Issues in Designing Middleware for Wireless Sensor Networks

Yang Yu, Bhaskar Krishnamachari, and Viktor K. Prasanna
Department of EE-Systems
University of Southern California
Los Angeles, CA 90089-2562
{yangyu, bkrishna, prasanna}@usc.edu
http://ceng.usc.edu/~prasanna, http://ceng.usc.edu/~anrg

***Abstract*— Wireless sensor networks are being developed for a variety of applications. With the continuing advances in network and application design, appropriate middleware is needed to provide both standardized and portable system abstractions and the capability to support and coordinate concurrent applications on sensor networks. In this paper, we first identify several design principles for such a middleware. These principles motivate a cluster-based lightweight middleware framework that separates application semantics from the underlying hardware, operating system, and network infrastructure. We propose a layered architecture for each cluster that consists of a cluster control layer and a resource management layer. Key design issues and related challenges within this framework that deserve further investigation are outlined. Finally, we discuss a technique for energy-efficient resource allocation in a single-hop cluster, which serves as the basic primitive for the development of the resource management layer.**

I. Introduction

Wireless sensor networks (WSNs) are a significant technology attracting considerable research attention in recent years. They are being developed for a wide range of civil and military applications, such as object tracking, infrastructure monitoring, habitat sensing, and battlefield surveillance. Typically, a WSN consists of hundreds to thousands of tiny sensor nodes that communicate over wireless channels and perform distributed sensing and collaborative data processing.

State-of-the-art techniques for WSNs focus on simple data-gathering style applications, and in most cases, support one application per network. Therefore, the design of the network protocols and applications are usually closely-coupled, or even combined as a monolithic procedure. However, such procedures are sometimes ad hoc and impose direct interaction with the underlying embedded operating system, or even the hardware components, of sensor nodes. We envision that the development of WSNs will finally demand systematic application design methods based on standard and portable abstractions of the system. In addition, multiple applications will be required to be concurrently executed over a single WSN. For instance, a building monitoring system may need to simultaneously monitor the temperature and luminance, check cracks on the wall, track traversing persons, and even communicate with systems in nearby buildings. Thus, middleware sitting between the network hardware, operating systems, and network stacks and the application is required to provide (1) standardized system services to diverse applications, (2) a runtime environment that can support and coordinate multiple applications, and (3) mechanisms to achieve adaptive and efficient utilization of system resources. Such a middleware is particularly useful for WSNs that host complex applications with large amount of information processing and/or stringent performance constraints.

This work is supported by NSF grant number IIS-0330445 and by NSF ITR medium grant, award number 0325875.

While there have been numerous efforts at developing routing and communication protocols for WSNs [1], the fundamental problem of identifying and developing an appropriate middleware for fully realizing the capability of sensor technologies and applications remains to be addressed. Traditional distributed middlewares (such as DCOM, CORBA, PVM, MPI) are normally heavyweight in terms of memory and computation requirements and therefore not suitable for WSNs with scarce energy and processing resources. Instead, simple, easily implementable, *lightweight* designs are desired. Moreover, the middleware design needs to address the unique operating modes of WSNs that are significantly different from traditional networks, including the ad hoc deployment, untethered operation, and dynamic operating environments.

In this article, we first identify several design principles for WSN middleware. These principles motivate a cluster-based middleware framework that provides a *virtual machine* abstraction to separate application semantics from the underlying infrastructure. Each cluster contains a set of spatially adjacent sensor nodes that cooperate as a basic functional unit of the middleware. We then propose a simple and lightweight layered infrastructure for each cluster that consists of a cluster control layer and a resource management layer. Design issues and related challenges within this framework that deserve further investigation are outlined. Finally, we discuss a technique for energy-efficient resource allocation in a single-hop cluster, which serves as the basic primitive for the development of the resource management layer.

