

# Distributed Services for Information Dissemination in Self-Organizing Sensor Networks \*

Alvin Lim

*Department of Computer Science and Engineering, Auburn University,  
Auburn, AL 36849*

---

## Abstract

Dynamic enterprise systems, such as the battlefield, use self-organizing sensor network infrastructure to gather and disseminate real-time information for controlling the enterprise. Very large number of highly mobile sensor data sources and users may be scattered over a wide area with little or no fixed network support. These large surveillance sensor networks must adapt rapidly to dynamic changes in sensor nodes configuration. Dynamic query processing and target tracking through this unstructured sensor network of surveillance information sources and users must use the appropriate distributed services and network protocols to solve the problems of mobility, dispersion, weak and intermittent disconnection, dynamic reconfiguration, and limited power availability. We provide three main distributed services: lookup service, composition service, and dynamic adaptation service. Through a distributed implementation of these services, other application-specific network and system services can be defined spontaneously in the sensor network. They also enable dynamic adaptation of these services to incremental addition and removal of sensor nodes, device failure and degradation, migration of sensor nodes, and changing requirements in tasks and networks. When placed together impromptu, sensor nodes should immediately know about the capabilities and functions of other smart nodes and work together as a community system to perform coordinated tasks and networking functionalities.

*Keywords:* Self-organizing sensor networks, reconfigurable smart nodes, distributed services, dynamic reconfiguration

---

## 1 Introduction

Many dynamically changing enterprise systems, such the battlefield and commercial inventory and distribution systems, must be controlled using adaptive methods

---

\*This research is supported in part by the Space and Naval Warfare Systems Center, Department of Navy.

that utilize critical, real-time information gathered from integrated low-powered sensors and mobile devices [14, 28, 25] deployed throughout the enterprise. These mobile and miniaturized information devices are equipped with embedded processors, wireless communication circuitry, information storage capability, smart sensors and actuators. These sensor nodes networked in an ad-hoc way, with little or no fixed network support, to provide the surveillance and targeting information for dynamic control of the enterprise. Sensor devices are mobile, subject to failure, deployed spontaneously and repositioned for more accurate surveillance. Despite these dynamic changes in configuration of the sensor network, critical real-time information must still be disseminated dynamically from mobile sensor data sources through the self-organizing network infrastructure to the components that control dynamic re-planning and re-optimization of the theater of operation based on newly available information.

With large number of sensor devices being quickly and flexibly deployed in most impromptu networks, each sensor device must be autonomous and capable of organizing itself in the overall community of sensors to perform coordinated activities with global objectives. When spontaneously placed together in an environment, these sensor nodes should immediately know about the capabilities and functions of other sensor nodes and work together as a community system to perform cooperative tasks and networking functionalities. Sensor networks need be self-organizing since they are often formed spontaneously from large number of mixed types of nodes and may undergo frequent configuration changes. Some sensor nodes may provide networking and system services and resources to other sensor nodes. Others may detect the presence of these nodes and request services from them.

The characteristics of sensor nodes necessary for creating self-organizing sensor networks are agility, self-awareness, self-configurability and autonomy. Sensor nodes with these features will have capabilities for self-assembling impromptu networks that are incrementally extensible and dynamically adaptable to device failure and degradation, mobility of sensor nodes, and changes in task and network requirements. Nodes are aware of their own capabilities and those of other nodes around them which may provide the networking and system services or resources that they need. Although nodes are autonomous, they may cooperate with one another to disseminate information or assist each other in adapting to changes in the network configuration. An impromptu community of these nodes may cooperate to provide continual coordinated services while some nodes may be newly deployed or removed from the spontaneous community.

Unlike traditional well-structured computer networks where connecting devices to the network using relatively static network configuration is often cumbersome, sensor networks are unstructured and consist of very large number of sensors. Building self-organizing sensor networks is difficult for the following main reasons. First, many different types of sensors with a range of capabilities may be deployed with different specialized network protocols and application requirements. Data-centric network protocols are becoming common in sensor network [9, 8]. With many mixed types of sensors and applications, sensor networks may need to support several data-centric network protocols simultaneously. Second, these mixed type of

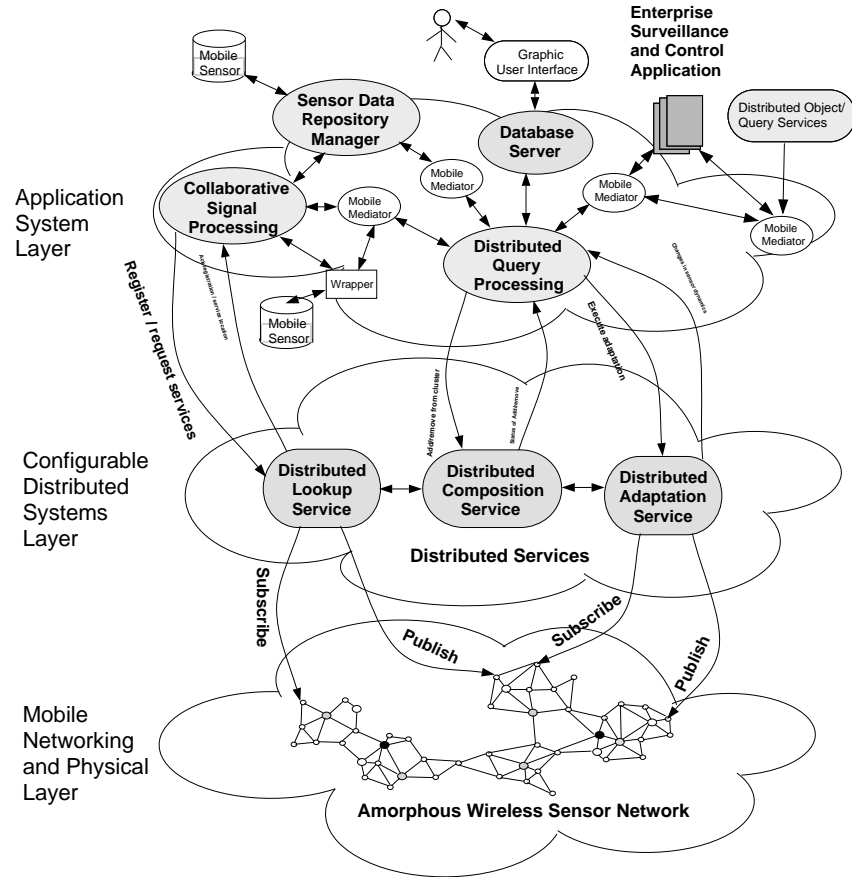
sensor nodes may be deployed incrementally and spontaneously with little or no pre-planning. The networks must be extensible to new types of sensor node and services. They must be deployed spontaneously to form efficient ad-hoc networks using sensors with limited computational, storage, and short-range wireless communication capabilities. They rapidly coordinate with each other to detect, track and report activities and disseminate the information efficiently through the impromptu network of sensors. Third, the sensor network must react rapidly to changes in the sensors composition, task or network requirements, device failure and degradation, and mobility of sensor nodes. Sensor devices may be deployed in very harsh environment, and subject to destruction and dynamically changing conditions. The configuration of the network will frequently change due to constant changes in sensor position, reachability, power availability and task requirements. The network protocols must be survivable in spite of device failure and frequent real-time changes. Sensor network must be secure in the face of this open and dynamic environment.

