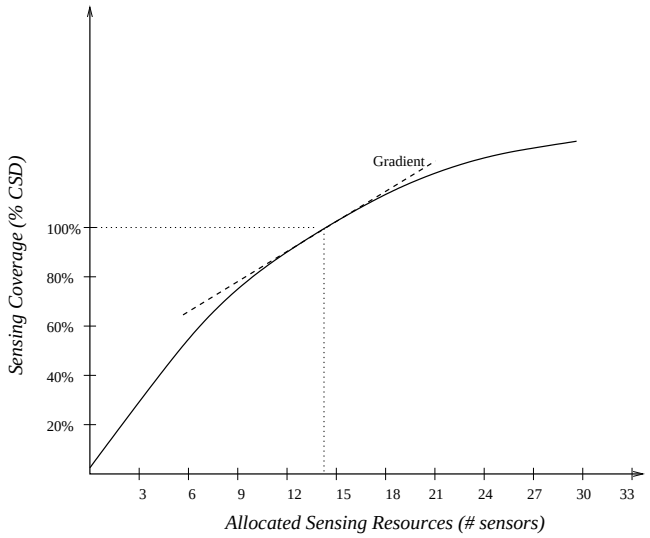

Figure 4.8: Marginal benefit to applications desiring sensing coverage

From the classical economic theories for resource allocation, it is observed that computing optimal resource allocations from sets of utility functions or service quality estimates is a *linearly constrained non-linear optimization problem*. In order to make the problem tractable, economic frameworks such as Game Theory and Mechanism Design constrain these utility functions to be *concave*. When applied to wireless sensor networks, it means that the marginal benefit of assigning additional resources e.g. sensors, network links, to a role configuration declines steadily and/or approaches zero: adding resources beyond some point does not result in meaningful improvement of service quality. From figure 4.8, we can see that the sensing capabilities (measured in terms of sensing coverage or cumulative sensing degree) does not really increase beyond a certain threshold. In the hypothetical concave domain model for sensing coverage utility versus the number of sensors, we see that to get 100% coverage, the number of sensor nodes needed is approximately 14. Similarly, one can develop models for communication latency versus number of hops, which may help in identifying the number of forwarders or routers needed to route data to the sink.

4.3.2 Role Energy (R–ΔE) Model

The role-energy model abstracts the domain-specific energy dissipation of elementary and complex role executions within a single node. With the role-based abstraction incorporating the in-network locality-based relationship between resources and tasks it becomes necessary to calculate the load imposed by a role on a node per unit time in terms of energy. In other words, the role assumed by node v at any time t (represented here as $role_t(v)$) can be perceived as the state of a node at time t.

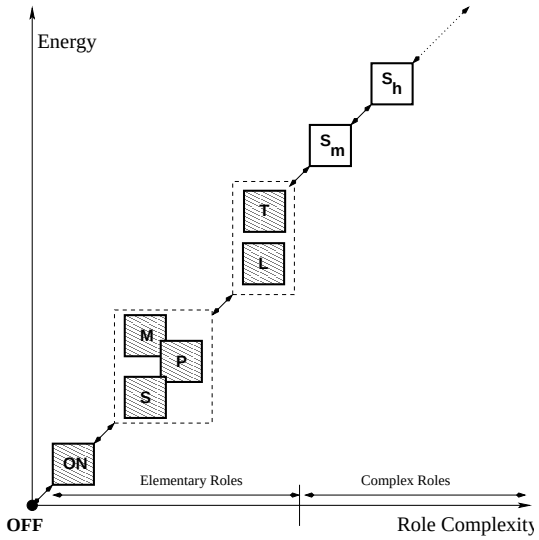

Figure 4.9: Hypothetical energy profile of various roles.

In order to profile the energy usage for each role, we need to analyze the local resources used by a role on a node including energy for performing a certain service. In order to account for unused resources by a role, we need to consider the fact that in smart low power sensor devices, it is possible to put predominantly unused resources into a low power mode where they draw less current. We account for temporary use of extra resources by accommodating them as relevant complex roles running at low power for that time. Thus complex roles can be used to account for additional power draw for both the used and unused resources for the respective time intervals.

We assume an energy model that can provide us with the cost per unit time for using the various sensor node actions (Shnayder et al., 2004). Thus, an energy model could specify the following costs:

1. c_{on}: Cost of turning ON a sensor node

2. c_{off}: Cost of turning OFF a sensor node

3. c_{idle}: Cost of an idle sensor node
4. c_s: Sensing Cost
5. c_p: Processing Cost
6. c_m: Memory storage Cost
7. c_t: Transmission Cost
8. c_l: Listening or receiving Cost

With this cost model, we can easily approximate the energy profile of various roles. A hypothetical energy profile is shown in figure 4.9.

4.3.3 Role Execution Time (R-ΔT) Model

The role-execution time model determines the approximate time needed for a standalone task to execute on an idle node. Thus, an execution-time model could specify the following ΔTs for the elementary roles:

1. ΔT_{on}: Time interval for turning ON a sensor node
2. ΔT_{off}: Time interval for turning OFF a sensor node
3. ΔT_s: Time interval for detecting an event
4. ΔT_p: Time interval for processing data in bits/second at a specific CPU clock rate (in Mhz)
5. ΔT_m: Time interval for storing data in flash memory in bytes/second
6. ΔT_t: Time interval for transmitting data on the radio transceiver at a specific bit rate

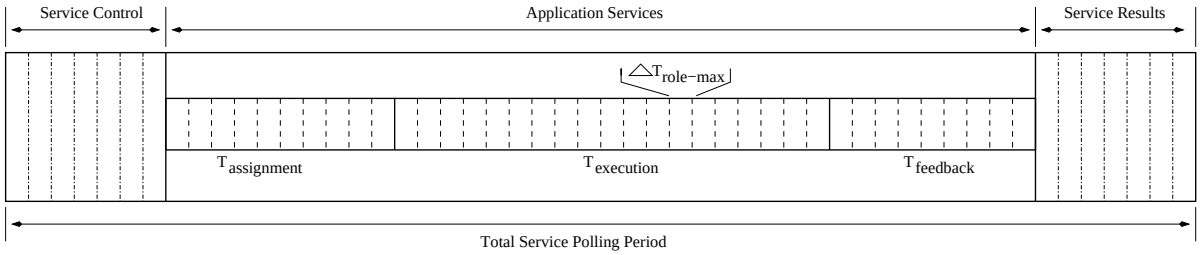

Figure 4.10: Role Execution Time Model: Application service schedule

7. ΔT_l: Time interval for listening to or receiving data on the radio transceiver at a specific bit rate

Since complex roles are assigned to nodes in the network, one can determine the number of slots needed to execute such a role on a node. The determination of the maximal number of slots needed for any role, sets the time slice for role execution, role assignment and role feedback. This limits the minimum interval an application can poll the network for services such as sensing, monitoring, and tracking. Figure 4.10 shows the determination of the service scheduling period based on the maximum time required for any standalone complex role (designated here as $\Delta T_{role-max}$) to execute on a node.

4.4 Design of the framework

In this section, we discuss the architecture of the role assignment framework as it unites the several aspects of the role-based abstraction along with the domain specific models discussed earlier. The model allows the collection and maintenance of several task, role, and service-related metrics of a node and its k-hop neighborhood, where it is recommended that $k \leq 3$ to limit message overhead and memory requirements. Such a central repository of state information and tuning interfaces

abstracted from specific role-abstraction levels would enable the role-assignment algorithm to incorporate relevant state attributes as rules in the assignment of roles to nodes. It would also allow roles to control or tune to the desired behavior in response to undesirable local node/network events. This is known as role load balancing and it is pursued as role reassignment to repair role failures. We will discuss role failures and role load balancing later in this section.

4.4.1 URAF architecture overview

Figure 4.11 shows the high level design architecture of the unified role-abstraction framework (URAF) in conjunction with a middleware (RBMW) that maps application specified services and expected QoS onto an ad hoc wireless sensor network with heterogeneous node capabilities.

The design of the framework is modular such that each module provides higher levels of network abstractions to the modules directly interfaced with it. For example, at the lowest level, we have API's that interface directly with the physical hardware. The resource usage and accounting module maintains up-to-date information on node and neighbor resource specifications and their availability. As discussed earlier, complex roles are composed of elementary roles and these are executed as tasks on the node. The state of the role execution at any point in time is cached by the task status table for that complex role. At the next higher abstraction, we calculate and maintain the overall role execution time and the energy dissipated by the node in that time. The available energy is thus calculated and cross checked against remaining battery capacity. There is another table that measures and maintains the failure/success of a role for every service schedule or period. This is used to calculate the load imposed by the service at different time intervals.

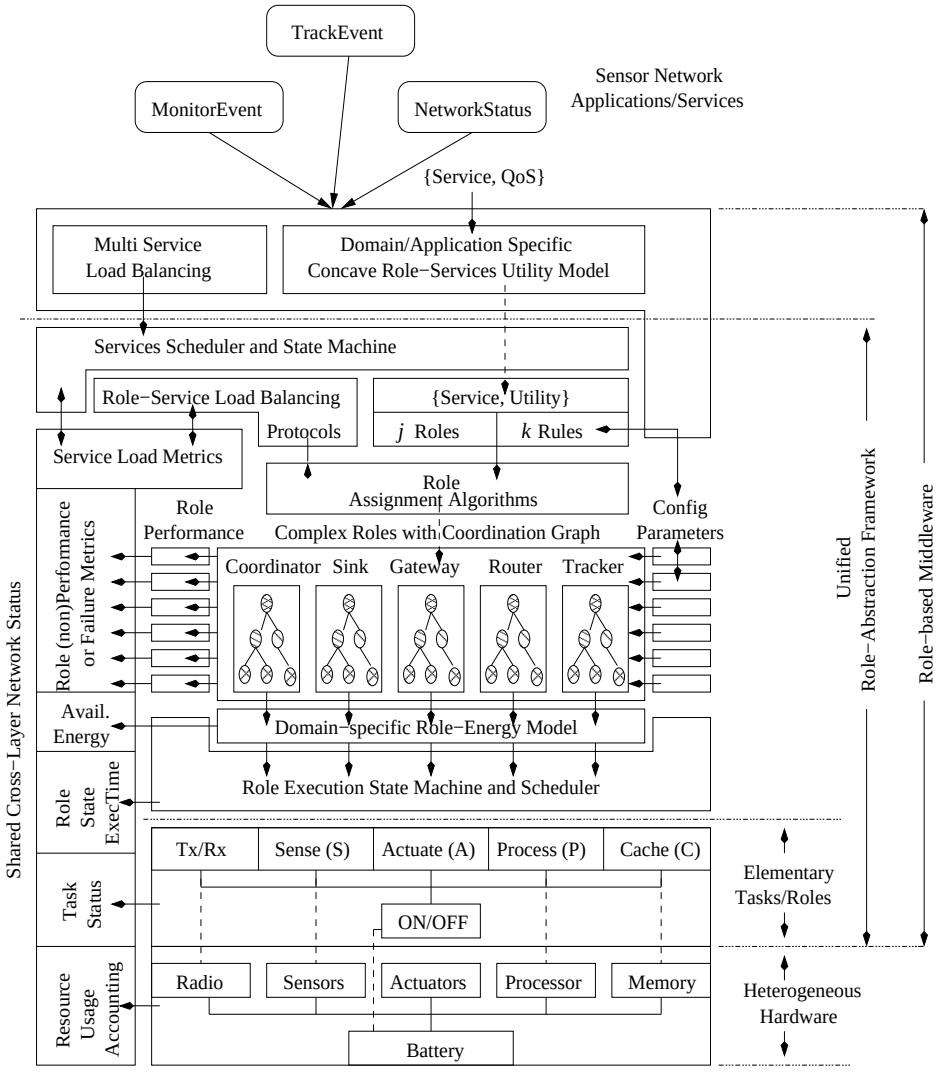

Figure 4.11: Design architecture of the Unified Role Assignment Framework

Role load balancing protocols try to balance the load across nodes in the network considering the energy dissipated per unit time by a node across services. Role load balancing protocols kicks off the role-assignment protocol to reassign roles in the network for balancing the load and avoiding hot spots. Similar strategies apply for multi-service load balancing where services are run for uniform energy usage. The domain specific RS–ΔU model provides initial inputs for the number of roles needed to meet an application requirement. With eventual degradation of the overall network lifetime, the model serves to provide the minimum number of roles needed to run the service on a best-effort basis.

