A DISTRIBUTED SYSTEM FOR MAD-COW DISEASE MONITORING AND TRACKING

Lei Zhang[1], Alvin S. Lim[2], Hui Song[1], and Xinliang Zheng[1]

[1]Dept. of Computer Science, Frostburg State University, Maryland, USA
lzhang@frostburg.edu, hsong@frostburg.edu, xzheng@frostburg.edu
[2]Dept. of Computer Science, Auburn University, Alabama, USA
lim@eng.auburn.edu

Received: 04th March 2019 Revised: 23rd May 2019 Accepted: 30th May 2019

ABSTRACT

Advances in hardware technology and wireless communication have enabled the deployment of large-scale sensor networks, where thousands to millions of self-powered, small and low-cost sensor nodes are distributed over a vast field to obtain sensing data. These sensor nodes are equipped with sensing, communicating, and data processing units, which allow sensor nodes to collect, exchange, and process data of the environment. The processing units used in the current generation of sensor nodes are already powerful enough to perform some complicated algorithms to process sensing data, and they are expected to be more powerful in the future. Due to these attractive characteristics, wireless sensor networks are ideal candidates for a wide range of civil applications and military operations. This paper provides an in-depth study of applying wireless sensor networks to real-world mad-cow disease monitoring and beef distribution safety system. An extensive survey of the state of the art to design a distributed system for pervasive computing is conducted. A set of techniques and mechanisms are compared and ranked in the paper. Then the characteristics for designing this kind of pervasive system are listed; the system architecture is presented; and an instance of the key mechanisms for monitoring the mad-cow disease and tracking beef distribution system is presented. This system supports real-time communication and multitasking scheduling as well.

KEYWORDS

Sensor Networks, Distributed System, Mad-Cow Disease (MCD)

1. INTRODUCTION

Since Mark Weiser described the vision of pervasive computing in his seminal paper [1], a lot of progress has been achieved in this area. With the advance in MEMS devices and embedded processors and radio, it will soon be feasible to deploy large-scale sensor networks to perform distributed microsensing and control of physical environment [2]. For example, Civil engineers are using motes to monitor building integrity during earthquakes [3]; biologists are planning mote deployments for habitat monitoring [4], [5]; administrators of large computer clusters are using motes to monitor the temperature and power usage in their data centers.

Recently, people are considering using sensors to monitor and track the mad-cow disease. The fast spread of mad-cow disease has caused severe results in the past. People got contaminated and economics was also affected by this event. To avoid disease spread out and track the distribution of beef, recording sensors have been put in different processing procedures [6]. The smart sensors and actuators are equipped with low-power processors and short-range radio

transceivers. They will automatically form multi-hop ad hoc networks to communicate with other sensors and remote base stations as well.

Radio Frequency Identification (RFID) (Figure 1) has been used by PM Beef on cow/calf farms, feedlots and PM's harvest and fabrication facility in Windam, MN [7]. However, those RFIDs act only as identification: They do not group into sensor networks.

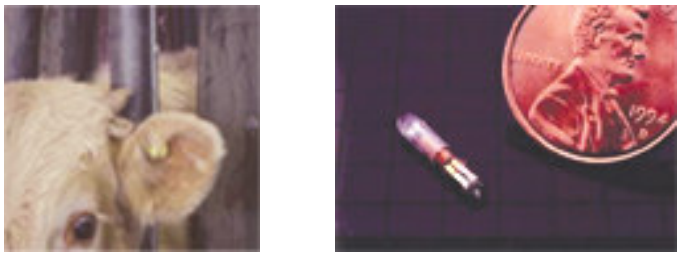


Figure 1. RFID tags

This paper presents a specific case of distributed systems that can be used in a mad cow disease monitoring and tracking system (MCDMS). Different from the RFID used by PM Beef, we assume that we have sensors capable of detecting mad cow disease and other sensors that can be used to track the cattle's body parameters, such as temperature, heart beating rate, blood pressure, etc. These sensors can be attached to wireless devices that are clipped on the ears of cows. Furthermore, we also study general solutions to develop effective sensor network architecture in the meadow domain.

This paper presents a specific case of distributed systems that can be used in a mad cow disease monitoring and tracking system (MCDMS). Different from the RFID used by PM Beef, we assume that we have sensors capable of detecting mad cow disease and other sensors that can be used to track the cattle's body parameters, such as temperature, heart beating rate, blood pressure, etc. These sensors can be attached to wireless devices that are clipped on the ears of cows. Furthermore, we also study general solutions to develop effective sensor network architecture in the meadow domain.

