Part 2 : Scientific Information

Main applicant:	Murphy, Amy
Project title:	XXX

Contents

Wireless sensor networks are growing in popularity due to the availability of low-power sensing devices and to their flexibility for a large number of applications ranging from environmental monitoring to personal health management. Designing and building applications for wireless sensor networks is complex because of (1) the distributed nature of the environment, (2) the fact that the available sensors change over time, and (3) the nature of the applications themselves, which typically require strict quality of service (e.g., data accuracy) that may change over time as the state of the system being monitored changes. As wireless sensors are typically battery powered, energy management is a critical task to extend application lifetime. Many low level protocols address energy management but typically ignore the application needs, providing only a best effort quality level to the application. In distributed computing, *middleware* has been a successful technique for bridging the gap between what the application needs and what the network provides, but most existing middleware does not reach below the operating system interface and therefore cannot exploit the energy management techniques available at the network level.

This proposal introduces a new kind of middleware that not only understands the needs of sensor network applications, but also reaches down into the network, adjusting network characteristics to optimize application lifetime while maintaining the needed quality of service. This middleware system, MiLAN (Middleware Linking Applications and Networks), accepts from the application a specification of its needed quality of service, parameterized by the state of the application. To interface to the low-level network protocols, MiLAN exploits a plug-in architecture that identifies feasible sets of sensors that meet the network constraints (e.g., bandwidth), and the energy consumption for different configurations (e.g., master/slave relationships in Bluetooth, or routing in IEEE 802.11). By combining the application and network information, MiLAN ensures that the network is optimally configured to support the application and will dynamically reconfigure the network in response to changes in the environment (e.g., nodes currently available) as well as changes in the application's state (e.g., based on the current state of the system being monitored).

Early exploration into MiLAN suggests many opportunities for research. These include: developing an API that is general enough to support a wide range of sensor network applications yet simple enough to reduce complexity for the application designer; selecting feasible sets of sensors that meet application quality of service needs as well as network constraints; sharing the sensor network resources among multiple applications; developing an application design style suited to the MiLAN API; and distributing both application computation and MiLAN decision-making among the nodes of the network. The PIs possess the depth of knowledge in both sensor networks and middleware to achieve quality results in this research. Furthermore, the PIs' association with the University of Rochester's Center for Future Health provides them a unique testbed to evaluate MiLAN with real sensor network applications for health monitoring.

Broader Impacts: MiLAN will enable the rapid development of a variety of critical sensor network applications, such as medical monitoring for early disease detection, environmental monitoring to detect biological or chemical agents, and battlefield surveillance. These applications will directly impact people's lives, yet they can only be useful if they accomplish their tasks while maintaining a reasonable lifetime. These goals can be met by MiLAN, which will be made freely available to both industry and academia through Open Source licensing. Furthermore, graduate and undergraduate students in both the Computer Science and Electrical and Computer Engineering departments at the University of Rochester will participate in this research, being introduced to networking, middleware, and system integration issues, and increasing interdepartmental collaboration. Women and minority graduate and undergraduate students will be encouraged to participate.

Interest in wireless sensor networks has blossomed recently due to technological advances enabling small, energy-efficient sensing devices. Applications in this domain include home security, machine failure diagnosis, chemical detection, medical monitoring, and surveillance. To give a feel for potential applications, we briefly describe two scenarios.

First, consider a surveillance scenario where multiple sensors (e.g., acoustic, seismic, video) are distributed in an area and the application must report a detected phenomenon (e.g., an intruder). To do this, a minimum percentage of the area must be covered by sensed data. In many cases the sensors have overlapping coverage areas, providing different kinds of information about the same region. If the application does not require this redundant information, it would be desirable to conserve the energy of some sensors, for example, using small sets of sensors to meet the application demands (in this case, minimum coverage area). This requires that the application interact with lower levels of the sensor network's protocol stack to manage the sensors over time. Such management can be as simple as turning sensors on and off, or as complex as selecting the routes for data to take from the sensor to the collection point in a multi-hop network.

With small, portable sensors, the potential exists for monitoring an individual's health as they move around their home or in their daily tasks. Consider a personal health monitor application running on a PDA that receives and analyzes data from a number of body worn sensors (e.g., ECG, EMG, blood pressure, blood flow, pulse oxymeter). The monitor reacts to potential health risks and records health information in a local database. Considering that most sensors used by the personal health monitor will be battery-operated and use wireless communication, it is clear that this application can benefit from intelligent sensor management that provides energy-efficiency as well as a way to manage QoS requirements, which may change over time with changes in the patient's state. For example, higher quality might be required for certain health-related variables during high stress situations (states) such as a medical emergency, and lower quality during low stress situations such as as sleep.

The goal of this proposal is not to develop these applications, but rather to develop a framework that eases the development process of this style of application paying special attention to maximizing the lifetime of the network itself. Much of the existing research targeted at increasing the lifetime of sensor networks make simplifying assumptions, such as best effort service to the application. Furthermore, most of these approaches alter some aspect of the protocol stack, such as the MAC or routing layers, to achieve energy efficiency for a particular application. While advances have been made that increase application lifetime, these approaches are not flexible enough to accommodate the variety of complex sensor network applications we target. Rather than create application-specific network protocols, our approach extends application lifetime by dynamically adjusting the characteristics of a wide variety of protocols such as Bluetooth, IEEE 802.11, and ad hoc routing protocols. Characteristics that can be modified include master/slave roles in Bluetooth networks, and transmission power and routing in multi-hop networks.