A. Application Challenges

Although spreading over a broad spectrum, most applications on WSNs can be characterized by several common features. Typically, sensor nodes are powered by batteries and

dispersed over an operational area where the phenomena of interest may appear. To conserve energy, sensor nodes can operate in several working modes with different functionalities or processing speeds, and thus different power consumption. Mechanisms for changing the working mode of a sensor node are called system "knobs". Examples of state-of-the-art system knobs include dynamic power management [2], dynamic voltage scaling [3], and modulation scaling [4]. Initially, to save energy, all sensor nodes may be switched into sleeping mode, except for a few "guarding" ones. Once the guarding nodes detect any signals that indicate the appearance of the target phenomena, they would be responsible for activating (waking up) other sleeping nodes. Ideally, enough number of nodes should be woken up around the phenomena to temporarily form a distributed subsystem that is capable of accomplishing the desired mission.

Several challenges must be overcome for completing the above procedure. First, efficient mechanisms are needed to selectively activate sleeping nodes with appropriate positions around the target and most remaining energy. Second, the activated nodes need to cooperate together for distributing the required signal processing tasks, aggregating the results, and routing the final decision to the base station. Third, in many cases, certain QoS requirements must be satisfied. Specifically, the requirements on the processing latency and fidelity of the decision determine the amount of needed computation/communication/sensing (CCS) resources and energy. Fourth, the network management can be complicated due to sensor heterogeneity (i.e., when different nodes have varying energy levels, processing capabilities and sensing modalities).

The above challenges are not trivial, considering the ad hoc characteristics of the network, the limited energy and CCS capabilities of sensors, the possible variations in the system and environmental conditions, and the fact that sensor nodes need to operate in an unattended manner. To relieve the burden on application designers, middleware is needed to provide a somewhat general runtime environment that stems from the above common features while taking the above challenges into account. To guarantee the proper functioning of concurrent applications under stringent resource constraints and high dynamics, the middleware is required to effect efficient tradeoffs between the multiple QoS dimensions of an individual application as well as between multiple applications.

B. Related Efforts

A general description of the middleware challenges for WSNs is presented in [5]. NEST [6] develops a real-time network coordination and control middleware. The interface between the OS and the application programmer is abstracted as so-called Microcells that can perform self-replication, migration, or grouping operations. The Smart Messages Project [7] proposes a distributed computation model based on execution migration. Smart Messages are migratory execution units that contain both data and code. In [8], location-centric computation is proposed, where regions are used to accommodate collaboration between sensor nodes around the target phenomena. In some sense, regions bear resemblance to the cluster concept in this paper. In [9], an adaptive middleware framework is proposed to explore the resource/quality tradeoffs during information collection. The main idea is to reduce the communication frequency at sensor nodes by lowering the sampling frequency without compromising the accuracy of the results. In [10], MiLAN is developed to enable dynamic network configuration (i.e., to identify and organize network resources) to fulfill the performance requirements from the applications. The design, algorithms, and implementations of middleware components to support queries over sensor database is investigated in [11]. Clustering techniques are used in [11] for intelligent in-network aggregation to reduce the amount of communication between sensor nodes.

The TinyOS [12] from Berkeley has recently become a *de facto* choice for operating system on individual sensor node, encouraging middleware components to be developed on top of TinyOS. For example, Maté, a tiny communication-centric component running on TinyOS is presented in [13] to facilitate frequent reprogramming of sensor networks in an energy-efficient style. However, while the goal of Maté is to provide a high-level program interface for single sensor node, our work focuses on providing a system-wide virtual machine abstraction to facilitate localized cluster control and resource management mechanisms, such that the performance of multiple applications can be satisfied under the stringent resource constraints and high dynamics of WSNs.

Lightweight variations of the traditional CORBA service are also being developed for accommodate wireless applications, such as in [14], [15]. However, CORBA is inherently based on "request/response" synchronous communication model, which is a misfit for the nature of WSNs, where the communication is packet based, highly variable in speed, and error-prone [16].

II. Design Principles

As discussed in [17], the software design for WSNs should follow several basic principles. We re-interpret those principles for the design of middlewares as follows:

(1) The middleware should provide *Data-centric* mechanisms for data processing and querying within the network. Due to its simplicity, flexibility, and robustness, cluster-based network architecture has been widely used in the design and implementation of network protocols and collaborative signal processing applications for WSNs, such as [11], [18]–[20]. Intuitively, cluster-based architecture is suitable for hosting the data-centric processing paradigm from both geographical and system design perspectives.