The role based middleware layer can be implemented on the end user or operator side of the network. In other words, the domain specific model for service performance in terms of specific role formulations should be maintained and updated by the operator or the client of the network (i.e. the base station or the sink). A minimal server side of the RBMW interface can reside on every node in the ad hoc wireless sensor network. The server side interface is responsible for keeping the client up-to-date with how a service is performing in the network with a specific role composition that is currently being executed by a set of nodes in the network. Depending upon whether the network currently is in final deployment stage or preliminary experimental stages, the server nodes can be configured to log messages to the client (or the base station) at greater or minimal debugging severity levels respectively. The multi service load balancing component of RBMW could reside anywhere in the network. In other words, this component can be executed by specialized roles in the network that act as higher level managers for a specific region that is involved with executing multiple services. A minimal server side interface of this component can be implemented by appropriate worker nodes that are assigned

multiple roles for executing multiple services.

The URAF should be implemented completely by nodes in the network. To minimize the complexity of translating $\{Service, Utility\}$ request from the RS–ΔU model into appropriate roles and rules, the URAF framework can combine the model, the request specification, and the roles/rules into appropriate domain-specific service templates that maintain a mapping of QoS to utility to an appropriate set of roles and rules that can meet the desired requirement. These three templates, $\{Service, QoS\} \rightarrow \{Service, Utility\} \rightarrow \{roles, rules\}$ can be complemented by an internal mapping to an appropriate role-behavior template represented as role-coordination graphs (RCGs). In other words, $\{Service, QoS\} \rightarrow \{Service, Utility\} \rightarrow \{roles, rules\} \rightarrow \{role, RCG\}$. A further generalization is possible by including a specific role-assignment strategy along with the $\{role, RCG\}$ template. In other words, role-assignment strategies for different role configurations can be done in a different way. Similar approaches for role-reassignment protocols for load balancing can be conceived for different role configurations or RCGs responsible for executing different services.

The operational measurements in terms of specific performance attributes across different layers in the stack, need to be measured and maintained by node and their neighbors uniquely for respective domains. The role energy model is also domain specific and it depends upon the hardware platform and the composition of tasks for a specific role. Since the simple roles are fundamental formulation of the role-based abstraction, these do not change with time. Their respective role energy profile also do not change with time. Hence the role energy model is the invariant that is the same and is programmed exactly for all nodes before deployment. Depending upon the flexibility of the high level programming language, the end programmer can be

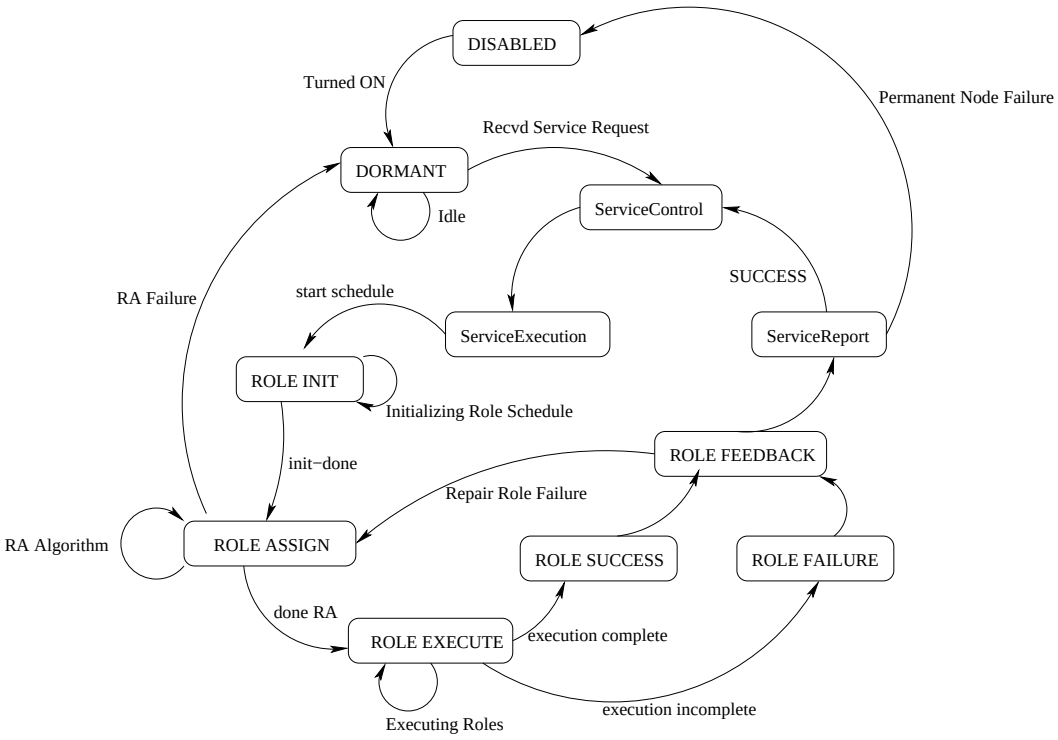

Figure 4.12: Role State Machine

given the flexibility to control role configurable parameters that are published in the $\{role, rule\}$ template. However, providing controls for role execution and its state machine entails a higher programming complexity and domain expertise from a programmer. Similar is the case for the procedure used to measure the operational performance that is maintained in a cross-layer status table. With the multi domain application of sensor networks, a minimal expectation is to have these measurement procedures documented by a standard committee. Only domain experts or system engineers should be given programming controls to design a modified measurement procedure for a specific performance metric.

4.4.2 Role state machine

The role state machine handles the behavior of the node as it goes through the process of executing a service through a periodic service schedule that consists of service feedback, service execution, and service reporting states. Within service execution, nodes formulate an application scheduling interval. In this interval, roles are initialized, assigned, and executed. The role execution stage then enters the feedback stage where the success or failure is reported at the time of service control. If the failure is irreversible or permanent, then the role state machine is permanently disabled else a repair is initiated which again kicks off the role-assignment process. If there is a role-assignment failure then the node decides to gracefully eliminate itself from participating in service execution and the role state machine transitions to the dormant state. Figure 4.12 shows the generic role state machine for any complex role that is to be executed as part of some service.

4.4.3 Role Failures

A wireless sensor network can be subject to several node or network variabilities. Some of these variabilities are temporary and these are usually caused by changes in the environment and/or network conditions. Of these temporary node/link failures, some of them may become permanent. Additionally, the software that guides the behavior of the node executing a particular role for a certain service has several constraints and dependencies. Mostly, constraints that are due to local state dependency among neighboring roles are usually temporal, spatial, and topological in nature. For example, a dependent role can fail to share vital state information with its dominating (or peer) role within a certain time duration. This failure, though local, can affect other dominating roles at a higher level in the hierarchical role

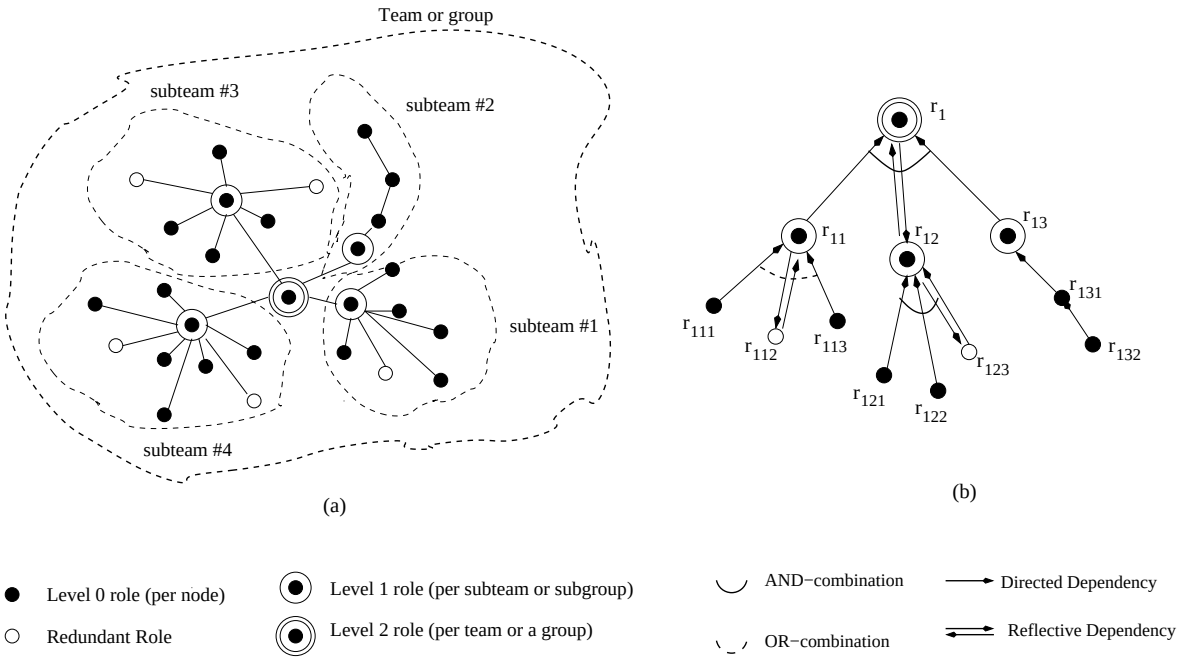

Figure 4.13: Monitoring Role failures: (a) Hierarchical Role-network organization and (b) Dominating roles acting as monitors for roles at lower level.

organization of a certain service. In the worst-case scenario, if the failure cannot be contained, its propagation can bring the whole service to a virtual halt.