The rest of the paper is organized as follows. The related work is discussed in Section 2. Section 3 presents an extensive survey on the existing techniques and mechanisms of pervasive computing architecture. In Section 4, we first analyze the characteristics of the disease monitoring and tracking system and then provide the key mechanisms in design of the system. Conclusion remarks are discussed in Section 5.

2. RELATED WORK

Sensor networks are ideal candidates for a wide range of applications and will be ubiquitous in the near future, serving as the bridge between humans and the physical world. They are expected to contribute significantly to pervasive computing and make our daily life more comfortable and enjoyable.

Sensor network was first proposed for military applications such as battlefield surveillance [8] and military situation awareness [9]. In another paper [10], Srisathapornphat et al. propose to use sensor networks to sense intruders on bases, to detect enemy units movements on land or sea, to detect chemical (or biological) threats, and to offer logistics in urban warfare. Recently,

researchers propose to use sensor network to form a smart-landmine network which can destroy the intruding targets using the minimum-cost pre-deployed mines [11].

The research on sensor network then quickly expands to civil applications, in which sensors are deployed to sense and collect data. Under this category, the existing applications can be classified into habitat monitoring, environment observation and forecasting, health, structure monitoring, and other commercial applications [12].

Cerpa et al. [13] describe habitat monitoring as a motivating application for wireless sensor network and introduce initial system building blocks designed to support such a system. They propose a tiered architecture and use a frisbee model to optimize energy-efficiency so as to prolong the lifetime of the network. In August 2002, researchers from UCB/Intel Research Laboratory deployed a tiered sensor network (using Mica motes) on Great Duck Island, Maine, to monitor the behavior of storm petrel [14]. Other research issues of using sensor network for habitat monitoring discussed in several other papers [15], [16], [17].

CORIE [18] is a pilot environmental observation and forecasting system (EOFS) for the Columbia River, Oregon. Its observation network, a real-time sensor network, includes an extensive array of 13 stationary sensor nodes in the Columbia River estuary and one mobile sensor station cruising the river several times over the years. Sensor data are collected and then transmitted via wireless link toward onshore master stations, where they are further forwarded to a data management system and finally fed into advanced numerical models. The acquired knowledge is transformed into data products to provide objective insights on spatial and temporal variability of the river, to understand the river-dominated estuaries and plumes, and to provide powerful planning and analysis tools for policy making for the region's natural resource management and regulation authorities.

In the area of health care, sensor network can be used for monitoring human physiciological data remotely, tracking and monitoring of doctors, patients, and drug administrator inside a hospital [19]. Milenkovic et al. [20] present a prototype sensor network for personal health monitoring that utilizes the off-the-shelf 802.15.4 compliant sensor nodes and custom-built motion and heart activity sensors. In such application, sensor network is promising in revolutionizing health care by allowing inexpensive, non-invasive, continuous, ambulatory health monitoring with almost real-time updates of medical records via the Internet.

Sensor network has significant commercial potential in structure monitoring, to collect and analyze the structural response to ambient or forced excitation. A first generation of sensor networks for structural monitoring is presented by Xu et al. [21], called *Wisden*, which is a data acquisition system that collect data and a single node for centralized processing. It mimics wired data acquisition systems and incorporates reliable data transport, time synchronization, and compression algorithms.

In his master thesis, Kim [22] design, implement, and test a wireless sensor network (targeting a deployment on the Golden Gate Bridge) to monitor the ambient vibration of a structure to analyze the collected data to determine the health status of the structure. Deployment at a footbridge showed the system is operating successfully, and the collected data matched theoretical expectation.

Li and Liu [23] propose a structure-aware self-adaptive wireless sensor network system to monitor the environment in coal mines, which can detect structure variation caused by underground collapse rapidly. The collapse holes can be located and outlined, and the detection accuracy is bounded with such a system. They deployed a prototype system with 27 Mica2 motes and measured the system error, detection latency, packet loss rate, and network

bandwidth. Based on the data collected in experiments, they also conducted a large-scale simulation to evaluate the system scalability and reliability.

Sensor network can also be used in many other areas, such as public safety [24], [25] (sensing and location determination at disaster sites), coordinated vehicle tracking [26], vehicle theft detection and tracking [27], smart badges and tags [24], [25], monitoring hazardous chemical levels and fires [24], managing inventory, monitoring product quality [28], [29], and many others.

As pointed out by Akyildiz et al. [30], The characteristics of sensor networks, such as low cost, easy deployment, flexibility, fault tolerance, and high sensing fidelity, will continue motivating and creating many new and exciting applications for remote sensing. In the future, this wide range of applications will make sensor networks an integral part of our lives.