Our approach encapsulates the complexity of parameter modification into a middleware that accepts the *policy* of how to manage and control the dynamic network from the application but hides *mechanisms* for implementing the policy inside the middleware. Our proposed middleware, MiLAN (Middleware Linking Applications and Networks), allows applications to specify their quality needs and then adjusts the network characteristics to increase application lifetime while still meeting those quality needs. Our approach is distinctive because (1) we explicitly include the ability of the application to change states over time and to specify a corresponding change in quality requirements, and (2) we exploit network characteristics to increase network lifetime. Our approach simplifies application development by allowing the designer to focus on application quality requirements, while the middleware determines the best approach to maximize lifetime exploiting the features of the underlying network protocols.

1 Related Work

Our work intersects two fields of research, namely low level sensor network research and existing middleware techniques. The resulting combination is middleware for sensor network. In this section we present related work on these two fields as well as several other emerging techniques for middleware being explored specifically for sensor networks. In later sections we will clearly distinguish our work from the others, showing how our unique application model is amenable to a variety of applications such as those outlined above and how we can achieve extended application lifetime in a unique way by manipulating standard network parameters rather that creating specialized protocols.

1.1 Sensor Networks

Certain existing protocols for sensor networks make use of low level node collaboration to reduce the energy cost of data transfer by aggregating data locally rather than sending all raw data to the application. For example, in the LEACH protocol [14], nodes form local clusters and all data within a cluster are aggregated by the cluster-head node before being transmitted to the base station. This limited form of low-level collaboration is also found in the query-based technique of directed diffusion [18], in which applications specify "interest queries" for the required data attributes, and the nodes collaborate to set up routes for this information to follow back to the application. MAC-level protocols, such as PAMAS [32] and S-MAC [33] reduce energy dissipation in the MAC protocol by turning nodes off as often as possible, often trading off latency in packet delivery for energy efficiency. As idle power can often be significant, these MAC-level techniques can greatly extend application lifetime. Other protocols such as [28] aim to determine the minimum transmit power necessary for a fully connected network, whereas protocols such as [21, 29] determine the optimal transmit power to minimize overall energy dissipation. Finally, topology control protocols such as ASCENT [4], Span [5], and STEM [30] turn sensors on and off to maximize network lifetime while keeping the network fully connected. Each of these protocols can increase system lifetime, but they do not take into account application desires and cannot easily be generalized or react to changes in application state.

As this summary shows, most sensor network research has focused on designing new network-level protocols (e.g., MAC layer, routing layer, topology control, etc.), without considering existing standards or how applications use the protocols. Our proposal is distinctive because we intend to work above existing protocols, leveraging the flexibility that exists in both standardized and non-standardized network protocols. For example, in a Bluetooth network [2], the master can be dynamically changed, allowing different nodes over time to assume this role and its associated energy burden. In multi-hop 802.11 networks [6], different routing protocols can affect the energy consumed at the nodes. Similarly, most of the sensor network specific protocols discussed above have parameters that can be modified to suit application goals while extending network lifetime. MiLAN will be designed to take advantage of the energy-saving techniques of *any* network protocol employed. Furthermore, existing approaches tend to address only a single application state, ignoring the dynamic changes of real sensor network applications. MiLAN allows the parameters of the network to be varied over time to best suit application needs as the application state changes.

1.2 Middleware

Traditionally, middleware sits between the application and the operating system, providing a high level application interface for coordination among applications, and hiding the underlying complexity of the environment. Corba [13], for example, hides the location of remote objects, simplifying the application's interactions with these remote objects by allowing all operations to appear local. Jini's [11] service discovery protocol and leasing mechanism allow client applications to discover services and manage client-server connections as the set of available services changes. The LIME middleware [24] provides a shared memory coordination scheme for mobile ad hoc components through a Linda-like tuple space [12].

Although most middleware systems do not modify the network characteristics, some adapt their own behavior based on detected network conditions. For example, both Limbo [8] and FarGo [16] reorder data exchanges or relocate components to respond to changing network conditions such as bandwidth availability or link reliability. Other middleware systems provide hooks to allow the applications to adapt. Applications built on the Odyssey platform [25] can register for notification of changes in the underlying network data rate. Context-aware systems, such as Rome [17] and comMotion [23], provide users with a context-aware message service that delivers messages when the recipient is in a certain location. CybreMinder [9] takes this adaptation one step farther, enriching the definition of context to include any facet of the environment (nearby people, weather forecast, etc.). In all of these systems, the current environment is detected, but the network itself is never altered.

1.3 Middleware for Sensor Networks

Recent efforts have focused on middleware specifically designed for sensor networks, exploring a variety of abstractions useful to the application programmers while simultaneously focusing on the restrictions of sensor environments.

EnviroTrack [1], a middleware for environmental tracking applications, supports event-driven programming by identifying an event at a given location, collecting the data from proximate sensors, and reporting the readings and event to the user. At a higher level, a sensor network middleware in development [34] allows applications to express the mapping between application quality and data quality. This middleware translates these data quality values into thresholds on sensor readings that specify when sensors should transition from a monitoring state into an active state (i.e., specifying the parameters that signify events of interest to the application, for which the application would like to receive data). Additionally, it is assumed that sensors can transition themselves to a low power consumption state when sensed data follows predefined prediction models to within certain error bounds.

Another approach to interacting with a sensor network is to access the data according to a traditional database model. TinyDB [22] provides an SQL-like interface with optimization for placement of parts of the query (e.g., joins, selects) to minimize power consumption. Cougar [3] and SINA [31] also provide a distributed database query interface towards a sensor network with an emphasis on power management either by distributing queries or clustering low-level information in the network.

Other data-oriented approaches, such as DSWare [20], address the redundancy of data collected by geographically proximate sensors. By aggregating the data of several sensors and reporting it as a single value, some amount of sensor failure can be tolerated. QUASAR [19] addresses quality concerns, allowing applications to express Quality aware Queries (QaQ). For example, QaQs can express quality requirements as either set-based (e.g., find at least 90% of the sensors with temperature greater than 50^oC) or value-based (e.g., estimate the average temperature within 1^oC).