Figure 4.13(a) shows an example mapping of a certain application service to a hierarchical role-based network organization. Figure 4.13(b) shows how dominating roles at a higher level in the hierarchy can act as *monitors* (Tambe, 1997) to track a groupmate's role performance or infer their role non-performance. For example, roles at a higher level can not only predict energy consumption of roles below but also act as arbitrators for fairness. However, it should be noted that accurately determining role performance is not easy as it is partly domain dependent and it may involve sharing a lot of contextual state information among roles. Depending upon the application requirements, the advantages of making a quick decision in a dynamically changing wireless sensor network based on approximate monitoring

quiet outweighs the overhead of delaying a decision based on the determination of an accurate value.

Figure 4.13(b) shows roles organized in terms of their dependencies. Intuitively, role failures in such an organization can occur in the following scenarios:

1. In an AND-combination where any sensor or subgroup fails in its role. For example, if one of the node(s) executing r_{11}, r_{12} or r_{13} fails, then the role r_1 fails.

2. An OR-combination when all the group members are role-dependent on a single individual or a single subgroup. For example, if roles r_{111} and r_{113} both fail then role r_{11} will fail. The role r_{112} is a redundant role and its failure will not affect the success of role r_{11}. However, role r_{112} can serve as a replacement when one of its equivalent roles r_{111} or r_{113} fail.

3. In a role dependency failure, where $role_j$ is dependent on $role_i$ ($role_i \rightarrow role_j$), failure of $role_i$ would mean that $role_j$ cannot execute. For example, if role r_{131} fails, then role r_{13} cannot execute. Also, since role r_{13} is also similarly dependent on role r_{132}, a candidate role say, r_x has to be determined such that $r_{132} \rightarrow r_x \rightarrow r_{13}$.

In the all these scenarios, a group or subgroup reconfiguration may be clearly warranted. This reconfiguration may involve determining and announcing candidate nodes for critical role substitution and deleting non-critical, conflicting, and redundant roles depending upon desired application requirements. A reactive approach where non-performing roles detect their failures and also determine and announce a candidate for role substitution would be preferable to a proactive one where neighbors periodically detect and correct each other's failures. Also, care

has to be taken to ensure that repairs or new role assignments for critical roles be confined to the local context of individual roles or sensors to avoid any global side effects.

4.4.4 Role reassignment and load balancing

Load balancing is pursued by the following two techniques. Both of them result in a repeated execution of the role-assignment algorithm. Hence, repetitive load balancing might be costly in a resource-constrained wireless sensor network and at times may outweigh the savings achieved by load balancing.

1. Pairwise Local Neighborhood Role Exchange: In this scheme, neighbors exchange roles according to their capabilities.

2. Pairwise Local Neighborhood Role Evolution: If the first scheme fails to find a replacement neighbor for a load intensive role, the node splits the complex roles into several simpler roles and role exchange is then pursued among nodes in the k-hop vicinity. A similar approach applies where disparate roles merge into a complex role and is assigned to the lowest loaded neighbor.

4.4.5 Role assignment strategies

The abstraction of services and the mapping criteria to nodes in terms of roles and rules respectively lends the URAF framework to support a variety of other role-assignment strategies. Here are the following role-assignment strategies:

1. Redundant (or naive) role-assignment technique: In this technique, given the number of roles and the geographic location where they should be assigned,

the strategy is to allow nodes to take over roles without any further arbitration. This algorithm is quite fast as it converges in O(1). However, it is not guaranteed that it achieves the requisite service performance. Usually this technique should be used to compare the performance of other RA strategies at various points in the network lifetime.

2. Greedy recursive dominating set based reduction technique: We have used this technique to organize a randomly deployed sensor network into a hierarchical role-based organization. Typically, such a technique should be used right after neighbor discovery and network (re)deployment. With such an organization, the network already has a list of dominators for assignment to load intensive roles. The candidate set for service-based role assignment is thus reduced and the role-assignment algorithm can converge (if possible) in fewer iterations.

3. Utility based role assignment by way of ranking: Given a global objective function that has to be optimized by a sensor network, suitable local role-utility functions are designed such that each sensor while "selfishly" optimizing its own local role-utility functions leads to sensors assuming roles that when collaboratively executed results in optimizing the desired global objective. A sensor's utility for a role can in fact be affected by the overall assignment of roles to sensors. The state-of-the-art for capturing inter-related utilities of this kind is the "*Markov Decision Process*", which is somewhat difficult to solve quickly enough for dynamic networks such as WSNs (especially when sensors have partial observability about the global network conditions are considered). Optimal role-assignment solutions that require higher overall optimal utility is impossible. This problem is similar to optimal scheduling. Usually utility based role assignment algorithms are modeled on Game

Theoretic and/or Mechanism design based strategies that require that Nash Equilibrium is achieved at the termination of the algorithm, where there is no role that a node can assume to maximize its utility. A common problem with dynamic role assignment is that small changes in utility estimates can cause roles to be reassigned very frequently, often in an oscillating fashion. Thus, the solution is to explicitly consider the cost of role transition in the utility computation, which will tend to induce a degree of hysteresis. This is usually done by adding a fixed amount of utility to the sensor retaining its current role.

4.5 URAF applications

4.5.1 Multi-objective Role-Assignment: MERA

An optimal algorithm for *RA* needs to map the services in such a way that it can simultaneously meet a number of objectives. For energy-constrained sensor networks, the most important objective is to have a minimum energy role assignment (*MERA*). As roles could be mapped to a single or a group of nodes, the *MERA* problem essentially translates to the minimization of the number of roles that are correspondingly being mapped to a topological organization that not only minimizes the number of flows among the roles but also leads to shortest paths communication among them. In other words, the solution to the *MERA* problem depends upon the following factors:

1. Minimum number of roles for a service,

2. Minimum number of nodes per role for a service,

3. Minimum dependency among roles that results in minimizing communication flows,

4. Shortest path connectivity among roles,

5. Minimum number of messages exchanged during any *RA* round, and

6. Minimum number of such *RA* rounds per service mapping.

The MERA problem is a MCDS problem where a set of dominators with desired properties need to identified in such a way that they form a minimum set. Also the dominators need to form a minimum connected backbone such that the non-dominator nodes are having at least one dominating node as their neighbor. Guha and Khuller (Guha and Khuller, 1996) firstly used the MCDS problem in general graphs to model the problem of computing a minimum size virtual backbone in heterogeneous wireless networks. Since a smaller size virtual backbone is expected to have less control overhead in terms of messages and reduced interference, the size of the CDS is considered as one of the major quality criteria in the literature. Since computing the MCDS is a well-known NP-hard problem, a combinatorial scheme that solves the MERA problem distributively is difficult and an optimal solution to this problem is NP-complete.

A centralized approach that solves for this many parameters using Linear/Dynamic Programming would anyways become suboptimal with minor changes in network dynamics. In order to adapt to these changes, such a centralized algorithm would have to be repeated a number of times thus making it prohibitively expensive both in terms of the role-reassignment overhead and its cost to gather updated network state information. Moreover, the problem of optimal role-assignment to meet desired service quality (QoS) is a problem that is domain specific and it needs detailed

and specific inputs in terms of service specifications that specify some of the above factors. In other words, factors 1, 2, and 3 can be generalized depending upon the type of service, its domain, and the network deployment on which it runs. Some or all of these can be specified by the base station application as inputs to the distributed *RA* algorithm. Thus, nodes need not waste energy solving for these parameters using partial network information.

In a random sensor network deployment, it is not necessary that having a minimum number of roles assigned to a minimum number of sensor nodes would guarantee minimization of the total energy expended in the execution of the service. For example, in a cluster-based organization, it is a well-known observation that having fewer clusterheads may result in cluster members expending a lot of energy to perform a long-haul communication to its nearest clusterhead. Similarly, minimizing dependency among roles would result in specialization of existing lightweight simpler roles to more complex roles that are more heavy-weight in terms of energy as they localize more tasks in their individual roles. In other words, some of these factors are in tradeoff with each other and it is difficult to find an optimal point where they converge. Also, such a point, if it exists, would change with the dynamics of the application and the underlying network on which it runs.

4.5.2 Energy-Latency (ΔE-ΔT) Minimization

The RA algorithm for ΔE-ΔT minimization, though NP-complete, can be modeled as follows:

Let us model a network of N nodes by an undirected graph $G = (V, E)$, where V is the set of nodes (vertices) and E is the set of links (edges). An edge between host pairs (v, u) indicates that both hosts v and u are within each others wireless

transmitter ranges. The corresponding undirected graph also known as a unit graph, thus has connections to hosts depending on their geographic distances.

For some node v, let us also denote $R(v)$ as its local set of resources consisting of $er(v)$ its remaining energy resource, $sr(v)$ its sensing resources, $mr(v)$ its available memory resources, $rr(v)$ its radio resource, and $pr(v)$ its processing resources; For battery-powered sensors, use of free resources is dependent upon available energy. Simply put, $R(v) = \{sr(v) \cup mr(v) \cup rr(v) \cup pr(v)\} \bigcap er(v)$. Let us denote, $role_t(v)$ as the role assumed by node v at time t. As discussed earlier, a role can also be perceived as the *state* of a node at time t. Finally, we denote $rule_{role}^{[1...l]}$ as the set of l rules used to assign a *role*. Rule sets select nodes per specified role-resource requirements.

Let us consider an arbitrary service involving n nodes $\{1, 2, 3, \ldots, n\}$ assuming m different roles $\{role_1, role_2, role_3, \ldots, role_i, \ldots, role_m\}$ in the vicinity of k hops for a time interval ΔT and if total $load_{\Delta T}^{role}(v)$ for a node v was normalized as the use of resources (R) with respect to energy, $er_{\Delta T}^{R}(v)$, then how does one minimize total energy expended and execution time for a particular network service.

$$load_{\Delta T}^{role} = \sum_{r}^{R} er_{\Delta T}^{r} \text{ where } r \in \{sr, mr, rr, pr\} \tag{4.1}$$

$$\Rightarrow load_{\Delta T}^{service} = \sum_{i=1}^{m} load_{\Delta T}^{role_i} \tag{4.2}$$

$$\therefore load_{\Delta T}^{service} = \sum_{v=1}^{n} \sum_{r}^{R} er_{\Delta T}^{r}(v) \tag{4.3}$$

4.5.3 Sensor Network Optimizations

The URAF lends itself to further optimization by incorporating the following general observations or experiences with sensor network applications. Utility-based modeling of optimization objectives that incorporate these observations have been considered for sensor networks by Byers and Nasser (Byers and Nasser, 2000). Similar approaches using weights and rules have been proposed in the research literature that we discussed in section 2.4 of chapter 2.

1. Energy dissipation increases with the number of sensors participating for a certain service. The URAF can support this in a number of ways either by reducing the number of roles or the number of nodes per role or both.

2. Not all subsets of sensors of a given size are created equal. In other words, two sensing regions with the same number of sensors are not necessarily equivalent from the perspective of their sensing capabilities. A rule based upon the CSD (Cumulative Sensing Degree) discussed in chapter 3 can be incorporated in the selection of the sensing collaborator.