3. SYSTEM ARCHITECTURE FOR MAD COW DISEASE

3.1. Characteristics of Mad Cow Disease Monitoring and Tracking System

As described in Section 1, the major functions of our mad cow disease monitoring and tracking system are twofold. One part is to track each cattle, from its birth to slaughtering house, to detect mad cow disease. The other part, deployed on transportation trucks and in the beef process, is to track the beef until beef is put on the grocery shelf. The tiny sensor device is able to detect virus, read the current level of body temperature, measure the heart-beating rate and blood pressure for each cow. Our system is to provide disease detection and safety processing monitoring. There are different kinds of events and sorts of queries, which can be divided into categories:

1) Urgent events that only include mad cow disease detection. Once the urgent events are observed, the sensor network should be able to report them to the closest base stations and the control centers. These events may not require any aggregation and must be forwarded by each intermediate node.

2) Routing observation events that include temperature measurement, heart beating counting, blood-pressure recording and other routine tracking parameter of the cow. Since each node on each cow can generate an amount of such information, sending all data to the base stations and processing the low-level observation externally would be very expensive in terms of energy and bandwidth consumption. It is improved in our system through using in-network processing to produce high-level events. This may involve a combination of local techniques (e.g. exchanging data with neighbor nodes) and global techniques (e.g. comparing temperature readings against the average reading in the network).

In addition to the various types of events in the system, there are also queries initialized by users to retrieve data in the networks. The queries generated can also be divided into two categories:

1) Group data query which looks like “What’s the average temperature of cows in barn 1” or “What’s the average temperature of cows in the field.” If the event information is stored locally, then queries must be flooded to all nodes. But if event information is stored using data-centric storage, the query can be sent to the sensor node associated with that event.

2) Disseminated data query which may be generated when one specific cow needs to be monitored. Disseminated data query is rare except when emergency happens, but it is required to be flooded into the whole sensor network.

Following the introduced characteristics above, we design our system architecture in the next section.

3.2. System Architecture Design

The monitoring system that is shown in Figure 2 has a tiered architecture. Each sensor can only communicate with its neighbors and it can extend the access path to a base station using directed diffusion routing protocol.

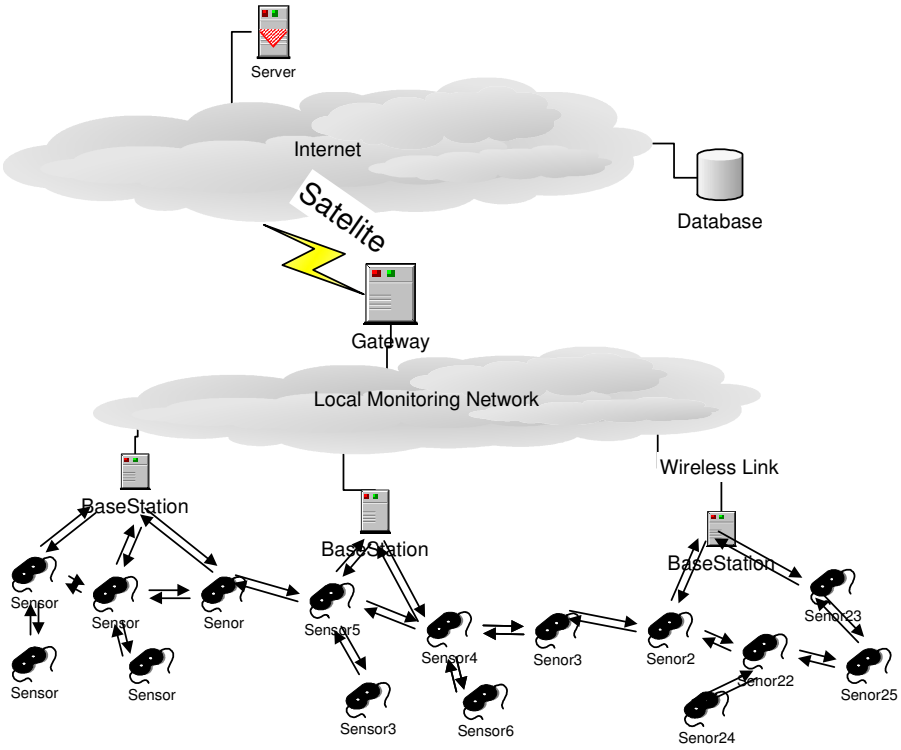


Figure 2. System architecture of mad cow disease monitoring and tracking system

Base stations can be devices that have higher processing capability and more energy source, such as PDAs. They are installed in the barn, stones or trees in the grass field, in the transportation trucks and slaughtering houses. All base stations, grouped into a local monitoring network, can forward information to a central base station which connects with the Internet and can save the data information or forward them to the control center. When the base station receives the integrated data information from the sensor network, it either forwards the information to the central base station (marked as a gateway in Figure 2) or stores the information to wait for more information to forward. While for urgent data, it forwards immediately.