2 Our Previous Work

The work proposed here is a continuation of the ideas presented in the attached paper [15], which presents the outline a new middleware for sensor networks called MiLAN, Middleware Linking Applications and Networks. The paper itself is a collaboration among the PI of this proposal (a computer scientist) and Wendi Heinzelman, an electrical engineering assistant professor at the University of Rochester during the time approximately three years ago when the two were co-located. The initial ideas have been well received by the community, and we hope to continue the work through the support of this proposal as well as a second proposal to be submitted in the US (see attached documentation).

The middleware that we propose is significantly different from other approaches because we focus on lengthening the lifetime of the network by *manipulating network parameters* while still meeting the expressed needs of the applications. We often refer to these needs as "Quality of Service", however we made the distinction up front that the QoS we address is application specific rather than standard notions such as processor time, memory, or bandwidth requirements. As

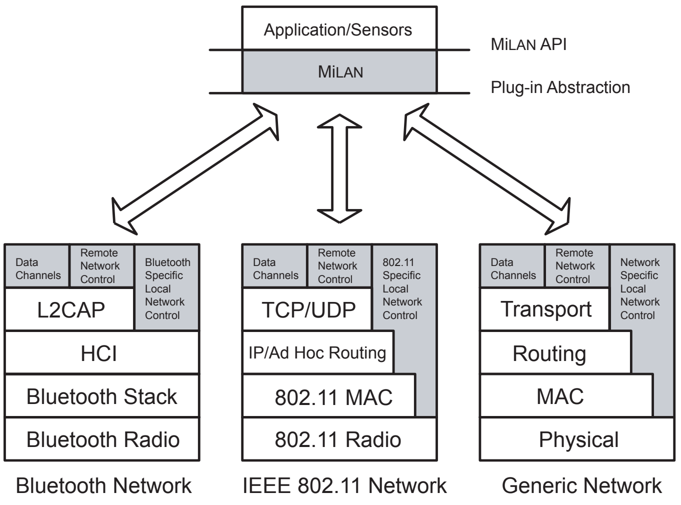


Figure 1: MiLAN components (shaded). MiLAN presents an API through which the application represents its desires with regard to different sensors that may be available. MiLAN also presents an abstraction from the network-level functionality through which it issues commands to determine available sensors and configure the network.

such, the existing body of work on QoS for middleware is not directly applicable in the domain we address.

Our work on MiLAN to date consists of a definition of a preliminary user interface, an architecture definition, and initial simulations to demonstrate the effectiveness of our ideas. In 2003 we collaborated with a cardiologist (H. Carvahlo, one of the authors of [15] to evaluate our interface on the health monitor application described previously. The remainder of this section describes our prior work while the next section describes the directions we intend to take during the period of this grant.

2.1 Overview

Traditional middleware is normally placed strictly between the application and the operating system, however because MiLAN intends to manipulate network parameters, it must reach down into the protocol stack (e.g., that of Bluetooth, 802.11, S-MAC, STEM, etc.). Therefore, we have designed an architecture as shown in Figure 1 that provides an abstraction layer allowing network specific plug-ins to convert MiLAN commands to protocol-specific commands that are passed through the usual network protocol stack. This allows MiLAN to continuously adapt the specific features of whichever network is being used for communication to best meet the applications' needs over time.

The notion of *best* for the application must be provided to MiLAN by the application itself. We propose to do this through the definition of specialized graphs that describe the quality needs and how various sensors can meet them. These graphs will be described shortly. Figure 2 shows the interaction among MiLAN, the applications, and the sensors. The figure makes a distinction between the network plug-ins and the core of MiLAN, emphasizing the separation of computation that is specific to the selected network type versus the computation that always occurs. Throughout this section, we refer to this description, providing details. To make the description of the MiLAN API and the network plug-in abstraction more concrete, we use the personal health monitor application described earlier as a running example. Note, however, that the MiLAN API is flexible enough to represent the needs of a large variety of sensor network applications.

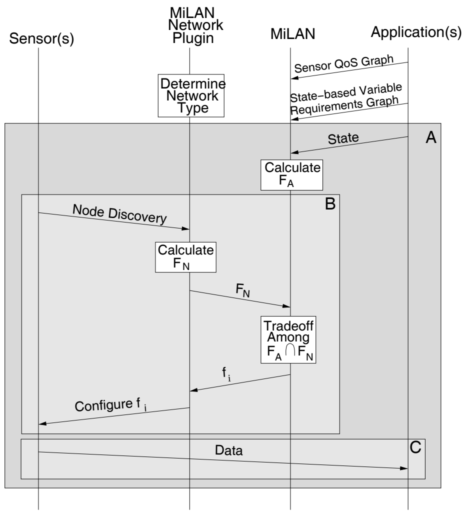


Figure 2: High level overview of MiLAN operation. Segment A repeats when the application changes its state based on data received from the sensors. Segment B repeats when sensors arrive in the network. Segment C repeats as data arrives from each sensor, and represents the normal operation of MiLAN conveying information from the sensors to the application. Note: arrows indicate dependencies among messages and computations. When no dependency exists, operations can be executed in parallel.

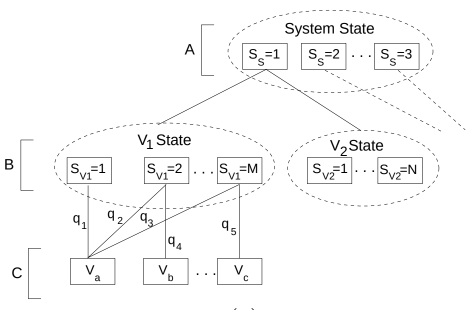
2.2 Specifying Application Performance

The fundamental design of most sensor network applications is to receive data from multiple sensors. The set of available sensors may change over time as energy reserves deplete or as sensors move in and out of range. When data is received, it is fed into various application variables that are used in the application's computation. This distinction between the raw data that can be provided by the network (the sensors) and the application's abstraction of those sensors (the variables) allows a variety of applications to be expressed in the same framework, and even for the same application to be supported by different underlying networks.