(2) *Application knowledge* can be used to tailor the design and implementation of softwares. It is thus important to integrate application knowledge into the services provided by the middleware. However, due to the mission to support and optimize for a broad class of applications, tradeoffs need to be explored between the degree of application-specific and the generality of the middleware. A practical policy is to embed the unique features of an application into the application code or specification, which can be interpreted by the middleware. Such embedded information can then be used to direct the operations of the middleware.

(3) *Localized algorithms* should be used to collectively achieve a desired global objective while providing good system scalability and robustness. Since the cluster-based architecture localizes the interaction of sensor nodes and hence the coordination and control overhead within a restricted vicinity, it is reasonable to regard each cluster as a basic function unit of the middleware. Consequently, the middleware performs as a distributed software composed of multiple clusters.

We believe that it is necessary to respect two additional principles as follows:

(4) Since the available resources of sensor nodes are low, the middleware itself should be *lightweight* in terms of the computation and communication requirements. The lightweight requirement necessitates simple and efficient heuristics to be used for suboptimal solutions.

(5) Due to the limited resources, it is very likely that the performance requirements of all the running applications cannot be simultaneously satisfied. Therefore, it's necessary for the middleware to smartly trade the QoS of various applications against each other. Note that this is different from the concept of adaptive fidelity algorithm [17] that trades the QoS of a specific application against its resource usage. Accordingly, we need mechanisms to specify and adapt the policy of tradeoffs conducted by the middleware.

Both principles (1) and (3) strongly motivate cluster-based architectures. In fact, such architectures have been widely investigated in ad hoc networks since they "promote more efficient use of resources in controlling large dynamic networks" [21]. Compared with mobile networks that incur a high cost for maintaining clusters throughout the network, WSNs usually consist of stationary sensor nodes with less dynamics. Hence, the cost for superimposing cluster architecture over the physical network is affordable, given the potential advantages offered by clusters in designing scalable and localized data-centric algorithms.

III. Overview of a Cluster-Based Middleware Architecture

In general, a cluster is a set of spatially adjacent sensor nodes that reside around the target phenomena and are capable of detecting and/or processing the data of interest. Clusters are dynamically formed during the lifetime of the system, triggered by the changing conditions of the environment, data source, and sensor nodes. Accordingly, the members of a cluster are dynamically adjusted. Multiple clusters may coexist within the system and overlap with each other. During the formation of a cluster, one node is elected as the cluster head, which is responsible for the control and coordination of sensor nodes within the cluster.

As shown in Figure 1, the middleware infrastructure is divided into two layers. We call the abstraction provided by the middleware a *virtual machine*, because of its similarity to the virtual machine concept in traditional distributed systems in terms of providing application semantic transparency from the physical infrastructure. While the cluster forming and control protocol is distributed among all sensor nodes, it is assumed that the code for resource management layer resides at the cluster head. A multi-cluster hierarchical architecture can be analogously developed, and is beyond the scope of this paper. Also, the eventual form and modularization of the middleware can be reasonably varied from the structure in Figure 1.