Critical sensor information must be disseminated through mobile transactions and dynamic query processing modules supported by the appropriate distributed services and network protocols to solve the problems of mobility, dispersion, weak and intermittent disconnection, dynamic reconfiguration, and limited power availability.

The underlying principles for building self-organizing sensor networks that can overcome the above challenges is to provide the fundamental mechanisms upon which other networking and system services may be spontaneously specified and reconfigured. The three fundamental mechanisms are service lookup, sensor node composition, and dynamic adaptation. Through a distributed implementation of these lookup servers, composition servers, and adaptation servers, other network and system services can be defined spontaneously in the sensor network. They also dynamically adapt these services to device failure and degradation, movement of sensor nodes, and changes in task and network requirements. Application-specific network and systems services may be provided impromptu by sensor nodes and supporting nodes, including data-centric routing, attribute-based addressing mechanisms, location services, naming and binding services, application-specific information dissemination and aggregation, caching and hoarding services, and security services.

Since these services are application specific, different protocols for a certain service may be specified for different applications and they may interoperate through the three fundamental mechanisms provided in the self-organizing sensor network architecture. For instance, some sensor networks may use directed diffusion method for routing sensor data from a source to a sink that initiated the interest in the information. Their specific routing services may be registered with the lookup server for its cluster. In another sensor network application, negotiation methods [13] may be more efficient for information dissemination. Sensor nodes will instead register these services with the lookup server. Each of these sensor networks may establish their own services spontaneously and independently. Two different sensor networks may interoperate through filtering and translation services that must be defined to route information between the two sensor networks.

## 2 Architecture



**Figure 1. Architecture of Self-Organizing Sensor Networks**

We use an approach that integrates three mobility-aware system layers: sensor information systems, configurable distributed services, and the network and physical device layer. The architecture avoids duplication of functionalities in the different layers and promotes efficient coordination between them. The sensor information layer contains mobility-aware mediators and adaptive sensor query processing. The runtime reconfigurable distributed system contains distributed services for supporting mobile sensor applications. The network and physical layer contains data-centric network routing protocols, physical wireless transmission modules and sensors that generate the raw data. The architecture consists of three key system layers (Figure 1):

1. *Application systems*, e.g. sensor information processing layer and collaborative signal processing
2. *Configurable distributed systems* that provide distributed services to the application systems
3. *Sensor networking and physical devices layer* that routes messages through the ad-hoc sensor network

At the physical device layer, different physical sensor and mobile devices may be assembled impromptu and reconfigured dynamically in an ad-hoc wireless network. Each sensor node contains battery power source, wireless communications, multiple sensing modality, computation unit and limited memory. Dual processors may be included for computation and real-time sensor processing. Three common sensing modality are supported – acoustic sensing using commercial microphones, seismic vibration using geophones, and motion detection using two-pixel infrared imagers. Wireless transceiver in the nodes provide communications between nodes, using Time Division Multiplexing and frequency hopping spread spectrum. Communication in each cluster of neighboring nodes is through a master node that establishes the frequencies used by the nodes. Each node contains a Global Positioning System (GPS) receiver that allows the node to determine its current location and time. GPS uses triangulation method with signals received from three satellites to calculate the location of the node with the accuracy of 1 meter. However, without clear line-of-sight to the satellites as in urban environments, GPS cannot be used. As we describe below, message routing and query processing uses these location information.

At the networking layer, ad-hoc routing protocols allow messages to be forwarded through multiple physical clusters of sensor nodes. Directed diffusion routing is used because of its ability to dynamically adapt to changes in sensor network topology and its energy efficient localized algorithms. To retrieve sensor information, a node will set up an interest gradient through all the intermediate nodes to the data source. Upon detecting an interest for its data, the source node will transmit its data at the requested rate.

The configurable distributed system uses the diffusion network protocol to route its messages in spite of dynamic changes in the sensor network. These distributed services will support applications systems, such as distributed query processing, collaborative signal processing, and other applications. The advantage of using these services is that application and system programs may use simpler communication interfaces and abstraction than the raw network communication interface and metaphor (e.g. subscribe/publish used in diffusion routing). Furthermore, these distributed services may enhance the overall performance, such as throughput and delay. These services will be implemented on top of the directed diffusion protocol which can still be used by applications concurrently with these distributed services. Directed diffusion can still be the preferred method for retrieving sensor data and will be used by some of the services. On the other hand, distributed services provide other forms of communication – such as interpersonal communi-

cation and impromptu establishment of community of services – required by other applications.

At the application system layer, distributed query processing and collaborative signal processing modules communicate with each other to support the surveillance and tracking functions of the enterprise. In sensor information systems, the cooperation between mobility-aware mediators, sensor agents and collaborative signal processing modules provide efficient access to diverse heterogeneous sensor data, surveillance and tracking information through the sensor network. The mobile sensor information layer is supported by three major components: interoperable mobile object, dynamic query processing and mobile transactions. We will not discuss mobile transactions in this paper. In the interoperable mobile object model, cooperative network of mobility-aware mediators and sensor agents will be configured to support interfaces to remote sensor data sources through multihop wireless network protocols.

### 3 Data-Centric Network Protocols

We use directed diffusion protocol[11] to implement all the distributed services and for retrieval of data through dynamically changing ad-hoc sensor networks. Diffusion routing converges quickly to network topological changes, conserves mobile sensor energy and reduces the network bandwidth overhead since routing information is not periodically advertised. Routing is based on the data contain in sensor nodes rather than unique identification. Directed diffusion is a type of reactive routing protocols which only update routing information on demand. In contrast, proactive routing protocols, such as links state routing, frequently exchange routing information. For sensor networks that experience greater dynamic changes, reactive routing algorithms are more appropriate, whereas for those that are more static and experience infrequent topological change, proactive routing algorithms are more efficient.