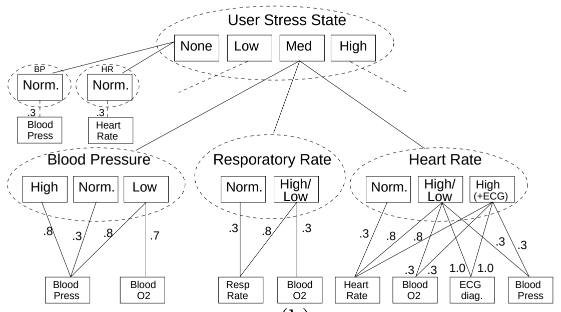
An additional consideration is the quality of the data required by the application. Fundamentally, the quality depends on which data is received and from which sensors that data is received. For example, in the personal health monitor, variables such as blood pressure, respiratory rate, and heart rate may be extracted from measurements obtained from any of several sensors [7]. Each sensor yields a certain quality for the variable it describes, e.g., a blood pressure sensor directly measures blood pressure, so it provides a quality of 1.0 in determining this variable. In addition, the blood pressure sensor can indirectly measure other variables such as heart rate, so it provides some quality, although less than 1.0, in determining these variables. The quality of the heart rate measurement would be improved through high-level fusion of the blood pressure measurements with data from additional sensors such as a blood flow sensor.

To determine how to best serve the application, MiLAN must know (1) the variables of interest to the application, (2) the required QoS for each variable, and (3) the level of QoS that data from each sensor or set of sensors can provide for each variable. Note that all of these may change based on the application's current state. As shown in Figure 2, during initialization of the application, this information is conveyed from the application to MiLAN via "State-based Variable Requirements" and "Sensor QoS" graphs. Examples of these graphs are shown in Figures 3 and 4.

Figure 3a, an abstract State-based Variable Requirements Graph, shows the required QoS for each variable of interest based on the current state of the application. Note that here we represent application state in a 2-level hierarchy, but this could easily be extended to more levels if required by the application. For a particular state (a combination of system state (level A) and variable

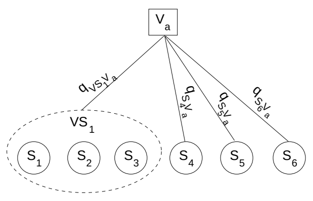


(a)

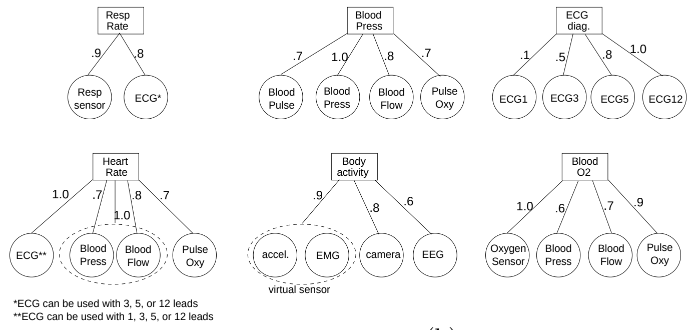


(b)

Figure 3: State-based Variable Requirements Graph for specifying the variables and the required QoS when the application is in various states. (a) Abstract example. (b) Example for the personal health monitor application. This graph illustrates only a subset of the application's possible states.



(a)



(b)

Figure 4: Sensor QoS Graph for specifying which sensors, or sets of sensors, can provide what level of QoS for each variable. (a) Abstract example. (b) Example for the personal health monitor application. This graph illustrates only a subset of the variables that should be considered by the application.

state (level B)), the State-based Variable Requirements Graph defines the required QoS for each relevant variable. Because variables (level C) can be named in multiple variable states (level B), MiLAN must extract the maximum QoS for each selected variable to satisfy the requirements for all variable states. Figure 3b shows the State-based Variable Requirements Graph for the personal health monitor. This application has two states, a system state that includes the overall level of stress that the patient is experiencing, as well as multiple states for each variable that can be monitored. The State-based Variable Requirements Graph specifies to MiLAN the application's minimum acceptable QoS for each variable (e.g., blood pressure, respiratory rate, heart rate, etc.) based on the current state of the patient. For example, the figure shows that when a patient is in a medium stress state and the blood pressure is low, the blood oxygen level must be monitored with a quality level of .7 and the blood pressure must be monitored with a quality level of .8.

For a given application, the QoS for each variable can be satisfied using data from one or more sensors. The application specifies this information to MiLAN through the Sensor QoS Graph; Figure 4a shows an abstraction of this graph. When multiple sensors are combined to provide a certain quality level to the variable, we refer to this as a single "virtual sensor." Figure 4b shows the Sensor QoS Graph for the personal health monitor. This graph illustrates some variables that are important to monitor when determining a patient's condition, as well as the sensors that can provide at least some quality to the measurement of these variables. Each line between a sensor (or virtual sensor) and a variable is labeled with the quality that the sensor (or virtual sensor) can provide in the measurement of that variable. For example, using data from a blood pressure sensor, the heart rate can be determined with a .7 quality level, but combining this with data from a blood flow sensor increases the quality level to 1.0.

Table 1: Feasible sets $\mathcal{F}_\mathcal{A}$ for the personal health monitor application for a patient in medium stress with high heart rate, normal respiratory rate, and low blood pressure.