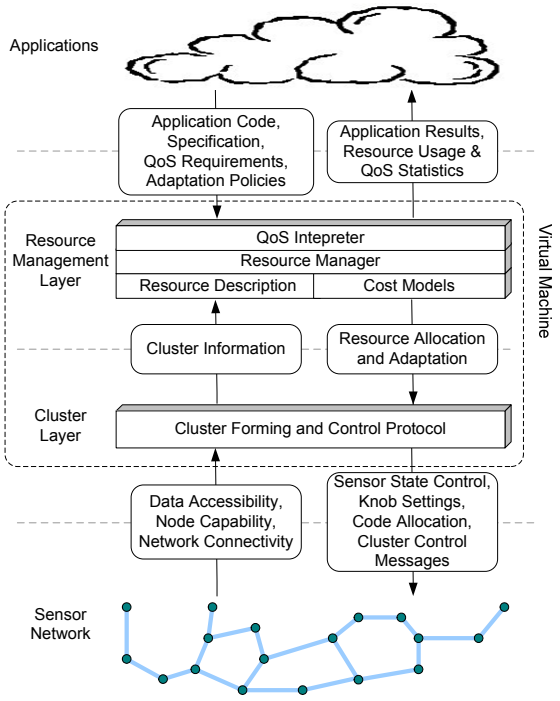


Fig. 1. Cluster-Based Middleware Architecture

Cluster Layer: The cluster layer is responsible for forming a cluster from a pool of sensor nodes that are around the target phenomena. Typically, the data accessibility, node capability (including remaining energy and CCS capabilities), and network connectivity are the criteria for determining the membership of a sensor node. Obviously, the gathering and exchanging of such information should be performed in a distributed way. Application-related knowledge is embedded in the application specification and passed down to the cluster layer after interpreted by the resource management layer. In addition, the cluster layer distributes the commands issued from the cluster head for resource management and cluster control purpose.

Most of the previous efforts on clustering WSNs, such as LEACH [18], tend to statically partition the network into multiple clusters a priori. However, our middleware is designed to dynamically form and manage clusters around the phenomena of interest. The challenges involve cluster adaptation and inter-cluster coordination, which will be discussed later.

Resource Management Layer: As the key component of the middleware, the resource management layer commands the allocation and adaptation of resources, such that the QoS requirements specified by the applications can be met. Resource allocation focuses on generating an initial solution when the cluster is formed, while resource adaptation controls the runtime behavior of the cluster. Both of these steps need to solve the problem of determining the scheduling of applications onto corresponding resources and the adjustment of system knobs.

In general, such problems turn out to be computationally hard. Thus, heuristics providing near-optimal solutions are needed.

It is crucial to ensure the robustness of the system. System and environmental variations may cause deterioration of the node capabilities and the quality of communication channels, or even node failure. Therefore this information needs to be periodically gathered through the cluster control protocol and updated at the cluster head. The cluster head is then responsible for taking adaptation actions at the sight of possible failure of the functionality or QoS requirements of applications. Directions from applications will be necessary in selecting the right adaptation actions, especially when the resource availability is tight. From one hand, *adaptive fidelity* algorithms are extremely useful in trading the quality of a single application for its resource usage. On the other hand, inter-application coordinations are necessary to perform system-wide tradeoffs and achieve optimized and balanced resource utilization among multiple applications. For instance, different priorities associated with applications will affect the fidelity adaptation of individual application.

We note that the routing mechanisms are not shown in the architecture as we assume that the proposed middleware is positioned upon existing network stacks. In fact, protocols such as Directed Diffusion [22] can be used to route information from a specific cluster to the base station or other destinations in the network. Nevertheless, further optimization for intra-cluster routing is possible given the cluster structure and connectivity available at the cluster head.

IV. Design Issues and Challenges

To implement the cluster-based architecture described in Section III still requires significant amount of work. In this section, we outline several concrete issues involved in the design and development of the architecture. Some related challenges and tradeoffs are also addressed.

Cluster Control: Due to the dynamic nature of the phenomena being monitored, and the need for collaborative signal processing, it is necessary to develop *on-the-fly self-configuring distributed clustering mechanisms in WSNs*. More specifically, such clustering mechanisms are responsible for forming the initial cluster and performing the follow-on cluster adaptation.

Initially, the guarding nodes that detect the phenomena send out signals (through an ultra-low power paging channel) to activate sleeping nodes. Daemons running on sensor nodes then start the cluster forming protocol to form an initial cluster. The cluster is formed based on a combination of several metrics of sensor nodes, including data accessibility, node capacity, and network connectivity. The most cost-effective node for performing tasks in the resource management layer will be elected as the cluster head. To perform the above procedure requires *distributed and energy-aware protocols for exchanging and comparing the information from multiple sensor nodes in the network*. For instance, it may be necessary to select ten nodes that have the most remaining energy from a pool of hundred nodes (e.g., using techniques proposed in [23]).

A cluster needs to incrementally adapt its location so that moving phenomena can be tracked. The key problem is to *dynamically determine the membership of nodes in the cluster as the phenomena moves*. In [24], an information-driven approach is studied that makes the decision based on information constraints as well as constraints on cost and resource consumption. When the target object is beyond the sensing range of the cluster head, another round of head election is necessary to find a new cluster head.