Directed diffusion is a data-centric protocol, i.e. nodes are not addressed by IP addresses but by the data they generate. Data generated by a node is named by the *attribute-value* pairs. A sink node requests for a certain data by broadcasting an *interest* for the named data in the sensor network. The interest and gradient is established at intermediate nodes for this request throughout the sensor network. When a source node has a data that matches the interest, the data will be “drawn” down towards that sink node using this interest gradient that was established. Intermediate nodes may cache, transform data, or direct interests based on previously cached data. The sink node can determine if a neighbor node is in the shortest path whenever it received a new data earliest from that node. The sink node will reinforce this shortest path by sending a reinforcement packet with a higher data rate to this neighbor node which forwards it to all the nodes in the shortest path. Other non-optimal paths may be negatively reinforced so that they do not forward data at all or at a lower rate.

Distributed services and applications use the publish and subscribe API provided by directed diffusion. Through the subscribe () function, an application declares an interest that consists of a list of attribute-value pairs. The subscription is then

diffused through the sensor network. A source node may indicate the type of data it offers through the `publish()` function. It then sends the actual data through the handle returned from the `publish()` function. The sink node then received the data that has propagated through the sensor network using a `recv()` function call with the handle returned from the `subscribe()` call.

## 4 Distributed Services

Self-organizing networks may be built from reconfigurable smart sensor nodes that may be developed independently but may interact with other smart sensor nodes. Some smart sensor nodes may execute autonomously to provide networking and system services or control various information retrieval and dissemination in the dynamically changing sensor network [15]. To enhance the ability to reconfigure their networking, configuration, and adaptation functionalities, smart sensor nodes may make use of three main classes of distributed services: Lookup service, composition service, and adaptation service (Figure 1).

The lookup service enables new system and network services to be registered and made available to other sensor nodes. Methods for calling the services remotely are also provided. The composition service allow clusters of sensor nodes to be formed and managed. The adaptation service allows sensors nodes and clusters to reconfigure dynamically as a result of sensor node mobility, failure and spontaneous deployment. These servers enable sensor nodes to form spontaneous communities in ad-hoc sensor networks that may be dynamically reconfigured and hierarchically composed to adapt to real-time information changes and events.

These distributed servers may be replicated for higher availability, efficiency and robustness. Distributed servers coordinate with each other to perform decentralized services, e.g. distributed lookup servers may work together to discover the location of a particular remote service requested by a node.

### 4.1 Reconfigurable Smart Nodes

By exploiting these distributed services, sensor nodes can be enabled to be self-aware, self-reconfigurable, and autonomous. These sensor nodes, known as reconfigurable smart nodes, can be used to build scalable and self-organizing sensor networks. (In this paper, we refer to reconfigurable smart sensor nodes as smart nodes or sensor nodes.) Smart nodes may represent sensor nodes, other types of mobile nodes, fixed nodes or cluster of these nodes. They may simultaneously be service providers for other smart nodes and clients of services that other smart nodes provide. Smart nodes may be dynamically composed into impromptu networked clusters forming clustered smart nodes that work together to provide abstract services for the agile sensor network. They may also adapt rapidly to abrupt changes in the sensors capabilities, events and new real-time information. Very large networks with hundreds of thousands of sensors nodes can built by hierarchically composing reconfigurable smart nodes.

Smart sensor nodes may consist of hardware devices and software for interacting with the real-world systems. The hardware may contain computational, memory, wireless communication, and sensing devices. Smart nodes may contain control

software for monitoring information from real-world devices such as simple sensors, engaging in distributed signal processing and generating appropriate control signals to produce a desired result in the real-world system. The control software takes advantage of the functionalities provided by the networking and system software.

Smart nodes interact with other smart nodes through well-defined interfaces (for networking and systems operations) which also maintain interaction states to allow nodes to be dynamically reconfigured. These explicit interaction states and behavior information allows localized algorithms with the adaptation servers to maintain consistency when autonomous nodes and clusters are reconfigured dynamically, moves around or recover from failure. The implementation and data of nodes (software and hardware) are encapsulated (hidden) from other nodes.

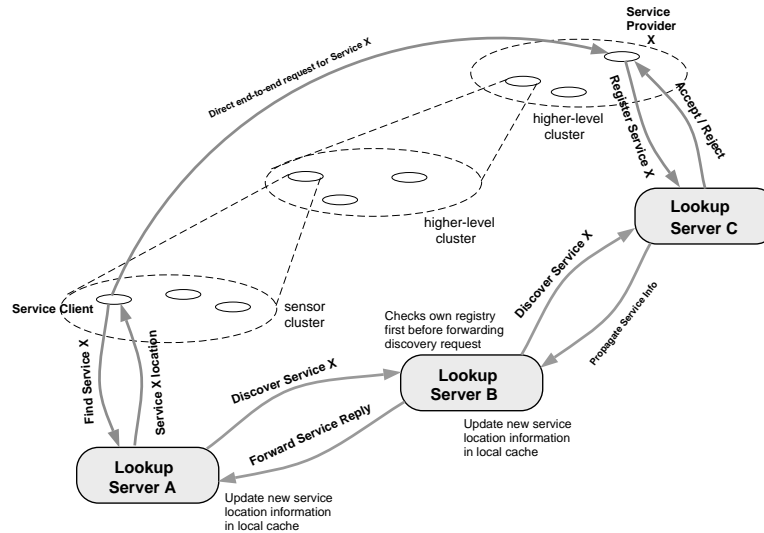
Different designers may independently develop smart nodes and their network and system services using different methods. For example, one designer may use a network protocol that is suited for a particular sensor application with its set of network requirements, such as low latency, power conservation, GPS capability, high error rate, and disconnection. In order to ensure consistency during dynamic reconfiguration and failure recovery of sensor nodes, the protocols may be analyzed by the adaptation servers based on their specification model.

When new smart nodes are added to the sensor network, they register their services with a lookup server (Figure 2). Other nodes that require a service will discover the services available in a cluster through the lookup servers that return the location and interface of the service nodes. This is similar to Jini [3] that manages system-level services based on Java code executing in IP-based networks. On the other hand, reconfigurable smart nodes may provide lower-level networking services using generic mobile codes executing in data-centric sensor networks. Client nodes then interact directly with the service node. Smart nodes are self-aware of their own location, configuration and services that they perform.