Set #	Sensors
1	Blood flow, Resp. rate
2	Blood flow, ECG (3 leads)
3	Pulse oxymeter, Blood pressure, ECG (1 lead), Resp. rate
4	Pulse oxymeter, Blood pressure, ECG (3 leads)
5	Oxygen measurement, Blood pressure, ECG (1 lead), Resp. rate
6	Oxygen measurement, Blood pressure, ECG (3 leads)

With the information from these graphs as well as the current application state, MiLAN can determine which sets of sensors satisfy all the application's needs. These sets of sensors define the *application feasible set* $\mathcal{F}_\mathcal{A}$, a set of sets, where each element is a set of sensors that provide QoS greater than or equal to the application-specified minimum acceptable QoS for each variable. For example, in the personal health monitor, for a patient in medium stress state with high heart rate, normal respiratory rate, and low blood pressure, the satisfactory application feasible sets are shown in Table 1. MiLAN can choose any of these sets and the application will be satisfied. The choice of which set depends on network-level information.

2.3 Network Properties

As we have seen, the application needs specify several options for what must be provided by the sensor network, however, not all of these options may be possible in the current network configuration. As shown in Figure 2, it is the job of the network plug-in to determine which sets of sensors can be supported by the network, and thus to define the network feasible sets, $\mathcal{F}_\mathcal{N}$.

If we assume that all nodes are on a single-hop, centralized network, bandwidth constraints place limitations on the total amount of data that can be transmitted to the application. However, in more complex environments such as Bluetooth scatternets or 802.11 multi-hop networks, network topology plays an important role in determining network feasibility and power costs. For example, in Bluetooth it is necessary to choose a feasible scatternet topology, where nodes selected allow the network to be fully connected. In addition, we must also consider how the power costs of nodes are affected by their roles in the network (e.g., piconet masters or bridge nodes in Bluetooth scatternets). The power cost of using a node is a combination of the power to run the device, the power to transmit its data, the power to forward the data of other nodes in the set, and the overhead of maintaining its role in the network. These costs can be influenced by MiLAN through techniques such as transmission power control, efficient traffic scheduling, and the setting of different sleep states. In multi-hop networks, routing data from nodes to the application also becomes an important factor. The plug-in should know all of the network's protocol-specific features that can be modified and choose how to set these features to make sets feasible and energy-efficient. Although we have begun to consider these issues, we have not yet developed a model to incorporate them into the middleware.

The subsets of nodes that can be supported by the network define a *network feasible set* $\mathcal{F}_\mathcal{N}$[1]. We believe the protocol-specific information (e.g., paths the data can take to reach the application, or piconet membership) can be maintained as annotations to each member of the set. As only sets in $\mathcal{F}_\mathcal{A}$ provide the required application QoS, we can combine these two constraints to get an overall feasible set:

$$\mathcal{F} = \mathcal{F}_\mathcal{A} \cap \mathcal{F}_\mathcal{N} \tag{1}$$

[1]While it is possible to calculate all possible network feasible sets, this can be exponentially large with the number of nodes in the network. Because we are only interested in the sets that are also feasible for the application, $\mathcal{F}_\mathcal{A}$ can be passed to the plug-in, and a subset of $\mathcal{F}_\mathcal{A}$ chosen as network feasible.

For the personal health monitor, suppose that the sensors and processors communicate using an IEEE 802.11b network. As these networks can support overall throughput of nearly 11 Mbps, the network is able to support the transmission of all data from each of the sensor sets in $\mathcal{F}_{\mathcal{A}}$ from Table 1 in real-time. However, if other applications (e.g., video gait monitoring [10]) are running simultaneously on the network and the system QoS specifications declare that the personal health monitor application should only be allowed to utilize 100 kbps of the throughput, the network would not be able to support the transmission of data from the ECG sensor with either 3, 5, or 12 leads. Thus, the set of network feasible sets $\mathcal{F}_{\mathcal{N}}$ will only partially overlap with $\mathcal{F}_{\mathcal{A}}$. This overlap is the set of feasible sets $\mathcal{F}$ and consists of sets 1, 3, and 5 in Table 1. MiLAN must choose a set of sensors from one of the sets in $\mathcal{F}$ based on the tradeoffs discussed in the next section. If $\mathcal{F}$ is empty, MiLAN should raise an exception to the application, allowing it to decide the appropriate action.

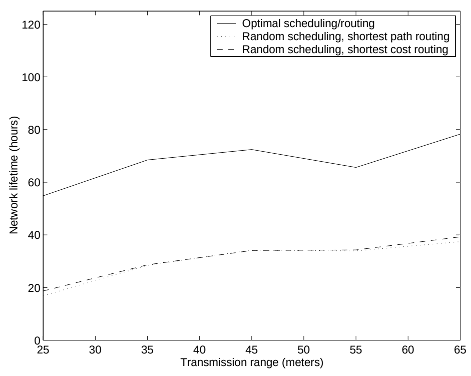
2.4 Tradeoffs

Among the elements in $\mathcal{F}$, MiLAN should choose an element f_i that represents the best performance/cost tradeoff. How should *best* be defined? This depends on the application and any method can be supported inside the MiLAN framework. In most sensor network applications, we want to allow the application to last as long as possible using the limited energy of each of the sensors. Simple approaches to choosing sensor sets may yield the set f_i that consumes the least power or that will run for the maximum lifetime before the first sensor dies. However, if we want to ensure that the application can run at the required QoS level as long as possible, we should instead optimize the total lifetime by intelligently choosing *how long* to use each feasible sensor set. In some cases, there are multiple ways to schedule sensors so that the same total network lifetime is achieved. In these cases, we may want to maximize the average quality of the sensor sets over time. For some applications, the goal may be to maximize some combination of lifetime and quality.