3. Not all nodes need to contribute data to the computation and therefore nodes can conserve energy so that data aggregation can be performed over larger periods of time. Role scheduling along with assignment of the sensor and aggregator roles can be tweaked to meet this objective.

4. For most applications, it is not necessary to have the most highly optimized output at a given time step. Performing a small number of high quality sensing operations is not beneficial compared to completing a large number of computations over longer time scales. Fusing data from the maximum number of

sensors is a short-sighted approach in a power constrained environment. All of these can be achieved by the appropriate organization of worker and manager roles and the relationship among them for the data aggregation service in the role-coordination graph.

5. If a sensing region (or sector) is too large and contains many sensors, then the communication channel used by the sector manager may become saturated, affecting both the manager and any other local nodes that use the same channel. If the sector is too small, then track managers may spend excessive effort sending information to different sector managers as its target moves through the environment. The URAF can support this observation by an appropriate domain specific model for the role-execution time model. This model can make sure that complex roles involving significant coordination messages are spread out over a longer schedule. Limiting specific role groups such as within the sector during the role assignment process by way of rules can also support this observation.

6. As the number of sensors (roles) in each sector increase there is a corresponding marked increase in disparity of sharing responsibilities among sensors. This is because few roles are communicating much more than their peers. Thus, as the sector sizes scale, specialized nodes (or roles) become "hotspots" of activity. The URAF has a domain specific role energy model that accounts for the load imposed by a role on a node in terms of energy. The role state machine initiates the role reassignment whenever load imbalance occurs among nodes. Limits to actionable load imbalance can be set as thresholds that act as hysteresis to avoid repetitive role reassignment.

4.6 Summary: Features and Limitations

The Unified Role Assignment Framework has decomposed the problem of resource management into a number of sub-problems. These include role formulation for a service, its assignment strategy in terms of rules (that are based on metrics or utilities), its reassignment scenarios, and finally its scheduling for service execution.

The URAF abstracts away domain related dependencies, attributed largely to the heterogeneous sensor network platforms deployed in different environments, to domain specific models. There are models for role energy usage and the execution time needed for it to run on a node without blocking. There is also a model that a role designer or the network programmer may need to decide the composition of roles for services that are to be executed by the wireless sensor network. This model requires detailed in-field experimentation that involves measuring the service performance with various role compositions and role node assignments.

In general, with battery powered ad hoc wireless sensor network, pursuing energy conservation or network lifetime along with end-to-end delay are the two guiding factors for efficiently managing network resources. The domain specific role service utility model is therefore considered as a concave model wherein addition of extra resources (in terms of roles) beyond a certain point is not going to give the end user an appreciable increase in service performance (measured in terms of energy efficiency and end-user delay). The closeness of these three models to the reality of the domain, controls the leverage the programmer can get by using the APIs provided by the URAF. This, along with the way the lower layers measure metrics that are used as rules for role assignment, limit the practicality of the framework in a real deployment.

Chapter 5 – Summary and Future Work

The research in wireless sensor networks is currently oriented toward a cross-layer programming abstraction that allows programmers to perform appropriate fine-grained or coarse-grained energy efficient resource control using expressive programming language constructs and APIs. In that regard, we have identified the role-based service paradigm for wireless sensor networks. The concept of roles allows a programmer to deal with the assignment and scheduling of tasks to nodes or group of nodes in the network. We have proposed the Role-based Hierarchical Self-Organization (RBHSO) protocol as the underlying network architecture that allows programmers to construct an application specific architecture based on appropriate sensing and networking metrics. This concept of role-based organization is then extended to provide a generic framework where domain specific properties are abstracted away by three models: the role energy model, the role execution time model, and the role service utility model. The service formulation in terms of roles, its assignment, scheduling, and load balancing are further generalized by the framework as programming interfaces. The language for such a programming abstraction and the implementation of the framework (both its domain specific parts and its generic components) are left as a software engineering exercise for the future.

The rest of the chapter is organized as follows. We summarize our RBHSO protocol and the generic Unified Role Assignment Framework (URAF). We conclude the chapter with a discussion on directions for future research.

5.1 Summary

In this section, we summarize the design philosophy and the contributions of our role-based hierarchical self organization and the unified role assignment framework.

5.1.1 Role-based Hierarchical Self Organization (RBHSO)

The RBHSO protocol aims to implement a hierarchical network architecture that is customized for distributed sensing applications for ad hoc wireless networks. It considers the use of local information that is attributed in part to the limitation of the distributed, ad hoc, and wireless nature of the network to implement a CDS based algorithm that uses 1-hop neighborhood information exchanged over three rounds of neighboring communication. Using the recursive domination and elimination scheme based on appropriate application metrics, such as network connectivity, sensing coverage, and available energy, the RBHSO algorithm forms a hierarchy of selected high energy, high cumulative network and sensing degree nodes called dominators. The RBHSO algorithm also ends up partitioning the network into sets of sensing regions each having a set of dominators readily available for region maintenance, collaborative sensing, data aggregation and routing to a remote basestation or neighboring sink.

Additionally, the RBHSO protocol also delineates specific tasks necessary to perform the whole end-to-end user desired sensing service. The RBHSO protocol identifies dominators for specific tasks such as collaborative sensing, sensing coordination, and routing. A dominator node selected for any task thus ends up playing a certain 'role' in the network. The network is thus abstracted from the lowest level of 1-hop connectivity to a higher abstraction level consisting of a hierarchy of routers that connect regional dominators acting as sensing coordinators, which

in turn arbitrate the collaborative sensing and data aggregation activity among its regional members known as sensing collaborators.

Since a hierarchical CDS based organization is similar to clustering, we compare our approach to one of the popular clustering protocol known as LEACH. We identify specific topological metrics such as size of the clusters, the number of cluster heads, average distance between cluster member to its leader, average sensing coverage (translated as sensing degree) offered by any cluster or a sensing region (in RBHSO architecture). Simulation results confirm that an application specific organization geared toward optimal fault tolerant sensing with reduced energy consumption, require that the organization protocol be deterministic instead of being random (as it is in LEACH).

With increasing network sizes, it became clear that traditional clustering approaches where the number of clusters is predetermined or cluster selection is based on some random notion of ID or highest available energy does not necessary result in a balanced organization. We realize that a balanced organization is one that uses to its advantage the redundancy in both sensing degree and connectivity to construct an application specific organization that can satisfy sensing requirements while simultaneously saving redundant nodes for future use such that eventual network partitioning is delayed as much as possible. In this regard, we also realize the limitations of our RBHSO protocol that puts a threshold into the number of rounds of message exchange along with the constraint of exchanging only 1-hop neighbor information. These limitations constrain the ability of RBHSO protocol to consistently form a balanced network organization for larger and random network deployments, although it performs better, in terms of some of the metrics discussed earlier, when compared to a randomized clustering protocol such as LEACH.

5.1.2 Unified Role Assignment Framework (URAF)

The QoS perception at the end user of a wireless sensor network is mainly measured in terms of sensing fidelity of the readings relayed to it and and the ability of the network to provide these sensing and relaying services with lower power consumptions. At times, depending upon the nature of the sensing application, periodic or otherwise, delay expectations also become one of the QoS requirements. At the end user level, the service specification is simply the type of service desired and the varying levels of QoS, the user is willing to tolerate at different times during the service execution. Hence, user level specifications are usually declarative. Between the end user and the actual sensor network platform, the developer or the programmer of the sensor network desires a much higher programming expressibility and flexibility without its accompanying programming complexity, debugging, and maintenance. In essence, there is a need for an efficient cross-layer programming abstraction that allows programmers to perform an appropriate level of resource control depending upon their actual domain expertise while still being able to use simple constructs to support energy-efficient network services. We feel that the concept of roles allows natural and adaptively expressive constructs for a programmer to deal with sensor network programming according to their level of comfort. In this regard, we extend the RBHSO protocol to develop a Unified Role Abstraction Framework (URAF) to model application services as roles performed by local in-network sensor nodes with sensor capabilities used as rules for role identification.

We discussed the underlying networking concepts of aggregation, hierarchy, approximation, and redundancy upon which the URAF is based. We use these concepts to develop generic roles for elementary tasks and use these to compose complex roles that abstract protocol specific interactions among nodes. We abstract these

complex protocol task interactions in the spatial and temporal domains by way of role-coordination graphs that highlight this need for dependency and subsequent coordination among roles. Using an example domain-specific empirical model for energy consumption, we profile the use of various node resources such as radio, sensing, battery, memory, and computation for both elementary and complex roles. This role service load profiling in terms of energy allows nodes to pursue application load balancing by way of an adaptive assignment and scheduling of roles to nodes. For example, with declining network resources roles become more adaptive as they evolve toward less energetic types. In this scenario, a node attempts to extend its remaining lifetime by arbitrating services to a much larger pool of neighbors. This facilitates reconfiguration and the subsequent reassignment of complex roles to a number of simpler roles.

The URAF can be implemented as a Role-based Middleware (RBMW). The components that are part of the framework can be implemented according to the flexibilities provided by the platform, both hardware and software. The basic design of the components and the functionalities used and/or provided to other components within the framework have been discussed. This allows a language designer to support appropriate declarative constructs for roles, rules, metrics, role assignment, and scheduling. The end user can also be given declarative constructs to specify application service requirements and tradeoffs either as a simple or as weighted sum of utilities and points in QoS space. The RBMW can then map these requirements in terms of a specific assignment of roles to nodes. The RBMW can also incorporate example load balancing strategies outlined by the framework across roles, nodes, and services in terms of pairwise neighborhood role exchange, role mergers, and role redirection.

5.2 Future work

With regard to the experiences gained while designing the solutions and understanding their limitations for both the RBHSO protocol and the URAF framework, we feel the research could progress further in a number of directions:

- A hybrid hierarchical organization architecture seems relevant for very large scale sensor networks (VLSNs). These are networks with hundreds of thousands of sensors deployed in a very large area (maybe a city block or even larger). In such an architecture, several levels of hierarchy could be organized with dominators at alternate levels in the hierarchy acting as routers. These routers could be organized in a tree like fashion that are rooted to the nearest proximal sink. Dominators above such routers could be organized in a virtual backbone and can be addressed on a larger regional basis (akin to blocks in the city). The lower three levels of the organization could be similar to the RBHSO architecture where sensors are grouped into sensing regions that are coordinated by sensing coordinators, which then report to sector (or regional) managers. In tracking applications, these managers, in addition to performing data aggregation from several regions, could also coordinate the tracking service to several nearby sensing regions.

- Micro-economic optimization approaches such as utility based decision making, game theory, and mechanism design are alternative role assignment techniques that should be explored in detail in the context of wireless sensor networks. In that regard, domain specific studies are needed to understand the capability of the system to meet application objectives specified by a combination of performance metrics as points in the QoS space. Multi-objective

specification can be generalized by a QoS specification expressed in terms of a weighted set of utilities. Similar to the three models proposed in the URAF, specific models should be abstracted for other QoS metrics for each domain. The measurement of QoS metrics at different layers of the sensor network protocol stack need to be studied and also generalized for every application service deployed in their respective domains. These are problems that have not been addressed within the sensor network community. One reason for this is the lack of wide scale technology adoption both the academia and the industry, and the absence of detailed studies for all such deployments.