## 4.2 Lookup Server

New network and system services may be introduced by a sensor node to other nodes in a self-organizing sensor network. A sensor node that provides a service is called a service provider and a node that uses the service is called a service client. Since service providers may be introduced or removed from the sensor network at any time, a lookup server is needed to keep track of the availability of these services. A sensor node may register a resource that it maintains or service that it can perform with a lookup server (Figure 2). Each smart node has a home lookup server which keeps track of the location of the node when it moves. A lookup server may contain information on services or resources at multiple clusters. Other nodes that require the service may request the service through a lookup server. If the service is recorded in the lookup server, it will return the location of that service to the requesting node. Otherwise, if the service is not recorded in the lookup server of the region, a discovery protocol is used to locate the service through other lookup servers (Figure 2). A request message is propagated to all the lookup servers and the server that contains the service registration information will return the reply with the service location. It may also return the cluster name of that service. The

lookup server that made the request will then cache that service location and cluster name information in its local registration cache. At regular frequency, service and resource registration information may be disseminated from one lookup server to other lookup servers in the agile sensor network.

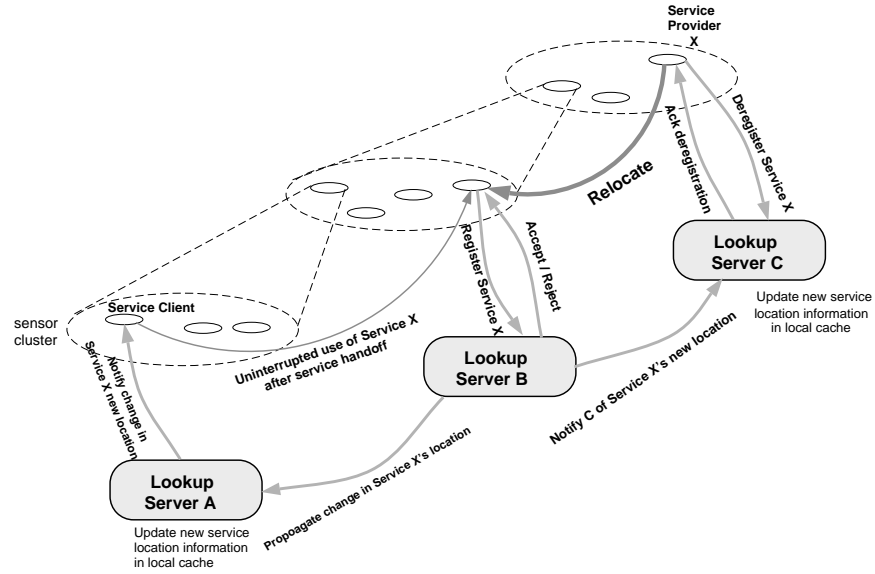


**Figure 2. Discovery of Services with Distributed Lookup Server**

Lookup servers support mobility of sensor nodes. When a sensor node moves to a different cluster at another location, it notifies, whenever possible, the previous lookup server that it is moving. When it arrives at another cluster in a new location, it will register with the new lookup server which will notify the previous lookup server (Figure 3). The new lookup server will propagate the change in the nodes' location to other lookup servers. The lookup server responsible for a sensor node that is interacting with the mobile node will notify the sensor node of the service location change. Existing interactions between the mobile node and other nodes will thus be handed over to the new location. Adaptation servers may be involved in the handover operation to preserve global consistency during the handoff, as discussed below, resulting in uninterrupted use of the service.

### 4.3 Compositional Server

The compositional server manages various smart nodes that may be added to (or removed from) clusters in the agile sensor network. It also manages network abstractions (or group behavior) of clusters and hierarchical composition of clusters. The compositional server simplifies dynamic reconfiguration of services provided by each smart node or cluster. It also simplifies the development of large self-organizing



**Figure 3. Mobility of Smart Nodes**

sensor network by allowing individual node and cluster to be specified and designed independently.

Compositional servers enhance compositionality and clustering abstraction of sensor networks. To enhance adaptivity in sensor networks, each node is designed independently and the networking requirements with other nodes may be specified separately. This decoupling of autonomous smart nodes from their networking requirements enables smart nodes to be easily adapted, replaced and reconfigured when triggered by dynamic events in the sensor network.

Clusters of smart sensor nodes may be formed under the management of a compositional server. Hierarchical clusters are also possible for larger scale sensor networks. A cluster of sensors may also provide distributed services by coordinating the tasks among the sensors, such as aggregating summary information. Clustered smart nodes encapsulates the networking and system capabilities provided cooperatively by the group of smart nodes. There will be a head smart node in the cluster that is responsible for the control of the cluster and inter-cluster communications and networking functions. Group communication to nodes in a cluster can be efficiently implemented by sending a message first to the cluster head which then multicasts it to the member nodes. Member nodes will elect a cluster head from the set of nodes with most powerful networking and system capabilities. Smart nodes in a cluster may cooperate to perform the networking and system functions for the cluster. Synchronization constraints associated with network protocols and system services among smart nodes may be specified in clustered smart nodes. The

capability to specify hierarchical composite clusters enables designers to build large and complex sensor networks by clustering together smaller network-enabled sensor devices at each level.

#### 4.4 Adaptation Server

Adaptation servers utilize information from the compositional server, lookup server, and analytical tools to control smart nodes during dynamic reconfiguration and failure recovery of the agile sensor network. Each smart node may execute autonomously to control different network operations in the sensor network and may interact and coordinate independently with other smart nodes to perform collaborative networking operations.

Adaptation servers monitor clusters of smart nodes during normal execution either through the spontaneous signal from the sensors, probing of the smart nodes or explicit network management directives for reconfiguration and failure recovery. When a runtime reconfiguration is requested or triggered, the adaptation server will generate the appropriate schedule of reconfiguration operations that will ensure the reconfigured and affected sensor nodes are globally consistent. To ensure correct adaptation and maintain consistency, the adaptation server makes use of analytical tools for dependency analysis and relevant information from compositional servers and lookup servers. When smart nodes are added or removed from the agile sensor network, a suite of analytical tools may be utilized to ensure that the sensor network still maintains its safety and liveness properties [17]. Smart nodes (or clusters of smart nodes) may be specified and analyzed independently.

#### 4.5 API for Lookup Service

Applications use the lookup service through the following API. We focus primarily on the lookup service API in this paper since it is responsible for enabling sensor networks to be self-organizing. In the next section, we will describe how these API functions are used by the various application systems of a surveillance sensor network.