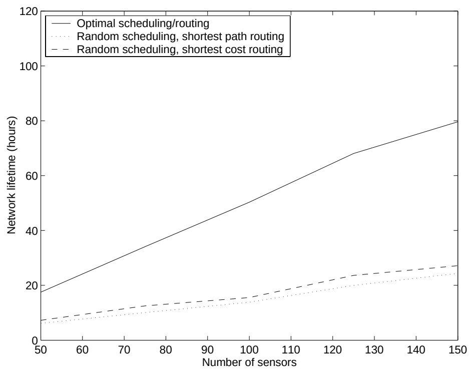
In Figure 2, we show this tradeoff computation occurring in the core MiLAN component. After the computation is complete and the first set of sensors is chosen, the MiLAN core informs the plug-in of the selection, and the plug-in configures the network accordingly, using information about the role each sensor should play (note: this information should be specified to MiLAN as annotations to $\mathcal{F}_{\mathcal{N}}$ by the network plug-in prior to the optimization).

To demonstrate the effectiveness of this idea, we have done preliminary work in maximizing system lifetime by choosing *how long* to use each element of $\mathcal{F}$. This problem can be modeled as a generalized maximum flow problem with some additional constraints, and it can be solved via linear programming. For a single-hop network, optimal scheduling of the sets in $\mathcal{F}$ can improve network lifetime by about 15% compared with simple scheduling schemes such as choosing the sensor set that consumes the least total power [26].

When sensor scheduling and data routing are jointly optimized, improvement in network lifetime is even more significant compared with simple sensor scheduling and low power routing protocols [27]. We ran experiments using the surveillance application described previously, where the application quality is specified by a certain percentage of the network area being covered by sensors. Figure 5a shows a plot of application lifetime using our optimal scheduling/routing technique and using randomly chosen sets with shortest path and shortest cost routing (where cost is a function of remaining node energy) as the transmission range of the radios is varied and the number of sensors and feasible sets are kept constant. Figure 5b shows a plot of application lifetime using our optimal scheduling/routing technique and using randomly chosen sets with shortest path and shortest cost routing as the number of sensors is varied and the transmission range and feasible sets are kept constant. As shown in these plots, the benefit of scheduling/routing optimization varies between a factor of two and a factor of four compared with shortest path and shortest cost routing.



(a)



(b)

Figure 5: Sensor network lifetime for optimized scheduling/routing, shortest path routing and shortest cost routing as (a) the transmission range is varied and (b) the number of sensors is varied.

3 Plan

The previous section lays the groundwork for the development of a middleware for sensor network. It describes a basic interface, an architecture, and directions for optimization, however the work supporting the outline has not been completed. In other words, the API has only been stressed on the personal health monitor, and although we believe it is widely applicable, this has not been verified. The components of the architecture including interaction with the application, extracting information from the network, and extracting sensor information have not been instantiated. Finally, although our initial experiments show success with a centralized optimization, we envision system with distributed decision making that makes decisions with less overhead but perhaps not optimal. In this section we explore some of these ideas.

3.1 MiLAN API

In the previous section, we discussed our initial API that allows applications to specify their desires to MiLAN using performance graphs, an approach based on our investigation into the features of different types of sensor network applications. This investigation showed that these applications require sensor data to measure variables, and that their requirements for which variables to measure and how important these measurements are change over time based on previous data. After attempting to encapsulate this information in several different formats, we found that the State-based Variable Requirements Graph, which represents the application needs for each particular state, meets our goals. Also, the relationship between individual sensors and the variables is effectively captured by the Sensor QoS Graph. Our experiences with the personal health monitor demonstrated that an expert in the application field can reasonably understand the concepts of these two graphs and generate them appropriately for the applications.

We will continue to research sensor network application requirements to see if our initial approach is, indeed, general enough to capture the QoS requirements of a wide variety of sensor network applications. Specifically, we will study other applications, such as the surveillance application described in the introduction, adapting the model as necessary. At the same time, we will explore other methods to specify and possibly generalize this information to find the best approach.

Another possible direction to guide our work is the connection between our graphs and the field of domain specific modeling. This connection has recently been brought to our attention, and we are just beginning to learn about the area. We expect the connections with software engineering researchers at the University of Lugano will ease this investigation.

3.2 Selecting Feasible Sets

There are many issues to be addressed with regard to the sensor feasible sets. First, we must determine how to find the sets of sensors that satisfy the application's QoS requirements (finding $\mathcal{F}_{\mathcal{A}}$) based on information from the performance graphs. We must also determine how to find the network feasible sets $\mathcal{F}_{\mathcal{N}}$ for different networks. Once we have found all feasible sensor sets, we must choose one of these based on some optimization criterion. Each of these tasks presents interesting research questions.

Finding the sets of components that can provide the application with acceptable QoS is not an easy problem. For example, consider the environmental monitoring application described previously. Suppose the application must guarantee that a certain percentage of the environment is being monitored at any given time. In this case, the number of sensor sets that match this constraint grows exponentially with the number of available sensors. Rather than finding all possible feasible sensor sets, MiLAN may find just a subset of the total. How should a "good," representative subset of $\mathcal{F}_{\mathcal{A}}$ be found? We will begin this investigation considering coverage area for homogeneous sensors, and later expand our research to include general sensor applications. In the example described at the end of the previous section, the selection was essentially random. However, we believe the process of selection can be improved even by straightforward methods. For example, we should choose sets that have minimal overlap, in order to increase the lifetime by using different sets of sensors. We will explore algorithms for making this selection, both for applications such as coverage area as well as more complex operations as found in the personal health monitor.