In both stages, it is crucial to design *efficient mechanisms for maintaining the cluster information at the cluster head and disseminating control messages from the head to cluster members*. Several mature algorithms, such as the minimal spanning tree, can be used to establish intra-cluster routing paths to organize the transmission of packets within the cluster.

Resource Management: The prerequisite of performing resource management is to periodically gather and update the cluster information at the cluster head, including the CCS capabilities and the remaining energy of sensor nodes and the network connectivity. Different policies can be used to determine the style and frequency of the gathering and update procedure such that a reasonable tradeoff between the incurred overhead and the system response time can be achieved. For example, the cluster head may initiate the gathering procedure across the entire cluster every ten seconds, or, a sensor node may send the information to the head if it detects that its remaining energy is lower than some threshold.

Inputs from the application include application structure, data/control flow, resource requirements, and performance constraints. It may also include policies for fidelity adaptation and tradeoff with other applications. *A critical open problem is the development of a formalized general specification for representing the above information.*

To translate the application level performance requirements into system level parameters, one of the key issues is to *establish accurate time/energy cost models for various CCS operations*. Such cost models should be parameterizable by the workload of the operations, settings of the system knobs, and conditions of the operating region. The soundness and accuracy of the cost models are essential for the design of energy-efficient mechanisms [25].

Further, *it is important to identify appropriate optimization metrics for resource allocation and adaptation.* Tradeoffs need to be explored between deliverable QoS and the power consumption; total energy dissipation and the energy balanceness across the system; system fairness and the ability to ensure mission-critical applications. Application specification and performance requirements provide necessary guidance and constraints for such tradeoffs. Furthermore, collaborative processing needs to explore the spatial or temporal correlation of input data. Thus, geographical placement or sensor density requirements of the application need to be considered.

One of the most challenging problems in middleware design is the development of fast, near-optimal algorithms for resource allocation and adaptation. Based on the above cluster information, cost models, and optimization metrics, the algorithms need to determine necessary hardware resources for the upper-level applications and the schedule of applications onto the resources. Consequently, the cluster head sends out control messages regarding the state of sensor nodes, the

setting of system knobs, and the application schedule. In case that the application code is not uniformly distributed in the cluster, code duplication and/or movement [7], [13] might be necessary to carry out the planned schedule.

Numerous efforts have been conducted to study resource management in traditional distributed (real-time) systems. However, to adapt traditional techniques into the context of WSNs needs careful reconsideration of the complexity, scalability, and flexibility of the techniques. The stringent energy constraint and the availability of various system knobs necessitate new mechanisms to efficiently explore the tradeoffs between multiple QoS dimensions among concurrent applications. Further, the underlying wireless network demands new mechanisms for scheduling communication requests, but also provides new opportunities for performance tradeoffs. We illustrate a simple but efficient technique for resource allocation in a single-hop cluster in Section V.

One important way to provide standardized system services to the application is through energy-aware APIs. Performance requirements on latency, reliability, and energy can be explicitly expressed as parameters at function-level. Such fine-grain control complements application-level performance specification in directing the resource manager to fully explore the tradeoffs between energy and application performance.

Inter-Cluster Coordination: In the proposed middleware, information exchange between clusters is necessary for both information sharing and coordination. For instance, data gathered at one cluster can be requested by either the base station or other clusters across the network. Regarding each cluster head as an information source, existing routing protocols such as Directed Diffusion paradigm [22] are still applicable in such scenarios.

The tradeoffs of energy against application fidelity is also important for inter-cluster routing. For instance, an energy-efficient packet scheduling scheme over an existing data gathering substrate is described in [26]. Similar techniques can be applied for information dissemination among clusters.

Another issue arises when multiple clusters overlap with each other. For instance, two separated objects initially tracked by two different clusters may move across or eventually move together, which leads to the overlap of the two clusters. In such a case, multiple clusters may compete for resources. It is therefore important to establish *necessary mechanisms for detecting the existence of overlapped clusters and coordinating between clusters to avoid unfairness, starvation or deadlock during resource competition*.