These API functions use the following parameters.

- `service_type` is the generic type of service for which there may be several instances. For example, a type of service may be temperature monitoring, whereas specific instances of temperature sensors may be sensor  $X$  at location  $Y$ .
- `service_name` is the specific name that identifies an instance of a service provider.
- `input_list` is a list of attribute-value pairs containing the input parameters to the service invocation.
- `output_list` is a list of attribute-value pairs containing the output values from the service invocation.
- `lifetime` is the time period in which a service information will be stored in a lookup server.

- `interface_type` is one of the following three types of interface that the callee of the service use:
  1. Location or address. This is used by the service client if the interface for interacting with the service provider is known. The service client just need to retrieve the location or address of the service provider to be used for invoking the service request.
  2. Interface definition. This is used by the service client to retrieve the definition of the interface for interacting with the service provider. The service client must have the interpreter or compiler for the interface definition.
  3. Mobile code. This is used by the service client to retrieve the mobile code that implements the protocol for interacting with the service provider. The mobile code must then be dynamically linked to the service client.

The following is a description of the purposes and the side-effects of the lookup service function calls.

1. `service_call(service_name, input_list, output_list, interface_type)`  
 This function is used by a service client to find and make a call for a service where the service client does not know the location or address of the service provider and/or the interface for using the service. It is implemented as a combination of the `lookup_service()` and `service_exec()` calls described below. This function requires specific service name to be provided. Generic service type cannot be used since several service provider instances may match the service type.
2. `status = lookup_service(service_type, service_name, input_list, output_list, interface_type)`  
 This function allows a service client to find the location or address of a service provider and/or the interface for using the service. If `service_type` is defined and `service_name` is NULL, then all service providers registered with the lookup server of that type is returned. Service lookup can also be based on cluster or predicate matching. The cluster information and predicate for matching service providers are contained in the `input_list`. Depending on the `interface_type` used, i.e. location, interface definition or mobile code, the respective results of the `lookup_service()` call will be placed in the `output_list`.
3. `status = service_exec(service_name, input_list, output_list, interface_type)`  
 This function allows a node to request for the service and gets the results back from the service provider. The input parameters are supplied by the client in the `input_list`. The service provider performs the requested service or remote procedure call and returns the results in the `output_list`. The `interface_type` defines the method used by the service client to communicate with the service provider.

4. `status = service_register(service_type, service_name, lifetime, input_list)`  
 This function allows a service provider to register its service with a lookup server in the region. Services will remain in the lookup server for the lifetime specified by the service provider. In the `input_list`, the service provider may supply one or more of the following service information: (i) location or address, (ii) interface definition, or (iii) mobile code. Lookup servers for different regions may coordinate with each other to update their list of service information.
5. `status = service_deregister(service_type, service_name)`  
 This function allows a service provider to remove its service from the lookup server registry.

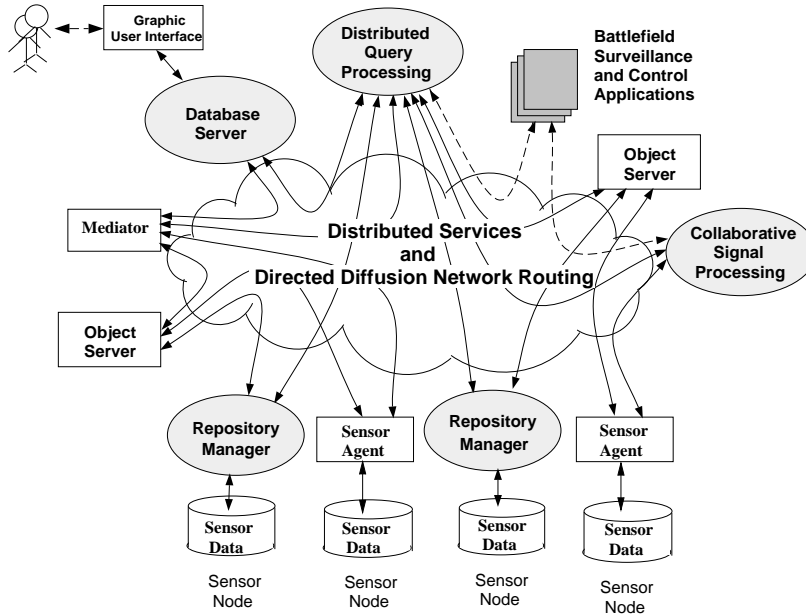
## 5 Application Systems

The self-organizing sensor network architecture allows sensor information system designers to specify their own specialized protocols and services that are most appropriate and efficient for the specific application, although most sensor nodes may use generic protocols and services. Since these services may be application specific, different protocols for a certain type of service may be specified for different applications. Each of these sensor applications may establish their own services spontaneously and independently of each other. Various types of network services may be independently defined for different sensor nodes. In the following subsections, we discuss some of these types of services.

### 5.1 Interoperable Mobile Object Model

The interoperable mobile object model is useful for retrieval of sensor information from sensor data sources to the surveillance application clients. The interoperable mobile object model extends the distributed interoperable object model (DIOM) [18] to sensor information networks. Application clients may access and update information sources in the sensor nodes through a group of mediators, object servers, sensor data repository managers, and sensor agents which communicate with each other using distributed services and diffusion routing (Figure 4).

An application end-user poses queries to the database server which coordinates with the mediators to decompose, schedule and route queries to the sensor information sources and collaborative signal processing modules. Mediators resolve the bindings of the sensor data sources through the object servers. The binding maps the object unique identifier to sensor agents or repository manager at the sensor nodes specified by their location, data types and application-specific information. Once the binding is resolved, mediators communicate with the sensor nodes directly. When sensor nodes move to new locations, routes may be changed through directed diffusion protocol if the change is incremental or through the distributed lookup service if the node moves quickly over a longer distance. Directed diffusion and the lookup services are responsible for solving the problem of mobility at the network routing and distributed service levels. However, applications services such as mediators and object servers must also be aware of the mobility of sensor data



**Figure 4. Distributed Services Support for Communication among Distributed Query Processing, Collaborative Signal Processing, Sensor Data Repository Managers, Mediators, Object Servers and Sensor Agents for Sensor Data Sources**

sources and cache location binding information. Distributed lookup servers notify mediators and object servers of relocation of sensor nodes.