On the network side, MiLAN plug-ins must determine feasible sets of sensors that meet network constraints. The network constraints will be different for different network protocols. Thus we need to determine what features of the different protocols MiLAN can manipulate, and how. For example, with Bluetooth networks, the MiLAN plug-in must determine not only which nodes can send data while keeping the total data rate below network capacity but also how to assign nodes to piconets, which nodes to assign the role of piconet master, and which nodes to assign the role of scatternet gateway if there are multiple piconets. In a multi-hop network, the MiLAN plug-in must determine such features as how nodes should set their transmit power, which nodes should send sensor data, and which nodes should remain awake to act as routers for other sensors' data. We will perform research to determine which features MiLAN can affect for different standard and sensor network specific protocols at every layer of the protocol stack. We will also investigate mechanisms to optimally set these features, keeping in mind the impact these decisions will have on node energy and possible interactions between protocols at different levels of the network stack.

An additional challenge is to keep the interface between MiLAN and the network specific plug-in uniform. We believe that the information about the changeable network parameters can be abstracted for multiple networks and passed as annotations from the network feasible sets down to the network plug-in. Our work to define the annotations will involve careful understanding of multiple protocols, initially Bluetooth and 802.11, the properties that can be tuned, and ways of abstracting these properties into general principles to guide the selection of the network feasible sets.

3.3 Multi-State Applications

Our previous work has consisted of optimizing the network while the application remains in a single state, so the feasible sets $\mathcal{F}$ always remain the same. We have begun to look at how to model the optimization when the application state changes. This can be done by assuming some probability distribution that describes the amount of time (predicted) that the application will spend in a certain state. We are also looking into reducing the optimization computation (which is quite costly) through local optimizations rather than one global optimization.

3.4 Distribution

Distribution can be exploited in MiLAN in two primary ways. First, the application itself can be partitioned in multiple points in the network. This would allow some of the computation to be performed closer to the sensors collecting the data, eliminating some of the network communication of raw data. This requires changes to the MiLAN model to allow the application to be split into parts based on the data input and output, essentially creating a computational component that can be moved throughout the network. This style of component application structure is supported by standard software architecture development techniques, and is amenable to sensor network applications. This adds another dimension to the optimization, namely the processor utilization that also requires energy from the sensors.

This leads to the other dimension of distribution, namely distribution of the optimization inside the middleware itself. In our experimentation to date, we were able to achieve the maximum application lifetime because all information about the network was concentrated at the single location were the computation was performed. The idea is to relax the optimality constraint with the tradeoff that the decision will be made locally involving less network traffic, and therefore costing less. This will require two aspects not present in the original model of MiLAN. First, the application needs will need to be sent out to the sensors. Second, the sensors themselves must be aware of the other network components and what decisions they are likely to take. We envision a multi-phase approach that first spreads information about the application needs and sensors share information about themselves. Next the local decisions are made, and some information about the decisions are distributed. During the normal operation, some information can be shared among the sensors to continue an online optimization.

Our initial work on the distribution will fix a specific network protocol, likely 802.11, and develop the algorithms necessary to perform the optimization. The algorithms must be general enough to accept the information from the application graphs and to adapt to multiple types of sensor networks. We will measure our success by comparing the achieved network lifetime with the optimal sensor lifetime calculated by the optimization technique described earlier.

4 Time Table

The work described in this proposal will be divided between the PI in Lugano and a professor in the USA. The work will proceed in parallel with meetings among the members a couple of times per year, as travel allows. It is worth noting that the work proposed here can proceed even if the funding in the USA does not come through, although the support of this proposal can

- Year 1: Our research will begin with a deep evaluation of the API, applying it to other sensor network applications, and extending it when necessary, paying particular attention to its ability to scale and deal with a variety of sensors. We will research ways to take the performance graphs specified to MiLAN and determine a "good" set of application feasible sets. In the USA, study will begin on the investigation of the parameters that can be tuned in the network protocols and how they can be abstracted for use in the network feasible set selection.

- Year 2: Our research will focus on the distribution of the optimization, the techniques for sharing information among the nodes, and the algorithms for making the online optimization decisions. Meanwhile in the USA the centralized techniques will be further refined, and a centralized version of the middleware will be fully instantiated. This will include all of the details currently overlooked, including the sharing of the sensor states and the distribution of the optimization information. With the two fully instantiated versions, comparison among the version and feedback from one to the other is natural.

5 Impact

MiLAN will enable the rapid development of complex sensor network applications, which require extended application lifetime using changing sets of energy-constrained sensors, in a way that is not possible using today's middleware and network protocols. All that is required of the application in such a dynamic environment is to quantify its needed quality of service for possible application states, and MiLAN takes care of interfacing with the specific network protocol and making optimal decisions to maximize application lifetime. This will enable new sensor network applications to be easily designed and implemented, separating application concerns from network concerns.

The API developed will be applicable to many sensor networks, providing an additional model for other researchers in sensor network middleware to consider. This will be shared in the community through publication at workshop and conference papers.

MiLAN will also have an impact on large scale sensor networks. Such networks typically consist of a number of sensors randomly spread throughout a region, where the goal is to detect a pathogen or an intrusion. These applications are becoming increasingly important, and they are most useful when the system lifetime can be maximized without sacrificing the quality of service. Application developers often do not have the expertise to exploit all of the potentially energy-saving techniques available from the network layer. By building this functionality into MiLAN, the application developer can focus on the application itself, leaving the optimization to the middleware.

Given previous experience by the PIs with Open Source projects, we anticipate releasing the results of this work under such a license, making the system available to other researchers. Additionally, we will document our experiences with constructing and exploiting wireless sensors so that others can recreate test environments similar to ours. We hope that through these activities, we can guide others in the development of distributed sensor network applications, as well as build relationships with companies and other universities to exploit this work.

The novelty of this project lies in the combination of sensor network technology and middleware. These two fields have developed independently, and it is difficult for a single individual to have the depth of knowledge of both fields to complete this research. Therefore, this proposal combines the effort of two PIs, one with background and experience in the design and development of sensor network protocols, and the other with the same skills in middleware for mobile computing applications. Based on our initial interactions, we believe that collaboration will lead to a balanced system that makes significant breakthroughs in both sensor network applications and middleware.