- Sensor network programming is a difficult proposition if the programming abstraction has to support both ease of programmability and the ability to support low-level cross-layer controls for achieving the desired performance. This is because of the multi-domain application of wireless sensor networks. We feel the role-based service paradigm is very relevant to supporting programming at different domain expertise levels. Future research efforts can be directed toward developing a generic role programming language for network abstractions that implement the URAF.

- With the availability of a multitude of sensor network platforms (both hardware and software) and a variety of algorithms in the literature for efficient performance of several sensor network services, it is unmanageable even for a domain expert to achieve programming tractability. Under these conditions, it is expected that for every application domain and sensor platform, a set of templates can be standardized for a variety of network sizes and application performance levels. This should relieve programmers of the technical domain expertise, and will also give them a starting template of a recom-

mended algorithm and parameters to tweak for a specific application. Future research efforts will be aimed at the standardization of service templates for heterogeneous sensor networks.

- Similar to the ubiquitous Internet, the generic role-based framework could serve as a foundation for seamless integration of large-scale sensor societies. Several new roles could be identified that allow inter-societal service interactions. For example, we could draw an analogy between sensor networks and human society, where government officials stand for administrative roles, police for security roles, the post office for message delivery, etc. The possibilities are endless and left only to the imagination of the researchers. Standardization efforts are thus warranted in the WSN arena for the universal adoption of roles.

- A long-term goal would be to provide a cross-platform portable middleware library for a variety of sensor network applications. In the future, as major advancements are made in the fields of circuit integration, microprocessors, memory, and battery power, we envision that an embedded virtual machine that interprets the role-based middleware to the native hardware language will make our vision possible. Also, as multiple applications use a universal language for service specifications, it paves the way for future standardization efforts and a unified framework for concurrent application specification, development, and deployment.

Appendix A – Protocol Pseudocode

A.1 Role-Based Hierarchical Self Organization

Algorithm A.1.1: NetworkInitialization($myId, radioRange$)

comment: Discover node neighbors

global $mySensingRange, sensingAccuracy, myCSD$

$(x, y) \leftarrow$ estimateNodePosition()

sendHelloMessage($radioRange$)

while waitForHelloRspTimeout()

do
- **if** rcvdHelloRspEvt($helloRspMsg$)
  - **then** $nbrs[i] \leftarrow$ getNbrFromHelloRsp($helloRspMsg$)
- $i \leftarrow i + 1$

return $(nbrs, myCSD)$

Algorithm A.1.2: MARKINGPROCESS($myId, radioRange, nbrs$)

comment: Localized node marking process

$amIMarked \leftarrow$ **false**

SENDNBRINFOMSG($nbrs$)

while WAITFORNBRINFORSPTIMEOUT()

do {
  if RCVDNBRINFORSPEVT($nbrInfoRspMsg$)
    then $nbrs[i].nbrs \leftarrow$ GETNBRNBRS($nbrInfoRspMsg$)
  $i \leftarrow i + 1$

while $i > 0$ & $amIMarked =$ **false**

do {
  if $canConnectTwoNbrs(myId, nbrs[i].nbrs)$
    then {
      $amIMarked \leftarrow$ **true**
      exit
  $i \leftarrow i - 1$

return ($amIMarked$)

Algorithm A.1.3: USEREDUCTIONONE($myId, amIMarked, nbrs$)

comment: Reduction one to eliminate (level-0) markers

$amIR1Dominating \leftarrow$ **false**

if $amIMarked =$ **true**

then while $i > 0$ & $amIR1Dominating =$ **false**

do
- **if** $myNbrsIncludeNbrsOf(i) =$ **true**
  - **then**
    - **comment:** Break tie in order:
    - $brokeTie \leftarrow$ USEENERGYRULE($myEnergy$) **or**
    - USECONNECTIVITYRULE($myNodeDegree$) **or**
    - USEIDRULE($nbrId$)
    - **if** $brokeTie =$ **true** $amIR1Dominating \leftarrow$ **true**
    - **exit**
- $i \leftarrow i - 1$

if $amIR1Dominating =$ **true** SENDR1INFOMSG($nbrs$)

return ($amIR1Dominating$)

Algorithm A.1.4: USEREDUCTIONTWO($myId, amIR1Dominating, nbrs$)

comment: Reduction two to eliminate level-1 dominators

$amIL2Dominating \leftarrow$ **false**

if $amIR1Dominating =$ **true**

then while $i > 0$ & $amIR2Dominating =$ **false**

do
- **if** $myTwoNbrsIncludeNbrsOf(i) =$ **true**
  - **then**
    - **comment:** Break tie in order:
    - $brokeTie \leftarrow$ USEENERGYRULE($myEnergy$) **or**
      USECONNECTIVITYRULE($myNodeDegree$) **or**
      USEIDRULE($nbrId$)
    - **if** $brokeTie =$ **true** $amIR2Dominating \leftarrow$ **true**
    - **exit**
- $i \leftarrow i - 1$

if $amIR2Dominating =$ **true** SENDR2INFOMSG($nbrs$)

return ($amIR2Dominating$)

Algorithm A.1.5: DOMSETREDUCTION($myId, radioRange, nbrs$)

comment: Dominating set reduction process

$amIDominating \leftarrow$ **false**

if $nbrs$ **not** $= 0$

then $amIMarked \leftarrow$ MARKINGPROCESS($myId, radioRange, nbrs$)

if $amIR1Dom =$ **true**

then $amIR1Dom \leftarrow$ USEREDUCTIONONE($myId, amIMarked, nbrs$)

if $amIR1Dom =$ **true**

then $amIR2Dom \leftarrow$ USEREDUCTIONTWO($myId, amIR1Dom, nbrs$)

if $amIR2Dom =$ **true**

then
- $amIDominating \leftarrow$ **true**
- **comment:** Send Domination Info to Neigbors
- SENDDOMINFOMSG($amIDominating, nbrs$)

return ($amIDominating$)

Algorithm A.1.6: amIProbableSensorCoordinator($myId, nbrs$)

comment: Algorithm to select among dominators a sensing coordinator

$amICoordinator \leftarrow$ **false**

$amIDominating \leftarrow$ domSetReduction($myId, radioRange, nbrs$)

if $amIDominating = TRUE$

then while $i > 0$ & $amICoordinator =$ **false**

do
- **if** $haveNbrsAsDominators(i) =$ **true**
  - **then**
    - **comment:** Break tie in order:
    - $brokeTie \leftarrow$ useEnergyRule($myEnergy$) **or**
    - useCumulativeSensingDegreeRule($myCSD$) **or**
    - useIdRule($nbrId$)
    - **if** $brokeTie =$ **true** $amICoordinator \leftarrow$ **true**
    - **exit**
- $i \leftarrow i - 1$

if $amICoordinator =$ **true**

then
- **comment:** Send Sensing Coordination Info to Neighbors
- sendScoordInfoMsg($amICoordinator, nbrs$)

return ($amICoordinator$)

Algorithm A.1.7: FORMSENSINGZONE($myId, nbrs$)

comment: Procedure to form sensing zones

```
amICoordinator ← AMIPROBABLESENSORCOORDINATOR(myId, nbrs)
SOLICITMEMBERSFORSENSINGZONE(myId, myCSD)
if no members solicited
  then JOINNEARESTSENSORCOORDINATOR(myId)
if sensing zone members less than specified minimum membership
  then { FINDNEIGHBORINGSENSINGZONE()
         if found neighboring sensing zone
           then MERGEWITHNEIGHBORINGSENSINGZONE()
           else DISSOLVESENSINGZONE()
if I am still orphan
  then JOINANYNEIGHBORINGSENSORCOORDINATOR()
 or JOINANYNEIGHBORINGSENSINGZONEMEMBER()
```

Algorithm A.1.8: MAIN($myId, nbrs$)

comment: Main RBHSO procedure

main

for each $sensor \in \mathcal{S}$

do
- **local** $amImarked, amICoordinator$
- **local** $currentHierarchyLevel, LevelsMarked$
- $amImarked \leftarrow$ **false**
- $amISensorCoordinator \leftarrow$ **false**
- $myNeighbors \leftarrow$ NETWORKINITIALIZATION($myId, radioRange$)
- $myCSD \leftarrow$ ESTIMATECUMULATIVESENSINGDEGREE()
- $currentHierarchyLevel \leftarrow 0$
- $markme \leftarrow$ MARKINGPROCESS()
- $currentHierarchyLevel \leftarrow 1$
- **while** I do not get the same set of dominating neighbors

  do
  - $markme \leftarrow$ DOMSETREDUCTION()
  - **if** markme == **true**
    then $LevelsMarked[currentHierarchyLevel] \leftarrow$ **true**
  - **if** currentHierarchyLevel == 1
    then FORMSENSINGZONE()
  - $currentHierarchyLevel \leftarrow currentHierarchyLevel + 1$

References

T. Abdelzaher, B. Blum, Q. Cao, et al. EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks. In *ICDCS, Tokyo, Japan*, March 2004.

F. Adelstein, G. Richard, and L. Schwiebert. Building dynamic multicast trees in mobile networks. In *ICPP Workshop on Group Communication*, pages 17–22, September 1999.

I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks (Elsevier)*, March 2002.

A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh. Max-Min D-Cluster Formation in Wireless Ad Hoc Networks. In *Proceedings of IEEE INFOCOM'2000, Tel Aviv*, March 2000.

V. Annamalai, S. K. S. Gupta, and L. Schwiebert. On tree-based convergecasting in wireless sensor networks. In *IEEE Wireless Communications and Networking Conference*, volume 3, pages 1942–1947, March 2003.

D. J. Baker and A. Ephremides. The Architectural Organization of a Mobile Radio Network via a distributed algorithm. *IEEE Transactions on Communications*, COM-29(11):1694–1701, November 1981.

A. Bakshi, V. Prasanna, J. Reich, and D. Larner. The Abstract Task Graph: A methodology for architecture-independent programming of networked sensor systems. In *Workshop on End-to-end Sense-and-respond Systems (EESR'05)*, 2005.

A. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees (CBT). In *ACM SIGCOMM*, pages 85–95, September 1993.

L. Bao and J.J. Garcia-Luna-Aceves. Topology Management in Ad Hoc Networks. In *4th ACM International Symposium on Mobile Ad hoc Networking and Computing (MOBIHOC 2003), Annapolis, MD, USA*, June 2003.

G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, 1999.

M. Bhardwaj and A. Chandrakasan. Bounding the Lifetime of Sensor Networks via Optimal Role Assignments. In *INFOCOM*, 2002.

S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han. MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms. *Mobile Network Applications*, 10(4): 563–579, 2005.