It is sometimes helpful to perform cluster combination if two clusters overlap on a large portion of geographical area and will co-exist for a long time period. Cluster combination can be used to achieve reduced coordination overhead and increased resource utilization, while necessitating high scalability of the cluster control protocol and resource management algorithms. The counterpart of cluster combination is cluster splitting. Cluster splitting is needed when two close objects tracked by a single cluster begin to move toward opposite directions. However, cluster splitting can also be regarded as the procedure of reducing the application load on the original cluster that tracks one object and forming another cluster for tracking the other object.

Note that the capability of distinguishing multiple and close objects within a cluster is a research issue of application-level signal processing and thus beyond the scope of this article.

V. An Illustrative Technique for Energy-Efficient Resource Allocation

As outlined in Section IV, much work remains to be done to develop key techniques for both the clustering and resource-management layers. Toward such a goal, we discuss one of the key components in the resource management layer and present our technique from [27]. More specifically, we consider the problem of allocating a real-time application onto a single-hop cluster such that the system lifetime of the cluster is maximized. This technique confirms the importance of the utilization of system knobs to achieve energy-efficiency. Moreover, it serves as the basis for the development of dynamic resource adaptation techniques.

A. Problem Description

We consider a set of homogeneous sensor nodes connected by a single-hop wireless network. Each sensor node is equipped with discrete dynamic voltage scaling [3] and modulation scaling [4]. These two techniques are widely used for efficient exploration of the energy-latency tradeoffs for computation and wireless communication activities, respectively. That is, we can reduce the energy dissipation of performing a specific computation or communication activity at the cost of increased latency.

For the sake of illustration, a real-time application that consists of a set of communicating tasks is considered. More precisely, an instance of the application is periodically activated and must be completed before the next application instance is activated. Such an application model is also used in the recently proposed epoch-based system [28], where the period is instantiated by the length of each epoch.

The application is represented as a directed acyclic graph, with the workload of tasks (in terms of CPU cycles), and the workload of communication activities (in terms of packet size) known a priori. The capabilities and cost models of the embedded processors and radios are assumed to be known. Hence, the time and energy costs of both tasks (under different voltage settings) and communication activities (under different modulation settings) can be calculated accordingly.

We are interested in determining an energy-efficient resource allocation of the cluster, including the *assignment* of tasks onto sensor nodes, the *voltage/modulation setting* for executing each task/communication activity, and the *scheduling* of tasks and communication activities. The goal is to maximize the lifetime of the cluster till the first node fails due to depleted battery, while the real-time constraint of the application is satisfied. Hence, an intuitive objective function is to minimize the maximal energy dissipation among all sensor nodes during each application period.

B. A 3-Phase Heuristic

The above problem can be formulated using integer linear programming, from which optimal solutions can be obtained by using commercial tools such as LINDO [29]. However, the time cost for solving an integer linear programming problem is prohibitive for large systems. To efficiently solve the problem, a 3-phase heuristic is proposed in [27].

In Phase 1, we assume an unlimited number of sensor nodes in the cluster. The tasks are partitioned into task-groups, which is a group of tasks to be assigned onto the same sensor node with a specific execution order among them. The goal is to minimize the overall execution time of the application. A key challenge is to schedule the communication activities in a single-hop wireless cluster. We use a simple first-come-first-serve policy in this heuristic.

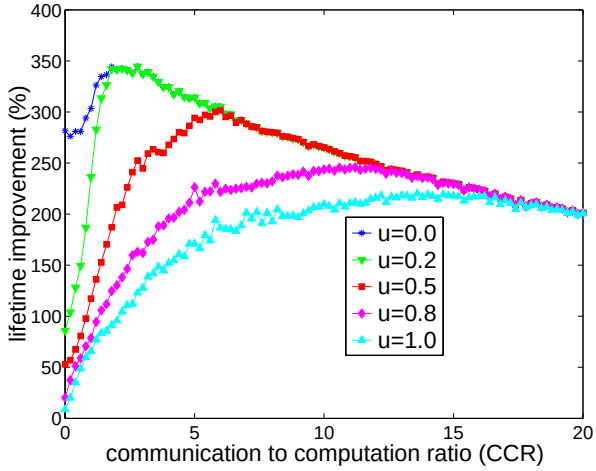
In Phase 2, we use a greedy policy to find an assignment of the task-groups obtained in Phase 1 onto the actual sensor nodes within the cluster, such that the maximal energy dissipation among all sensor nodes is minimized. Note that multiple task-groups can be assigned to the same node, due to the limited number of sensor nodes.