References

[1] T. Abdelzaher, B. Blum, D. Evans, J. George, S. George, L. Gu, T. He, C. Huang, P. Nagaraddi, S. Son, P. Sorokin, J. Stankovic, and A. Wood. EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems*, 2004.

[2] Bluetooth Protocol. http://www.bluetooth.org.

[3] P. Bonnet, J. Gehrke, and P. Seshadri. Querying the Physical World. *IEEE Personal Communication*, 7(5):10–15, October 2000.

[4] A. Cerpa and D. Estrin. ASCENT: Adaptive Self-Configuring Sensor Network Topologies. In *Twenty-first International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, June 2002.

[5] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. *ACM Wireless Networks*, 8(5), September 2002.

[6] IEEE Computer Society LAN MAN Standards Committee. WirelessLAN Medium Access Control (MAC) and Physical Layer(PHY) Specifications. In *New York, New York, 1997. IEEE Std. 802.11-1997*, 1997.

[7] J.C.D. Conway, C.J.N. Coelho, D.C. da Silva, A.O. Fernandes, L.C.G. Andrade, and H.S. Carvalho. Wearable Computer as a Multi-parametric Monitor for Physiological Signals. In *Proceedings of the IEEE International Symposium on Bioinformatics and Bioengineering (BIBE)*, pages 236–242, 2000.

[8] N. Davies, S. Wade, A. Friday, and G. Blair. Limbo: A tuple space based platform for adaptive mobile applications. In *Proceedings of the International Conference on Open Distributed Processing/Distributed Platforms (ICODP/ICDP '97)*, Toronto, Canada, May 1997.

[9] A. Dey and G. Abowd. CybreMinder: A Context-Aware System for Supporting Reminders. In *Proceedings of the Second International Symposion on Handheld and Ubiquitous Computing*, pages 172–186, September 2000.

[10] S. L. Dockstader and A. M. Tekalp. Multiple Camera Tracking of Interacting and Occluded Human Motion. *Proceedings of the IEEE*, 89(10):1441–1455, Oct. 2001.

[11] K. Edwards. *Core JINI.* Prentice Hall, 1999.

[12] D. Gelernter. Generative Communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, January 1985.

[13] Object Management Group. *The Common Object Request Broker: Architecture and Specification Revision 2.2.* 492 Old Connecticut Path, Framingham, MA 01701, USA, 1998.

[14] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. *IEEE Transactions on Wireless Communication*, 1(4):660–670, Oct. 2002.

[15] W. Heinzelman, A.L. Murphy, H. Carvalho, and M. Perillo. Middleware to support sensor network applications. *IEEE Network Magazine Special Issue*, January 2004.

[16] O. Holder, I. Ben-Shaul, and H. Gazit. System Support for Dynamic Layout of Distributed Applications. In *Proceedings of the 19th International Conference on Distributed Conputing*, pages 403–411, 1999.

[17] A. Huang, B. Ling, S. Ponnekanti, and A. Fox. Pervasive Computing: What is it good for? In *Proceedings of the ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 84–91, August 1999.

[18] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. *Proceedings of ACM Mobicom '00)*, Aug. 2000.

[19] Iosif Lazaridis, Qi Han, Xingbo Yu, Sharad Mehrotra, Nalini Venkatasubramanian, Dmitri V. Kalashnikov, and Weiwen Yang. QUASAR: Quality-Aware Sensing Architecture. *SIGMOD Record*, 33(1):26–31, March 2004.

[20] S. Li, S. Son, and J. Stankovic. Event Detection Services Using Data Service Middleware in Distributed Sensor Networks. In *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks*, April 2003.

[21] E. Lloyd, R. Liu, M. Marathe, R. Ramanathan, and S. Ravi. Algorithmic Aspects of Topology Control Problems for Ad Hoc Networks. In *Processings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002)*, pages 123–134, June 2002.

[22] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. The Design of an Acquisitional Query Processor for Sensor Networks. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, 2003.

[23] N. Marmasse and C. Schmandt. Location-Aware Information Delivery with comMotion. In *Proceedings of the Second International Symposium on Handheld and Ubiquitous Computing*, pages 157–171, September 2000.

[24] A.L. Murphy, G.P. Picco, and G.-C. Roman. LIME: A Middleware for Physical and Logical Mobility. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, pages 524–533, April 2001.

[25] B. D. Noble and M. Satyanarayanan. Experience with Adaptive Mobile Applications in Odyssey. *Mobile Networks and Applications*, 4(4):245–254, 1999.

[26] M. Perillo and W. Heinzelman. Optimal Sensor Management Under Energy and Reliability Constraints. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '03)*, March 2003.

[27] M. Perillo and W. Heinzelman. Providing Application QoS through Intelligent Sensor Management. In *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA '03)*, May 2003.

[28] R. Ramanathan and R. Rosales-Hain. Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, pages 404–413, March 2000.

[29] V. Rodoplu and T. Meng. Minimum Energy Mobile Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1333–1344, August 1999.

[30] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Optimizing Sensor Networks in the Energy-Latency-Density Design Space. *IEEE Transactions on Mobile Computing*, 1(1):70–80, January 2002.

[31] C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo. Sensor Information Networking Architecture and Applications. *IEEE Personal Communication*, 8(4):52–59, August 2001.

[32] S. Singh and C. Raghavendra. PAMAS: Power Aware Multi-Access Protocol with Signalling for Ad Hoc Networks. *ACM Computer Communication Review*, 28(3):5–26, July 1998.

[33] Y. Wei, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, June 2002.

[34] X. Yu, K. Niyogi, S. Mehrotra, and N. Venkatasubramanian. Adaptive Middleware for Distributed Sensor Environments. In *Distributed Systems Online*, May 2003.