A database server is connected to the Internet, which provides statistical data for the control server. Queries toward to the sensor networks are transmitted to the local monitoring network, each base station will accept it. But one or some of them may broadcast the query in their sensor networks using directed diffusion protocol.

3.3. Distributed System

The distributed system is dedicated to the sensor network that includes the base station the sensors connected to. The system view is shown in Figure 3 that is a tiered architecture. The distributed system falls between sensor and network layer and application layer. It is able to provide API for up-layer applications. The lowest level consists of the sensor nodes that perform general purpose computing and networking, in addition to application-specific sensing.

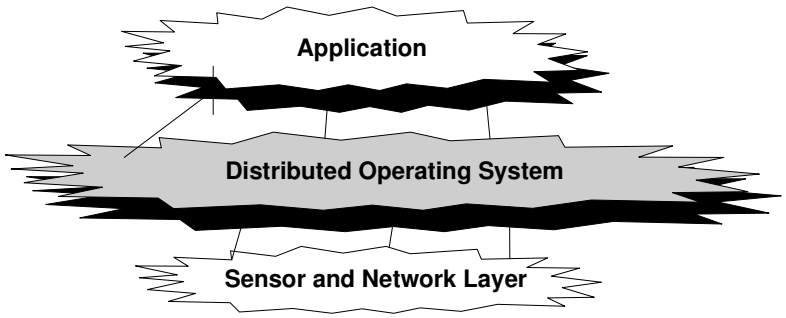


Figure 3. System architecture overview

Figure 4 illustrates the designed system which includes six components: a task scheduler, concurrency control, power management, communication and invocation mechanism, and resource discovery and database management.

The task scheduler provides a priority queue with two levels of priorities. Urgent tasks such as virus detection, abnormal temperature and heart rate are assigned a higher priority. The higher priority can preempt the lower priority tasks. This task scheduler provides real-time communications.

The concurrency control is the same as TinyOS [31] that task-level concurrency is provided. This concurrency provides fast switching which helps to save energy.

The power management adopted idle listening with periodic listening. The lower level sensors can be set to power save mode to only response to incoming events. And some sensors can be set to power off for a short of period.

Resource discovery is used by sensors to discover service from the base station if the sensors are one hop away from the base station and the neighbor nodes if the node is away from the base station.

Our system adapts data-centric storage mechanism provided by database management component. It supports data aggregation in the network. This mechanism decreases the data information propagation in the network, leveraging the computation and communication tradeoff.

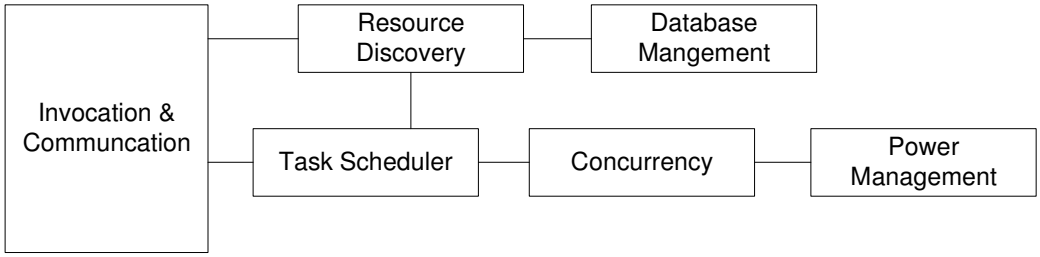


Figure 4. Distributed system for mad cow disease monitoring and tracking system

4. TECHNIQUES AND MECHANISMS

4.1. Task scheduler

Usually, each sensor node is equipped with more than one sensor to support more applications. For instance, in our mad cow disease monitoring and tracking system, each node can have virus detection sensor, temperature reading sensor, heart beating counting sensor and blood pressure testing senor. Every node should be able to send/receive bit information that an antenna is required. To schedule the tasks triggered by the events or generated in the application layer, the system has to provide an appropriate task scheduling mechanisms.