U. Bischoff and G. Kortuem. A state-based programming model and system for wireless sensor networks. In *Proceedings of the 5th International Conference on Pervasive Computing and Communications*, 2007.

L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J. Hubaux, and J. Le Boudec. Self-organization in mobile ad-hoc networks: the approach of terminodes. *IEEE Communications Magazine*, 39(6):166–174, June 2001.

B. Blum, P. Nagaraddi, A. Wood, T. Abdelzaher, S. Son, and J. Stankovic. An Entity Maintenance and Connection Service for Sensor Networks. In *MobiSys, San Francisco, CA*, May 2003.

P. Bonnet, J. Gehrke, and P. Seshadri. Querying the physical world. *IEEE Personal Communications*, 7(5):10–15, 2000.

C. Boonthum, I.B. Levinstein, S. Olariu, L. Wang, and Q. Xu. Assurance-aware Self-organization of Sensor Networks. In *Proceedings of the IASTED International Conference on Networks and Communication Systems (NCS-2006), Chiang Mai, Thailand*, 2006.

C. Borcea, C. Intanagonwiwat, P. Kang, U. Kremer, and L. Iftode. Spatial programming using smart messages: Design and implementation. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, 2004.

S. C. Botelho and R. Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'99)*, pages 1234–1239, 1999.

A. Boulis, C.-C. Han, and M. B. Srivastava. Design and implementation of a framework for efficient and programmable sensor networks. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys'03)*, pages 187–200, 2003.

D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Pro-*

ceedings of the 1st ACM International Workshop on Wireless Sensor Networks, pages 22–31, 2002.

R. Brooks, P. Ramanathan, and A. Sayeed. Distributed target classification and tracking in sensor networks. In *Proceedings of the IEEE*, volume 91, pages 1163–1171, August 2003.

N. Bulusu. *Self-Configuring Localization Systems*. PhD thesis, University of California, Los Angeles (UCLA), 2002.

J. Byers and G. Nasser. Utility-based decision-making in wireless sensor networks. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking and computing*, pages 143–144, 2000.

M. Ceriotti, L. Mottola, G. Picco, A. Murphy, S. Guna, M. Corrá, M. Pozzi, D. Zonta, and P. Zanon. Monitoring heritage buildings with wireless sensor networks: The Torre Aquila deployment. In *Proceedings of the 8th International Symposium on Information Processing in Sensor Networks (IPSN'09)*, 2009.

H. Chan and A. Perrig. ACE: An emergent algorithm for highly uniform cluster formation. In *Proceedings of the 1st European Workshop on Wireless Sensor Networks (EWSN)*, pages 154–171, 2004.

C. Chang and H. Chang. Energy-aware node placement, topology control and mac scheduling for wireless sensor networks. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 52(11):2189–2204, 2008. ISSN 1389-1286. doi: http://dx.doi.org/10.1016/j.comnet.2008.02.028.

B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An Energy-Efficient

Coordination Algorithm for Topology Maintenance in Ad hoc Wireless Networks. In *Proceedings of MobiCom 2001*, pages 70–84, July 2001.

G. Chen, F. Garcia, J. Solano, and I. Stojmenovic. Connectivity Based k-hop Clustering in Wireless Networks. In *Proceedings of the IEEE Hawaii Int. Conf. System Science*, Jan 2002.

C. Chevallay, R. E. Van Dyck, and T. A. Hall. Self-organization Protocols for Wireless Sensor Networks. In *Thirty Sixth Conference on Information Sciences and Systems*, March 2002.

M. Chiang. Balancing transport and physical layers in wireless multihop networks: jointly optimal congestion control and power control. 23(1):104–116, Jan 2005.

H. J. Choe, P. Ghosh, and S. K. Das. Cross-layer design for adaptive data reporting in wireless sensor networks. In *IEEE International Conference on Pervasive Computing and Communications (PerCom'09)*, 2009.

D. Chu, L. Popa, A. Tavakoli, J. Hellerstein, P. Levis, S. Shenker, and I. Stoica. The design and implementation of a declarative sensor network system. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys'07)*, 2007.

P. Ciciriello, L. Mottola, and G.P. Picco. Building virtual sensors and actuator over Logical Neighborhoods. In *Proceedings of the 1st International ACM Workshop on Middleware for Sensor Networks (MidSens'06)*, 2006.

L. P. Clare, G. J. Pottie, and J. R. Agre. Self-Organizing Distributed Sensor Networks. In *Proc. SPIE, Unattended Ground Sensor Technologies and Applications*, volume 3713, pages 229–237, 1999.

P. Costa, L. Mottola, A. Murphy, and G. Picco. Programming wireless sensor networks with the TeenyLime middleware. In *Proceedings of the 8th International USENIX/ACM Conference on Middleware*, 2007.

CPLEX. CPLEX Online. URL `www.cplex.com`.

CrossBow. Crossbow technology inc.: smart sensors in silicon, san jose, california. URL `http://www.xbow.com/`.

S. Cui, R. Madan, A. Goldsmith, and S. Lall. Joint routing, mac, and link layer optimization in sensor networks with energy constraints. In *IEEE ICC 05*, volume 2, pages 725–729, May 2005.

D. Culler, P. Dutta, C. T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao. Towards a sensor network architecture: Lowering the waistline. In *International Workshop on Hot Topics in Operating Systems (HotOS)*, 2005.

T. Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of ACM 1st International Conference on Embedded Networked Sensor Systems (SenSys03)*, pages 171–180, Nov. 2003.

K. Dasgupta, M. Kukreja, and K. Kalpakis. Topology-aware placement and role assignment for energy-efficient information gathering in sensor networks. In *Proceedings of the eight IEEE International Symposium on Computers and Communications*, 2003.

F. S. H. de Souza and G. R. Mateus. Exact and heuristic approaches for role assignment problem in wireless sensor networks, 2006.

Z. Drezner and H. W. Hamacher. *Facility location: applications and theory*. Springer-Verlag, Berlin, 2004. ISBN 3540213457.

A. Dunkels, O. Schmidt, T. Voigt, and M. Ali. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys'06)*, pages 29–42, 2006.

R. E. Van Dyck. Detection performance in self-organized wireless sensor networks. In *IEEE International Symposium on Information Theory, Lausanne, Switzerland*, June 2002.

EnviroTrack: an Enviromental Programming Paradigm for Sensor Networks. EnviroTrack Homepage. URL `http://www.cs.virginia.edu/\texttilde{ll4p}/EnviroTrack/`.

D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *ACM MOBICOM'99*, 1999.

D. Estrin et al. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. RFC 2362, Internet Engineering Task Force, June 1998.

P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of Publish/Subscribe. *ACM Computing Surveys*, 2(35), 2003.

Y. Fang and A.B. McDonald. Dynamic Codeword Routing (DCR): A Cross-Layer Approach for Performance Enhancement of General Ad Hoc Routing. In *Proceedings of the First IEEE International Conference of Sensor and Ad Hoc Communication Networks (SECON), Santa Clara, California*, pages 255–263, October 2004.

D. Ferrara, L. Galluccio, A. Leonardi, G. Morabito, and S. Palazzo. MACRO: An Integrated MAC/Routing Protocol for Geographical Forwarding in Wireless Sensor Networks. In *IEEE Infocom*, pages 1770–1781, March 2005.

D. A. Fisher and H. F. Lipson. Emergent Algorithms: A New Method for Enhancing Survivability in Unbounded Systems. In *Proceedings of the Hawaii International Conference On System Sciences*, January 1999.

C. Frank and K. Romer. Algorithms for generic role assignment in wireless sensor networks. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys'05)*, 2005.

C. Frank and K. Romer. Solving generic role assignment exactly. In *Proceedings of the 14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS'06)*, 2006.

S. Ganeriwal, R. Kumar, and M. Srivastava. Timing-sync protocol for sensor networks. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 138–149, November 2003.

D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. An empirical study of epidemic algorithms in large scale multihop wireless networks. Technical Report TR-795, Intel Research, February 2002.

D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *In PLDI03: Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2003.

D. Gelernter. Generative communication in Linda. *ACM Computing Surveys*, 7(1), 1985.

B. P. Gerkey and M. J. Matarić. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18:758–768, 2002.

B. Greenstein, E. Kohler, and D. Estrin. A sensor network application construction kit (SNACK). In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*, pages 69–80, 2004.

S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. Technical Report 3660, Univ. of Maryland Inst. for Adv. Computer Studies, Dept. of Computer Science, Univ. of Maryland, College Park, June 1996.

R. Gummadi, O. Gnawali, and R. Govindan. Macro-programming wireless sensor net- works using Kairos. In *Proceedings of the 2nd International Conference on Distributed Computing in Sensor Systems (DCOSS'05)*, 2005.

S. K. S. Gupta and P. K. Srimani. An adaptive protocol for reliable multicast in mobile multi-hop radio networks. In *IEEE Workshop on Mobile Computing Systems and Applications*, pages 111–122, 1999.

J. Haapola, Z. Shelby, C. Pomalaza-Raez, and P. Mahonen. Cross-layer energy analysis of multi-hop wireless sensor networks. In *EWSN*, pages 33–44, 2005.

Z.J. Haas, M.R. Pearlman, and P. Samar. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. Internet Draft draft-ietf-manet-zone-zrp-04.txt, Internet Engineering Task Force, July 2002.

W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *International Conference on System Sciences*, January 2000.

W. Heinzelman, A. Murphy, H. Carvalho, and M. Perillo. Middleware to Support Sensor Network Applications. *IEEE Network Magazine Special Issue*, Jan 2004.

J. Hill. A software architecture supporting networked sensors. Master's thesis, University of California at Berkeley, 2000.

J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *In ASPLOS-IX: Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

Y-W. Hong, L. F. Cheow, and A. Scaglione. A simple method to reach detection consensus in massively distributed sensor networks. In *Proceedings of the International Symposium on Information Theory*, pages 250–255, 2004.

M. Inanc, M. Magdon-Ismail, and B. Yener. Power Optimal Connectivity and Coverage in Wireless Sensor Networks. Technical Report 03–06, Rensselaer Polytechnic Institute, Dept. of Computer Science, Troy, New York, 2003.

C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking, Boston, MA, USA*, pages 56–67, August 2000.

K. Jamieson, H. Balakrishnan, and Y. C. Tay. Sift: A MAC protocol for event-driven

wireless sensor networks. Technical Report MIT-LCS-TR-894, Massachusetts Institute of Technology, 2003.

K. Jamshaid and L. Schwiebert. Seken (secure and efficient key exchange for sensor networks). In *IEEE Performance Computing and Communications Conference (IPCCC)*, April 2004.

P. Juang, H. Oki, Y. Wang, M. Martonsi, L. S. Peh, and D. Rubenstein. Energy efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X'02)*, pages 96–107, 2002.

V. Kawadia and P. R. Kumar. A Cautionary Perspective on Cross Layer Design. *submitted to IEEE Wireless Communication Magazine*, 2004.