### 5.1.1 Mediators

Mediators cooperate with each other to implement dynamic query processing and mobile transaction in sensor networks. When a database server first receives a query, it may decompose it into multiple sub-queries and forward them to selected mediators that may be associated with sensor object agents or collaborative signal processing agents for answering each sub-query. Sensor agents (described in Section 5.1.3) provide accesses to their local sensor data sources. Collaborative signal processing agents (described in Section 5.2) provides more accurate tracking and surveillance data by combining readings from multiple sensors. Each mediator may dynamically determine and update its association with the sensor agents and signal processing agents in the region. Mediator may discover the sensor agents and signal processing agents in its region by calling `lookup_service()` using a generic `service_type` parameter to retrieve all agents that provide the type of service required to process the sub-query.

The selection of the mediators for each sub-query is based on database server's

knowledge, stored in its cache, of the current location of the sensor data sources involved in the query. If the database server does not have location of the mediators, the database server may locate the selected mediators using the `lookup_service()` call. It then calls `service_exec()` to execute the sub-queries in each selected mediator. Results of the sub-queries are returned from these `service_exec()` calls. The database server may alternatively use `service_call()` that may automatically perform all the operations above by locating, executing the sub-queries and retrieving the results from each of the selected mediators. Mediators may be grouped into clusters using the composition service. Clusters may represent logical data partition and enable more efficient group communication to a cluster of mediators.

Dynamic query processing are performed within each mobile transaction. While the initial mediator is responsible for the overall transaction, the subsequent mediators are responsible for the sub-transactions and their related locking and logging functions. Mediators are aware of mobility of sensor nodes through notification from the distributed lookup service. Sensor nodes that have moved to a new location may register, using `register_service()` with another lookup server. The new lookup server will update the previous lookup server which sends a notification event to the mediator that is accessing the sensor node information. When sensor nodes have moved, mediators must reconfigure the routing and scheduling of its sub-queries to current locations of the sensor nodes. In addition to mobility, mediators may reconfigure their sub-queries when they detect disconnection and bandwidth variability.

### 5.1.2 Object Server

Object servers are primarily responsible for caching information on the location, availability and attributes of sensor data sources. When a mediator is interested in retrieving a particular sensor information, it can either inject an interest through the diffusion protocol or try to locate the sensor data source from an object server. The first option is preferred if the target sensors are within a certain range. The second option is preferred if the target sensors are located further away, in which case, the object server will return the location and application-specific attributes of the sensor data sources. Once the sensor data sources are located, the mediator communicates with them directly using directed diffusion.

Each mobile sensor node has a *home object server*. A large sensor network may contain many object servers, each responsible for a subset of sensor nodes. Each object server registers its service with a lookup server in its region using the `service_register()` call. This enable other sensor nodes to locate and register its object information with the object server. The home object server caches the current location of sensor nodes whenever they move. If the sensor node moves to a new location in short distance, directed diffusion protocol will handle mobility and change in the network topology. Location updates at the home object server is unnecessary for mobility over a short distance. However, for mobility over a longer distance, sensors that have moved will try to update its location with its home object server. It first searches for the object server using the `lookup_service()` call. Several lookup servers may be used to locate the object server. Then it updates

its location with the object server using the `service_exec()` call. Object servers keep track of location and application-specific information of objects whereas lookup servers keep track of the location and system-level information of services.

In continual queries, data are continually sent from the sensor nodes to the mediators through the `service_exec()` call. When sensor nodes move over a large distance, they will update its location with their home object servers through the distributed lookup services. They also update its information with the lookup server in the current region, which will propagate the information of all distributed lookup servers. Changes in the sensor location will trigger the handover procedure in the `service_exec()` call which will redirect the distributed service call to retrieve data from the sensor new location. The update of the location cache in the object server will improve the performance of future queries.

If a sensor node's movement affects the performance of an on-going continual query, the mediator may re-evaluate the query decomposition. Alternate query route and schedule may be used for updates and queries to sensor nodes that have moved. Early detection and notification of mobile sensor relocation by the lookup servers allows query routes and schedules to be updated to reflect current sensor node locations, thus improving the performance of distributed queries.

### 5.1.3 Sensor Agent

A sensor agent is a software module that serve one sensor data source. The sensor agent is built around the existing sensor data source to turn it into a local agent for the sensor object in order to make an existing sensor information source available to the network of mediators. Sensor agents enable different types of sensors to be deployed incrementally in the ad-hoc sensor network. The local agent is responsible for accessing sensor information source and obtaining the required data for answering the query. Sensor agents are customized to integrate with applications methods used in the sensor nodes, such as autonomous identification and location management. Services provided by sensor agents may also include translating a subquery into a sensor information retrieval command or language expression, submitting the translated query to the target sensor information source, and packaging the subquery result into a mediator object.

The sensor agent is also responsible for local management for sensor data stored in the node. It contains the local locking mechanisms for the global concurrency control scheme and the local recovery mechanism based on write-ahead logs. Agents may convert data in raw format to interoperable sensor object format.

Each sensor agent registers itself with a home object server and supply all its related application-specific object information, such as object identification and location. Other processes, such as mediators, may then locate the sensor objects through the object server. Sensor agent also register its service with the lookup server through the `service_register()` call in order to handle sensor mobility and reconfiguration more efficiently at the distributed services level.

When the sensor agent moves, it will inform the object server of its current location. However, it must first determine how to reach the object server through

the `lookup_service()` call and then updates the home object server through the `service_exec()` call. The object servers may in turn notify the mediators of the change in sensor movement or connection variations. The mediators may determine whether to modify the sub-query schedule.

## 5.2 Collaborative Signal Processing

Retrieval of sensor values from multiple sensors can increase the accuracy of the data in target recognition and tracking [6]. The multiple reading of the sensor values can be statistically combined to derive more accurate tracking data. Sensors in a region may be clustered together through the distributed composition server where their sensor reading may be combined using a weighted voting algorithm [23] to provide more accurate data. For each cluster, a sensor node may be elected to be the head of the cluster. Readings from multiple sensors in the clusters will be propagated to the cluster head through diffusion routing where the algorithm is applied. Results from each cluster head may be propagated to higher level cluster heads for data fusion.

In many sensor applications, data generated by the sensor node may be very large. The cost of propagation of these sensor data as in the previous scheme will be prohibitive. One solution to address this problem is to use mobile agents to migrate from node to node to perform data fusion using the local data [22]. Instead of transferring large amount of data throughout the sensor network, this approach only transfers the mobile agent code which is smaller than the sensor data. The result is an improvement in the execution time of the collaborative signal processing algorithm.