In Phase 3, the voltage and modulation settings of tasks or communication activities are adjusted. An iterative approach is used. In each iteration, we find the task (or communication activity) that by lowering its current voltage (or modulation) settings to the next level, the system lifetime can be increased the most without violating the real-time constraint.

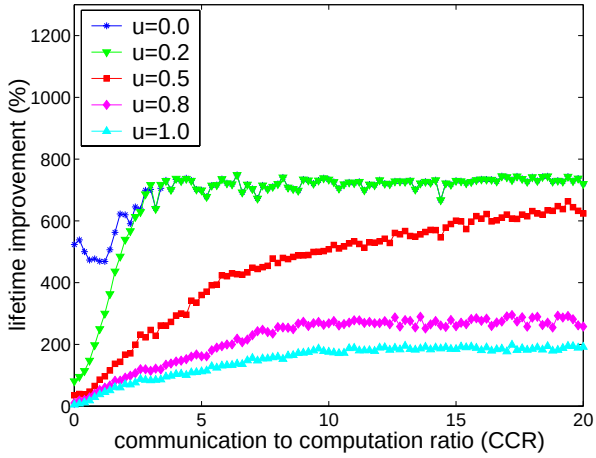
C. Simulation Results

Detailed simulation results on the above 3-phase heuristic are presented in [27]. We study two important system parameters. The first one is called *system utilization* (u), which indicates the tightness of the real-time constraint. The second one is called *communication to computation ratio* (CCR), which signifies the relative heaviness of computation against communication activities, in terms of the time cost.

The simulation results show that the heuristic can result in solutions that are within 60% of the optimal. In Figure 2, we show that a lifetime improvement up to a factor of 8 can be achieved by our heuristic, compared with the case when no voltage and modulation scaling is used. It is also observed that the improvement increases as the system utilization decreases (u approaches 0), which leads to more latency laxity to trade for energy. Moreover, by using modulation scaling, much more improvements can be achieved when CCR is large.



(a) Use dynamic voltage scaling only



(b) Use both dynamic voltage and modulation scaling

Fig. 2. Lifetime improvements achieved by our heuristic

VI. Concluding Remarks

Due to the continuing advances in network and application design in WSNs, the development of an appropriate middleware for WSNs is becoming necessary and possible. A cluster-based middleware architecture to provide a virtual machine model for diverse applications on WSNs has been presented in this article. Application knowledge is required to direct the operations of the middleware. Distributed algorithms are needed to self-configure and adapt clusters for monitoring the target phenomena. To maintain the robustness of the system under stringent resource constraints and high dynamics, the key is to provide flexible and adaptive resource management mechanisms to effect efficient tradeoffs between multi-dimensional performance of multiple applications. Inter-cluster coordination mechanisms are also needed to avoid unfairness, starvation, and deadlock when two clusters compete for the same resource pool in terms of sensor nodes.

However, to implement a successful middleware still needs significant work to resolve a set of technical issues, as outlined in Section IV. The middleware architecture also needs to adapt to newly emerging applications, such as mobile robot-based network [30]. The mobility of sensor nodes would impose additional challenges in both cluster control and resource management. Further, an open problem is the validation and evaluation of the design concept and implementation techniques of the middleware in real WSN environments. The performance of the middleware could be enhanced by identifying proper system parameters for a wide range of environments and application scenarios.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, 2002.

[2] Q. Qiu and M. Pedram, "Dynamic power management based on continuous-time Markov decision processes," in *Design Automation Conference (DAC)*, June 1999, pp. 555–561.

[3] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *USENIX Symposium on Operating Systems Design and Implementation*, Nov. 1994, pp. 13–23.

[4] B. Prabhakar, E. Uysal-Biyikoglu, and A. E. Gamal, "Energy-efficient transmission over a wireless link via lazy packet scheduling," in *IEEE InfoCom*, 2001.