M. Kochhal, L. Schwiebert, and S. K. S. Gupta. Role-based hierarchical self organization for ad hoc wireless sensor networks. In *WSNA, San Diego, CA, USA*, pages 98–107, September 2003.

M. Kochhal, L. Schwiebert, and S. K. S. Gupta. Integrating sensing perspectives for better self organization of ad hoc wireless sensor networks. *Journal of Information Science and Engineering*, 20(3), May 2004.

J. Koshy and R. Pandey. VMSTAR: Synthesizing scalable runtime environments for sensor networks. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys'05)*, pages 234–254, 2005.

P. Krishna, N. H. Vaidya, M. Chatterjee, and D. Pradhan. A cluster-based approach

for routing in dynamic networks. *ACM SIGCOMM Computer Communication Review*, 27(2):49–65, April 1997.

B. Krishnamachari, S. Wicker, R. Bejar, and C. Fernandez. On the complexity of distributed self-configuration in wireless networks. *Telecommunication Systems*, 22(1-4):33–59, 2003.

R. Krishnan and D. Starobinski. Message-Efficient Self-Organization of Wireless Sensor Networks. In *IEEE WCNC 2003*, 2003.

M. Kubisch, H. Karl, A. Wolsz, L. Zhong, and J.Rabaey. Distributed Algorithms for Transmission Power Control in Wireless Sensor Networks. In *IEEE Wireless Communications and Networking Conference (WCNC), New Orleans, LA*, March 2003.

R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, and U. Ramachandran. DFuse: A Framework for Distributed Data Fusion. In *SenSys, Los Angeles, CA, USA*, pages 114–125, Nov. 2003.

P. Levis and D. Culler. Maté: A tiny virtual machine for sensor networks. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOSX'02)*, pages 89–95, 2002.

P. Levis, D. Gay, and D. Culler. Active sensor networks. In *Proceedings of the 2nd USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI'05)*, 2005a.

P. Levis, D. Gay, V. Handziski, J.-H Hauer, B. Greenstein, M. Turon, J. Hui, K. Klues, C. Sharp, R. Szewczyk, J. Polastre, P. Buonadonna, L. Nachman,

G. Tolle, D. Culler, and A. Wolisz. T2: A second generation os for embedded sensor networks. Technical Report TKN-05-007, Telecommunication Networks Group, Technische Universitat Berlin, Berlin, Germany, 2005b.

S. Li, Y. Lin, S. Son, J. Stankovic, and Y. Wei. Event detection services using data service middleware in distributed sensor networks. *Telecommunication Systems*, 26(2), 2004.

S. Liang and H. Dimitrios. A cross-layer architecture of wireless sensor networks for target tracking. *IEEE/ACM Transactions on Networking (TON)*, 15(1):145–158, 2007. ISSN 1063-6692. doi: http://dx.doi.org/10.1109/TNET.2006.890084.

T. Liu and M. Martonsi. Impala: A middleware system for managing autonomic, parallel sensor systems. In *Proceedings of the Ninth ACMSIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'03)*, pages 107–118, 2003.

L. Luo, T. Abdelzaher, T. He, and J. Stankovic. EnviroSuite: An environmentally immersive programming framework for sensor networks. *IEEE Transactions on Embedded Computing Systems*, 5(3), 2006.

R. Madan, S. Cui, S. Lall, and A. Goldsmith. Cross-layer design for lifetime maximization in interference-limited wireless sensor networks. In *IEEE INFOCOM05*, volume 3, pages 1964–1975, March 2005.

S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 30(1):122–173, 2005.

P. J. Marrón, A. Lachenmann, D. Minder, J. Hähner, R. Sauter, and K. Rothermel. Tinycubus: A flexible and adaptive framework for sensor networks. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN'05)*, 2005.

W. P. Mccartney and N. Sridhar. Abstractions for safe concurrent programming in networked embedded systems. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys'06)*, pages 167–180, 2006.

S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage Problems in Wireless Ad-Hoc Sensor Networks. In *IEEE INFOCOM '01*, volume 3, pages 1380–1387, April 2001a.

S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure In Wireless Ad Hoc Sensor Networks. In *ACM SIGMOBILE (Mobicom)*, pages 139–150, July 2001b.

T. Melodia, M. C. Vuran, and D. Pompili. The state of the art in cross-layer design for wireless sensor networks,in. In *Proceedings of EuroNGI Workshops on Wireless and Mobility, Springer Lecture Notes on Computer Science, LNCS 388*, 2005.

C. J. Merlin and W. B. Heinzelman. X-lisa: A cross-layer information-sharing architecture for wireless sensor networks. Technical Report, University of Rochester, CS dept., December 2006.

C. J. Merlin and W. B. Heinzelman. Network-aware adaptation of mac scheduling for wireless sensor networks. In *In Proc. 3rd Conf. on Distributed Computing in Sensor Systems (DCOSS07 Poster Session)*, June 2007.

K. L. Mills. A Brief Survey of Self-Organization in Wireless Sensor Networks. *Wireless Communications and Mobile Computing (WCMC)*, 7:823–834, May 2007.

J. Mirkovic, G. P. Venkataramani, S. Lu, and L. Zhang. A Self Organizing approach to Data Forwarding in Large Scale Sensor Networks. In *IEEE International Conference on Communications (ICC'01), Helsinki, Finland*, June 2001.

J. Moore, T. Keiser, R. Brooks, S. Phoha, D. Friedlander, J. Koch, A. Reggio, and N. Jacobson. Tracking targets with self-organizing distributed ground sensors. In *IEEE Areospace Conference*, pages 2113–2123, 2003.

L. Mottola and G. Picco. Logical neighborhoods: A programming abstraction for wireless sensor networks. In *Proceedings of the 3rd International Conference on Distributed Computing in Sensor Systems (DCOSS'06)*, pages 150–168, 2006.

L. Mottola and G. P. Picco. Programming Wireless Sensor Networks: Fundamental Concepts and State of the Art. *ACM Computing Surveys (to appear)*, 2010.

R. Newton and M. Welsh. Region streams: Functional macroprogramming for sensor networks. In *Proceedings of the 1st International Workshop on Data Management for Sensor Networks*, 2004.

R. Newton, Arvind, and M. Welsh. Building up to macroprogramming: An intermediate language for sensor networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, pages 37–44, 2005.

R. Newton, G. Morrisett, and M. Welsh. The Regiment macroprogramming system. In *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN'07)*, 2007.

S. Ni, Y. Tseng, Y. Chen, and J. Chen. The Broadcast Storm Problem in a mobile ad hoc network. In *in Proceedings of MOBICOM*, pages 151–162, August 1999.

C. Nitta, R. Pandey, and Y. Ramin. Y-threads: Supporting concurrency in wireless sensor networks. In *Proceedings of the 2nd International Conference on Distributed Computing in Sensor Systems (DCOSS'06)*, pages 169–184, 2006.

M. Ocean, A. Bestavros, and A. Kfoury. snBench: Programming and virtualization framework for distributed multitasking sensor networks. In *Proceedings of the 2nd International Conference on Virtual Execution Environments (VEE'06)*, 2004.

S. Olariu, Q. Xu, and A.Y. Zomaya. An energy-efficient self-organization protocol for wireless sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing Conference (ISSNIP)*, pages 55–60, December 2004.

E. Pagani and G. P. Rossi. Reliable Broadcast in Mobile Multihop Packet Networks. In *Proceedings of the Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 34–42, 1997.

N. A. Pantazis, D. D. Vergados D. J. Vergados, and C. Douligeris. Energy efficiency in wireless sensor networks using sleep mode tdma scheduling. 7(2):322–343, March 2009.

J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of 2nd International Conference on Embedded Networked Systems (SENSYS)*, 2004.

J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica. A unifying link abstraction for wireless sensor networks. In *Proceedings of the 3rd*

international conference on Embedded networked sensor systems (SenSys'05), San Diego, CA, USA, pages 76–89, 2005.

I. Raicu, O. Richter, L. Schwiebert, and S. Zeadally. Using wireless sensor networks to narrow the gap between low-level information and context-awareness. In *International Conference on Computers and Their Applications*, pages 209–214, April 2002.

C. V. Ramamoorthy, A. Bhide, and J. Srivastava. Reliable Clustering Techniques for Large, Mobile Packet Radio Networks. In *Proceedings of IEEE INFOCOM'87*, pages 218–226, 1987.

R. Ramanathan and R. Rosales-Hain. Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment. In *Proceedings of IEEE INFOCOM 2000, Tel-Aviv, Israel*, March 2000.

T. G. Robertazzi and P. E. Sarachik. Self-Organizing Communication Networks. *IEEE Communications*, 24(1):28–33, 1986.

J. Ó Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, Oxford, 1987. ISBN 0195039653.

A. Rowe, R. Mangharam, and R. Rajkumar. RT-link: A global time synchronized link protocol for sensor networks. *Ad hoc Networks, Elsevier*, 6(8):1201–1220, 2008.

F. Royo, T. Olivares, and L. Orozco. A synchronous engine for wireless sensor networks. *Wireless Sensor and Actor Networks, IFIP, Springer*, 248(40):107–118, 2007.

C. S. Raghavendra S. Lindsey. PEGASIS: Power Efficient GAthering in Sensor Information Systems. In *IEEE Aerospace Conference*, March 2002.

A. Salhieh and L. Schwiebert. Power aware metrics for wireless sensor networks. In *IASTED International Conference on Parallel and Distributed Computing and Systems*, pages 326–331, November 2002.

P. Santi. Topology Control in Wireless Ad Hoc and Sensor Networks. In *submission to ACM Comp. Surveys*, 2003.

C. Schurgers, V. Tsiatsis, and M. Srivastava. STEM: Topology Management for Energy Efficient Sensor Networks. In *IEEE Aerospace Conference*, pages 78–89, March 2002.

L. Schwiebert, S. K. S. Gupta, J. Weinmann, A. Salhieh, M. Kochhal, and G. Auner. Research challenges in wireless networks of biomedical sensors. In *MOBICOM, Rome, Italy*, pages 151–165, July 2001.

K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari. Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In *ACM SenSys'04*, November 2004.

C. Shen, C. Srisathapornphat, and C. Jaikaeo. Sensor information networking architecture and applications. In *IEEE Personal Communications*, pages 52–59, Aug. 2001.

E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical layer driven protocol and algorithm design for energy- efficient wireless sensor networks. In *Proceedings of the ACM MobiCom 2001, Rome, Italy*, July 2001.

V. Shnayder, M. Hempstead, B. Chen, G. Werner-Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04), Baltimore, MD*, November 2004.

M. L. Sichitiu. Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks. In *Proceedings of INFOCOM 2004, Hong Kong (PRC)*, March 2004.

S. Singh, M. Woo, and C. S. Raghavendra. Power aware routing in mobile ad hoc networks. In *in Proceedings of MOBICOM*, pages 181–190, 1998.

R. Sivakumar, B. Das, and V. Bharghavan. An Improved Spine-based Infrastructure for Routing in Ad hoc Networks. In *IEEE Symposium on Computers and Communications '98, Athens, Greece*, June 1998.