A sensor node that needs to execute a mobile agent code must download the mobile code from a mobile code repository manager in its region. It may use the `lookup_service()` call to first locate the repository manager that stores the relevant mobile code. It then calls `service_exec()` to the repository manager to download the code. An alternative method is to store the mobile code with the lookup server. The sensor node can then use `lookup_service()` to retrieve the mobile code directly from the lookup server.

The entire surveillance area is split to several sub-areas. Each sub-area is controlled by a signal processing agent, also known as processing element (PE), which may dispatch several mobile agents into that sub-area. The signal processing agent may register itself with the lookup server using `service_register()` to allow other nodes, such as the mediators, to access its tracking results. Each mobile agent will migrate from node to node to perform data fusion, for instance using multi-resolution data integration algorithm [22]. Each mobile agent, addressed by an identification, contains an itinerary, data, method and an interface. The identification is a 2-tuple composed of the identification of the dispatcher and the serial number assigned by the dispatcher. The itinerary describes the migration route assigned by the dispatcher. The data is the agent's private data that contains the integration results. The method describes the multi-resolution data integration algorithm. The interface is the function by which the agent communicates with the processing element and for the processing element to access the agent's private data.

When mobile agents migrate from node to node, the results of the sensor integration algorithm from previous nodes are cached in the mobile agents. These state information must be transferred with the mobile agent as it migrate from node to node. Agent migration and state transfer are supported by the distributed adaptation service. As the mobile agents visit each node, it may register with the lookup server. The primary signal processing agent (PE), may find these mobile agents as they move around and retrieve results using the `service_call()` function. A mobile agent may also retrieve intermediate results from other mobile agents through `service_call()`.

For migration of mobile agents far beyond a region managed by a lookup server, the agent will register with another lookup server in the new region. A service client may try to contact the mobile agent through the first lookup server may need to use several intermediate lookup servers to get the information about the mobile agent current location.

Multi-resolution signal processing algorithms may be implemented using hierarchical clustering of sensors provided by the distributed composition server. Results from a sensor cluster may be passed to agents responsible for signal processing for higher-level clusters. Agents for higher-level clusters may send messages to multiple sensor clusters using group communication method provided by the `service_exec()` call by specifying the appropriate `service_type` parameter.

## 6 Related Work

In recent years, the development of new integrated low-power sensor devices [14, 28, 25] has spurred more research on sensor networks for remote surveillance [6], pervasive computing, mobile computing, and computing continuum. Some research on sensors and RFID tags have also focused on the technical issues in wireless link between the sensor and the interrogator [16], low power consumption [25], sensor types and capability [19], position determination [27] and identification. Many of the current commercial systems transmit data from sensors to interrogator through wireless links, but forward them through direct static connections to a fixed network, either through a cable or wireless link. However, over-reliance on fixed network infrastructure is not feasible for large sensor information network. Instead, sensors must themselves be capable of assembling ad-hoc networks by themselves when they are deployed spontaneously in an area. The sensors must self-organize in spite of changes and failure in the sensors and the network topology.

Several novel concepts and protocols have been developed for many aspects of self-organizing sensor networks design and implementation. In ad-hoc network routing, localized algorithms have been developed for autonomous sensor devices to route information in an energy efficient way. Directed diffusion routing protocol [11], based on the localized computation model, provides energy-efficient and robust communication for dynamic network with small incremental changes. For dynamic networks with large-scale changes and high level of mobility, directed diffusion may not adapt very well. Similar diffusion routing concept have also been presented in other work [21]. Another localized protocol for information dissemination in sensor network use meta-data negotiation to eliminate redundant transmission [13].

Other self-organizing network routing protocols are dynamic source routing [12] and destination-sequenced distance vector [20], although it is not clear if these algorithms are energy efficient enough for sensor networks. In this paper, we allow sensors to form high-level clusters and use directed diffusion within clusters. Clusters may be formed using localized algorithm [9] for coordinating among sensors to elect extremal sensors.

Since sensor devices must be quickly and flexibly deployed in large number to coordinate through impromptu networks, each sensor device must operate autonomously in determining the capabilities of the sensor nodes in the vicinity and participate with the entire community of sensors to achieve global objectives. Distributed lookup services allow remote sensor nodes to be located more efficiently. The end-to-end and group communication services between nodes over a wide area are more efficient when localized routing algorithms are restricted within clusters. Discovery of services in mobile systems provides critical support for self-organizing sensor systems when sensors are being deployed and removed on the fly. In Jini [3], service discovery relies on mobile Java codes and is implemented based on TCP and UDP. It is not clear how these may be implemented using data-centric, ad-hoc sensor networks with services based on more generic mobile codes. Service Location Protocol (SLP) [10] is an IETF protocol for service discovery that is designed solely for IP-based networks.. Bluetooth [4] devices have a range of 10 meter and can directly communicate with at most seven other Bluetooth devices in a piconet. Bluetooth Service Discovery Protocol (SDP) [4] allows devices to browse and retrieve services by matching service classes or device attributes. Only services within the range of the device are returned. Our lookup service may retrieves services that could be multiple hops from the requesting node.

Communication in sensor network are data-centric since the identity of the numerous sensors are not as important as the data they contain [9, 8]. The networking infrastructure may provide more efficient dissemination of data through replication, caching and discovery protocols [8]. Communication protocols must be energy efficient since sensors [14, 9] have very limited energy supply. In our architecture, caching and aggregation of information may be provided by some sensors. The discovery of these sensors can be made through the distributed lookup servers that is implemented using diffusion routing. Changes in the sensor network are propagated to other caching and aggregation services through the adaptation servers. Our framework facilitates consistent adaptation of networking and system services as well as distributed sensor applications. Unlike other centralized control networks [7], our servers are associated only with sensors in a vicinity. Servers in different clusters will coordinate among themselves through information diffusion.

Recent advances on continual query and active database can be exploited for remote surveillance in sensor network. The DIOM system [18] is an object-based database designed primarily for integrated access to heterogeneous data sources. We extend this system to support sensor data sources and mobile nodes. DIOM continual queries may repeatedly retrieve and update sensor data for target tracking purposes. Cougar [5] is a distributed database designed specifically for network of sensors. Sensor devices are ADT objects in an object-relational database. Sensor

ADTs may contain asynchronous methods for retrieving readings from multiple sensors. Database operations, such as join, may be modified for these asynchronous methods.