[5] K. Römer, O. Kasten, and F. Mattern, "Middleware challenges for wireless sensor networks," *ACM SIGMOBILE Mobile Communication and Communications Review*, vol. 6, no. 2, 2002.

[6] A Network Virtual Machine for Real-Time Coordination Services. [Online]. Available: http://www.cs.virginia.edu/nest

[7] Smart Message Projet. [Online]. Available: http://discolab.rutgers.edu/sm

[8] P. Ramanathan, K.-C. Wang, K. K. Saluja, and T. Clouqueur, "Communication support for location-centric collaborative signal processing in sensor networks," in *DIMACS Workshop on Pervasive Networks*, May 2002.

[9] X. Yu, K. Niyogi, S. Mehrotra, and N. Venkatasubramanian, "Adaptive middleware for distributed sensor networks," *IEEE Distributed Systems Online*, May 2003.

[10] A. Murphy and W. Heinzelman, "MiLan: Middleware linking applications and networks," University of Rochester, Tech. Rep. TR-795, 2002.

[11] Cougar Project. [Online]. Available: http://www.cs.cornell.edu/database/cougar

[12] TinyOS website. [Online]. Available: http://webs.cs.berkeley.edu/tos/

[13] P. Levis and D. Culler, "Maté: A tiny virtual machine for sensor networks," in *10th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2002.

[14] G. Coulson, S. Baichoo, and O. Moonian, "A retrospective on the design of the gopi middleware platform," *ACM Multimedia Journal*, vol. 8, no. 5, pp. 340–352, Dec. 2002.

[15] S. S. Yau and F. Karim, "Reconfigurable context-sensitive middleware for ADS applications in mobile ad-hoc network environments," in *5th IEEE International Symposium on Autonomous Decentralized Systems*, May 2001.

[16] S. Maffeis, "Communication middleware for mobile applications - a comparison," http://www.softwired-inc.com/people/maffeis/articles/softwired/corba_vs_jms.pdf.

[17] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1999, pp. 263–270.

[18] W. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application specific protocol architecture for wireless microsensor networks," *IEEE Trans. on Wireless Networking*, 2002.

[19] M. Singh and V. K. Prasanna, "A hierarchical model for distributed collaborative computation in wirelss sensor networks," in *5th Workshop on Advances in Parallel and Distributed Computational Models*, 2003.

[20] M. Younis, M. Youssef, and K. Arisha, "Energy-aware routing in cluster-based sensor networks," in *International Symposium on Modeling, Anaysis and Simulationof Computer and Telecommunication Systems*, Oct. 2002.

[21] C. E. Perkins, Ed., *Ad Hoc Networking*. Addison-Wesley, 2001, ch. Cluster-Based Networks.

[22] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[23] M. Singh and V. K. Prasanna, "Optimal energy-balanced algorithm for selection in a single hop sensor network," in *IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, May 2003.

[24] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration for tracking applications," *IEEE Signal Processing Magazine*, Mar. 2002.

[25] R. Min and A. P. Chandrakasan, "Top five myths about the energy consumption of wireless communication," *ACM SIGMOBILE Mobile Communication and Communications Review*, vol. 6, no. 4, 2002.

[26] Y. Yu, B. Krishnamachari, and V. K. Prasanna, "Energy-latency tradeoffs for data gathering in wireless sensor networks," to appear in IEEE InfoCom 2004.

[27] Y. Yu and V. K. Prasanna, "Energy-balanced task allocation for collaborative processing in wireless sensor networks," accepted by MONET special issue on Algorithmic Solutions for Wireless, Mobile, Ad Hoc and Sensor Networks.

[28] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation service for Ad-Hoc sensor networks,," in *Symposium on Operating Systems Design and Implementation (OSDI)*, Dec. 2002.

[29] L. Schrage, *Linear, Integer, and Quadratic Programming with LINDO*. Redwood city, CA: The Scientific Press, 1986.

[30] A. Howard, M. J. Mataricié, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *6th International Symposium on Distributed Autonomous Robotics Systems*, June 2002.