P. Skraba, H. Aghajan, and A. Bahai. Cross-layer optimization for high density sensor networks: Distributed passive routing Decisions. In *Ad-Hoc Now, Vancouver*, July 2004.

S. Slijepcevic and M. Potkonjak. Power Efficient Organization of Wireless Sensor Networks. In *IEEE International Conference on Communications (ICC'01), Helsinki, Finland*, pages 472–476, June 2001.

K. Sohrabi and G. Pottie. Performance of a Novel Self-Organization Protocol for Wireless Ad hoc Sensor Networks. In *50th IEEE Vehicle Technology Conference, The Netherlands*, September 1999.

K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie. Protocols for Self Organization of a Wireless Sensor Network. In *IEEE Personal Communication Magazine*, volume 7, pages 16–27, October 2000.

L. Song and D. Hatzinakos. A cross-layer architecture of wireless sensor networks for target tracking. *IEEE/ACM Transactions on Networking (TON)*, 15(1):145–158, 2007. ISSN 1063-6692. doi: http://dx.doi.org/10.1109/TNET.2006.890084.

M. Steenstrup, B. Beranek, and Newman. *Ad hoc Networking*, chapter Cluster-Based Networks, pages 75–135. Addison-Wesley, first edition, December 2000. ISBN 0201309769.

P. Stone and M. Veloso. Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. *Artificial Intelligence*, 110:241–273, 1999.

L. Subramanian and R. H. Katz. An architecture for building self-configurable systems. In *IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC 2000), Boston*, August 2000.

R. Sugihara and R. K. Gupta. Programming models for sensor networks: A survey. *ACM Transactions on Sensor Networks (TOSN)*, 4(2), March 2008.

M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

D. G. Thaler and C. V. Ravishankar. Distributed Top-Down Hierarchy Construction. In *Proceedings of IEEE INFOCOM*, pages 693–701, 1998.

D. Tian and N. D. Georganas. A Coverage-Preserving Node Scheduling Scheme for Large Wireless Sensor Networks. In *Proceedings of ACM Workshop on Wireless Sensor Networks and Applications (WSNA'02), Atlanta*, October 2002.

Y. Tian and E. Ekici. Cross-layer collaborative in-network processing in multi-hop wireless sensor networks. 6(3):297–310, March 2007.

Y. Tian, E. Ekici, and F. Ozguner. Cluster-based information processing in wireless sensor networks: An energy-aware approach. 7(7):893–907, September 2007.

Y.-C Tseng, Y.-N Chang, and B.-H Tzeng. Energy Efficient Topology Control for Wireless Ad hoc Sensor Networks. In *23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03), Providence, Rhode Island, USA*, May 2003.

L. van Hoesel, T. Nieberg, J. Wu, and P. J. M. Havinga. Prolonging the lifetime of wireless sensor networks by cross-layer interaction. *IEEE Communications*, 11 (6):78–86, December 2004.

P. K. Varshney. *Distributed Detection and Data Fusion*. Springer-Verlag, New York, 1996. ISBN 0387947124.

L. Victor. Scaling up multi-agent systems through organizational structuring. In *WI '04: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 5–5, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2100-2. doi: http://dx.doi.org/10.1109/WI.2004.125.

M. Vinyals, J. A. Rodriguez-Aguilar, and J. Cerquides. A survey on sensor networks from a multiagent perspective. *The Computer Journal*, 2010. URL `http://comjnl.oxfordjournals.org/cgi/content/abstract/bxq018v1`.

M. C. Vuran and I. F. Akyildiz. Spatial Correlation-based Collaborative Medium Access Control in Wireless Sensor Networks. *IEEE/ACM Transactions on Networking (TON)*, June 2006.

M. C. Vuran, O. B. Akan, and I. F. Akyildiz. Spatio-Temporal Correlation: Theory

and Applications for Wireless Sensor Networks. *Computer Networks Journal (Elsevier)*, 45(3):245–261, June 2004.

M. C. Vuran, V. B. Gungor, and O. B. Akan. On the interdependency of congestion and contention in wireless sensor networks. In *Proceedings of SENMETRICS*, July 2005.

H. Wang, K. Yao, G. Pottie, and D. Estrin. Entropy-based sensor selection heuristic for target selection. In *Proceedings of Information Processing in Sensor Networks (IPSN)*, pages 36–45, 2004.

I. Warrier, A. Aia, J. Min, and M. L. Sichitiu. Z-mac: a hybrid mac for wireless sensor networks. pages 511–524, June 2008.

R. Wattenhofer, Li Li, P. Bahl, and Yi-Min Wang. Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad Hoc Networks. In *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Anchorage, Alaska*, April 2001.

M. Welsh and G. Mainland. Programming sensor networks with abstract regions. In *USENIX/ACM Symposium on Network Systems Design and Implementation*, 2004.

G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal. Firefly-inspired sensor network synchronicity with realistic radio effects. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 142–153, 2005.

K. Whitehouse, C. Sharp, E. Brewer, and D. Culler. Hood: A neighborhood abstraction for sensor networks. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys'04)*, pages 99–110, 2004.

P. Winter. Steiner Problem in Networks: A Survey. *Networks*, 17(2):129–167, 1987.

J. Wu. *Handbook of Wireless and Mobile Computing*, chapter Dominating Set Based Routing in ad hoc wireless networks, pages 425–450. John Wiley, 1st edition, Feb. 2002. ISBN 0471419028.

J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 7–14, August 1999.

G. Xing, M. Sha, G. Hackmann, K. Klues, O. Chipara, and C. Lu. Towards unified radio power management for wireless sensor networks. *Wireless Communications and Mobile Computing*, 9(3):313–323, 2009. ISSN 1530-8669. doi: http://dx.doi.org/10.1002/wcm.v9:3.

H. Xu, L. Huang, J. Wu, Y. Wang, J. Wang, and X. Wang. *Lecture Notes in Computer Science: Mobile Ad-hoc and Sensor Networks*, chapter Self Organization Data Gathering for Wireless Sensor Networks, pages 650–661. Springer Berlin, Heidelberg, November 2006. ISBN 03029743.

Y. Xu. *Adaptive Energy Conservation Protocols for Wireless Ad hoc Routing*. PhD thesis, University of Sourthern California (USC), 2002.

Y. Xu, J. Heidemann, and D. Estrin. Geography Informed Energy Conservation for Ad hoc Routing. In *ACM/IEEE International Conference on Mobile Computing and Networking*, pages 70–84, July 2001.

Y. Xu, S. Bien, Y. Mori, J. Heidemann, and D. Estrin. Topology control protocols to conserve energy inwireless ad hoc networks. Technical Report 6, University of

California, Los Angeles, Center for Embedded Networked Computing, January 2003. submitted for publication.

Y. Yao and J. Gehrke. The Cougar approach to in-network query processing in sensor networks. *ACM Special Interest Group on Management Of Data (SIGMOD'02)*, 31(3), 2002.

F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *International Conference on Mobile Computing and Networking (MobiCOM)*, pages 148–159, September 2002a.

F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route filtering of injected false data in sensor networks. *IEEE Journal on Selected Areas in Communications*, 23 (4):839–850, 2004.

W. Ye, John Heidemann, and Deborah Estrin. An Energy Efficient mac protocol for Wireless Sensor Networks. In *IEEE INFOCOM, New York, USA*, pages 3–12, June 2002b.

W. Yu and K. Liu. Attack-resistant cooperation stimulation in autonomous ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 23(12):2260–2271, 2005.

Y. Yu, L. J. Rittle, V. Bhandari, and J. B. Lebrun. Supporting concurrent applications in wireless sensor networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys'06)*, pages 139–152, 2006.

J. Yuan, Z. Li, W. Yu, and B. Li. A cross-layer optimization framework for multicast

in multi-hop wireless networks wireless internet. In *Proc. WICON 05*, pages 47–54, July 2005.

W. Zhang and G. Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. In *IEEE INFOCOM*, pages 2434–2445, 2004.

ZigBee. The zigbee alliance. URL `http://www.zigbee.org`.

M. Zorzi and R. Rao. Geographic random forwarding (GeRaF) for ad hoc and sensor networks: multihop performance. *IEEE Transactions on Mobile Computing I*, 2 (4):337–348, December 2003.

M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *Proceedings of IEEE SECON*, pages 517–526, October 2004.

ABSTRACT

UNIFIED ROLE-ASSIGNMENT FRAMEWORK FOR WIRELESS SENSOR NETWORKS

by

MANISH M KOCHHAL

August 2010

Advisor: Dr. Loren Schwiebert

Major: Computer Engineering

Degree: Doctor of Philosophy

Wireless sensor networks are made possible by the continuing improvements in embedded sensor, VLSI, and wireless radio technologies. Currently, one of the important challenges in sensor networks is the design of a systematic network management framework that allows localized and collaborative resource control uniformly across all application services such as sensing, monitoring, tracking, data aggregation, and routing.

The research in wireless sensor networks is currently oriented toward a cross-layer network abstraction that supports appropriate fine or course grained resource controls for energy efficiency. In that regard, we have designed a unified role-based service paradigm for wireless sensor networks. We pursue this by first developing a Role-based Hierarchical Self-Organization (RBSHO) protocol that organizes a

connected dominating set (CDS) of nodes called dominators. This is done by hierarchically selecting nodes that possess cumulatively high energy, connectivity, and sensing capabilities in their local neighborhood. The RBHSO protocol then assigns specific tasks such as sensing, coordination, and routing to appropriate dominators that end up playing a certain 'role' in the network.

Roles, though abstract and implicit, expose role-specific resource controls by way of role assignment and scheduling. Based on this concept, we have designed a Unified Role-Assignment Framework (URAF) to model application services as *roles* played by local in-network sensor nodes with sensor capabilities used as *rules* for role identification. The URAF abstracts domain specific role attributes by three models: the role energy model, the role execution time model, and the role service utility model. The framework then generalizes resource management for services by providing abstractions for controlling the composition of a service in terms of roles, its assignment, reassignment, and scheduling. To the best of our knowledge, a generic role-based framework that provides a simple and unified network management solution for wireless sensor networks has not been proposed previously.

Keywords: Wireless sensor networks (WSNs), sensing metrics, roles, rules, network organization, role assignment, role scheduling, and generic control interfaces or abstractions.

AUTOBIOGRAPHICAL STATEMENT

Manish Kochhal is currently a Ph.D. candidate in the Networking Wireless Sensors Laboratory (NeWS Lab) in the Department of Computer Science at Wayne State University. He received his MS degree in Computer Engineering from Wayne State University, Detroit, Michigan and his BE degree in Electronic Engineering from University of Bombay, Bombay, India. He is also currently working as a Senior Software Engineer at Airvana Inc., Chelmsford, MA. His areas of research interests include QoS support in wireless networks and developing efficient communication protocols for large scale embedded systems.

Manish is a member of IEEE and ACM. He can be reached at manishk@wayne.edu.