Embedded networks research [24, 26, 2] have focus on supporting mobile devices through large infrastructure of networks in which each access point cover a small range [24]. This enables development of various applications for personal location, office information location, equipment tracking, item search in warehouses [26, 2, 27]. Due to lack of infrastructure in inhospitable environments where sensors are usually spontaneously deployed, sensors cannot rely on these embedded networks.

The concept of self-organizing wireless network have also been proposed in SWAN [29], although that research focus on circuit-switched connections for voice communication that use multiple channel transceivers with a slotted model. Some of the elements of reconfiguration are also discussed in amorphous computing systems [1] that capable of organizing themselves into zones.

## 7 Conclusions

We have described how distributed sensor applications, such as sensor information retrieval and remote surveillance, can be supported by distributed services in self-organizing sensor networks. The three basic distributed servers are lookup servers, composition servers, and adaptation servers. Through these servers, sensor nodes may be placed together impromptu in spontaneous environments and these sensor nodes will immediately know about the capabilities and functions of other sensor nodes and work together as a community system to perform cooperative tasks and networking functionalities. Newly deployed sensor nodes may provide new services. Other sensor nodes may locate and use these services spontaneously. These distributed services are implemented using directed diffusion network routing which provides energy-efficient and data-centric data communication. While diffusion routing can adapt dynamically to limited mobility and topological change, the distributed services supports large mobility and changes in sensor nodes. Diffusion uses data-centric communication model whereas in distributed sensor applications, it may be more convenient to use end-to-end process-oriented communication. Sensor nodes that discover services provided by other sensor node may call these services either through well-known interfaces, interpreted interface definition or mobile codes downloaded from the lookup server. The benefits of using these distributed services is that application and system programs may use simpler communication interfaces and abstraction than the raw network communication interface and metaphor of the sensor network layer (e.g. subscribe/publish used in diffusion routing). Furthermore, these distributed services may improve the overall performance, such as throughput and delay.

## References

- [1] H. Abelson, et. al., "Amorphous Computing," *Communications of the ACM*, Vol. 43, No. 5, May 2000.
- [2] N. Adams., "An Infrared Network for Mobile Computers," *USENIX Mobile and Location-Independent Computing Symposium*, MA, August 1993.

- [3] K. Arnold, et. al., *The Jini Specification*. Addison Wesley, 1999.
- [4] *Specification of the Bluetooth System*, available at <http://www.bluetooth.com/developer/specification/specification.asp>
- [5] P. Bonnet, J. Gehrke, and P. Seshadri, "Querying Processing in a Device Database System," Technical Report Tr99-1775, Computer Science, Cornell University, October 1999.
- [6] R. R. Brooks, "Chapter 26: Modern Sensor Networks," *CRC Handbook of Sensor Fusion*, accepted for publication.
- [7] Echelon, The LonWorks Company. LonWorks Solutions. <http://www.echelon.com/Solutions/>
- [8] M. Esler, et. al., "Next Century Challenges: Data-Centric Networking for Invisible Computing," *ACM Mobicom*, 1999.
- [9] D. Estrin, et. al., "Next Century Challenges: Scalable Coordination in Sensor Networks," *ACM Mobicom*, 1999.
- [10] E. Guttman, "Service Location Protocol: Automatic Discovery of IP Network Services," *IEEE Internet Computing*, Vol. 3, No. 4, July/Aug. 1999, pp. 71-80.
- [11] C. Intanagonwiwat, R. Govindan, D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *ACM Mobicom*, 2000.
- [12] D.B. Johnson and D.A. Maltz. "Dynamic Source Routing in Ad-Hoc Wireless Networks," in T. Imielinski and H. Korth, editors, *Mobile Computing*, Kluwer Academic Publishers, 1996.
- [13] J. Kulik, et. al., "Negotiation-based Protocols for Disseminating Information in Wireless Sensor Networks," *ACM Mobicom*, 1999.
- [14] J.M. Kahn, et. al., "Next Century Challenges: Mobile Networking for Smart Dust," *ACM Mobicom*, 1999.
- [15] A. Lim, "Architecture for Autonomous Decentralized Control of Large Adaptive Enterprises," *DARPA-JFACC Symposium on Advances in Enterprise Control*, San Diego, California, November 1999.
- [16] A. Lim and K. Mok, "Wireless Media Access Control for Highly Mobile Information Servers: Simulation and Performance Evaluation," *ACM Mobile Computing and Communication Review*, Vol 1, No. 2, 1997.
- [17] A. Lim, "Automatic Analytical Tools for Reliability and Dynamic Adaptation of Complex Distributed Systems," *IEEE ICECCS*, Nov 1995.
- [18] L. Liu and C. Pu, "The distributed interoperable object model and its application to large-scale interoperable database systems," *ACM CIKM*, 95.
- [19] H. T. Nagle, et. al., "The How and Why of Electronic Noses," *IEEE Spectrum*, Sep 98.

- [20] C. E. Perkins and P. Bhagwat, "Routing Over Multi-Hop Wireless Network of Mobile Computers," in *Mobile Computing*, edited by T. Imielinski and H. Korth, Kluwer Academic Publishers, 1996, pp. 183-206.
- [21] R. Poor, "Hyphos: A Self-Organizing, Wireless Network," Master's Thesis, MIT Media Lab., June 1997.
- [22] H. Qi, S. S. Iyengar and K. Chakrabarty, "Distributed Multi-Resolution Data Integration Using Mobile Agents," *Proc. IEEE Aerospace Conference*, 2001.
- [23] D. G. Saari, "Geometry of Voting: A Unifying Perspective," *Proc. Workshop on Foundations of Information/Decision Fusion with Applications to Engineering Problems*, DOE/ONR/NSF, Washington DC, August 1996.
- [24] F. Stajano, et. al., "The Thinnest Clients: Controlling it all via Cellphone," *ACM Mobile Computing and Communication Review*, Oct. 1998.
- [25] The Ultra Low Power Wireless Sensors project.  
[http://www-mtl.mit.edu/~jimng/project\\_top.html](http://www-mtl.mit.edu/~jimng/project_top.html)
- [26] R. Want, et. al., "The Active Badge Location System," *ACM Trans. on Information Systems*, Jan 1992.
- [27] J. Werb, et.al., "Designing a Positioning System for Finding Things and People Indoors," *IEEE Spectrum*, Sep 98.
- [28] G.J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM*, Vol. 43, No. 5, May 2000.
- [29] K. Scott and N. Bambos, "The Self-Organizing Wireless Adaptive Network (SWAN) Protocol for Communication Among Mobile Users," *Globecom*, 1995.