Due to the special characteristics of the mad-cow disease system, our multi-hop wireless sensor network system adapts real-time communication architectures, event-driven OS with priority-based multitasking, inter-task communication and synchronization. In the real time tracking system, there are two kinds of events in different locations, normal and abnormal. Abnormal events are exceptional symptoms of the cows, such as abnormal body temperature, blood pressure and the heart beating, which are to be processed immediately. Under this emergency conditions, both deadline-aware and distance-aware in the task scheduling mechanisms should be concerned. By deadline-aware, we mean that a shorter deadline has a higher priority. Each deadline is decided by the event urgency. By distance-aware, we mean that a longer distance has a higher priority. From the fairness point of view, our communication scheduling policies to deal with the urgent events, have to balance the weight of both time and space.

In order to find a method to implement of this cognizant time and space balance in the real time architecture of mad-cow disease scheduling system, we conduct wide range of survey. A novel policy called Velocity Monotonic Scheduling (VMS) proposed by [32], in which packet priority is decided based on both distance and deadlines is suitable for packet scheduling in our sensor networks. Based on a notion of packet requested velocity, the scheduling mechanism assigns the priority to a packet. A packet with a higher requested velocity is assigned a higher priority. Since this scheduling mechanism assigns the “right” priorities to packets based on their different deadlines urgencies on the current hop, it improves the number of packets that meet their deadline. In addition, this mechanism also solves the fairness problem that packets far away from the base station will tend to have higher priorities when it competes against other packets that are closer to the destination. Each packet is expected to make its end-to-end deadline if it can move toward the destination at its requested velocity, which reflects its local urgency. This mechanism can outperform deadline-based packet scheduling because velocity more accurately reflects the local urgency at each hop when packets with the same deadline have different distances to their destinations. In this way, both the weights of time and space in the urgent events have been balanced.

Each packet is assigned a priority based on its requested velocity and queued at the network layer when there are multiple outstanding packets. Several options are available for implementing priority queues.

The approach we apply to the system queue priority is to maintain multiple FIFO queues, each of which is corresponding to a fixed priority level. Each priority corresponds to a range of requested velocities. A packet is first mapped to a priority, and then inserted into the FIFO queue that corresponds to its priority. This approach is more efficient because no ordering needs to be performed for every incoming packet. The per-packet overhead is logarithmic only in the number of priority levels, not the number packets. There are other kinds of packet scheduling schemes, which have their own advantages but don’t fit our system very well.

In TinyOS, the scheduler is power aware: the prototype puts the processor to sleep when the thread queue is empty, but leaves the peripherals operating, so that any of them can wake up the

system. This behavior provides efficient battery usage. Once the queue is empty, another thread can be scheduled only as a result of an event, thus there is no need for the scheduler to wake up until a hardware event triggers activity. More aggressive power management is left to the application. But this FIFO without priority does not support real time communication. It is small, which needs very small size of RAM.

In [33], Job Mulder et al. considered that the system which provides flexible application software support should offer at least real time scheduling mechanisms, memory management, and resource management. PEEROS proposes a new scheduling algorithm – EDFI [34] to provide more functionality within limited resources. Each task has a deadline or defined priority and the lowest priority task can use infinite loops. Infinite loops may cause higher energy consumption. PEEROS is a combination of TinyOS and other multitask system. It is written in C language that causes the system 10 times bigger than TinyOS, which causes PEEROS requires big memory support.

4.2. Communications and Invocation

In this section, by analyzing the characteristics of different communication models, we select one proper communication model from many existing models and exploit it into our system.

Client/Server Model [35]

Traditional Distributed Sensor Networks, using the client/server model, clients send data to the servers where data processing tasks are carried out. This method shows its advantages in the way of resources rational utilization by keeping the local sensors as simple as possible and let the processing elements to do the complicated jobs. Though, this model has its own advantages, it also introduces some factors which don't match the requirements of our system. The problems for Client/Server Model are not scalable, not appropriate for real time communication, can't respond to instance load change, lots of energy consumed for transmission.

Agent [36]

To improve the performance of traditional distribute of sensor network, a new method called computer agent appeared: data stay at the local site, while the integration process is moved to the data sites. By transmitting the computation engine instead of data, this paradigm offers some benefits:

Network bandwidth requirement is reduced. Instead of passing large amounts of raw data over the network through several round trips, only the agent with small size is sent. Generally speaking, agent is a special kind of software that can be executed autonomously. Once dispatched, it can migrate from one node to another node, performing data processing autonomously, while software can typically only be executed when being called upon.

This is especially important for the real-time applications where the communication is through low-bandwidth wireless connections:

1. Better network scalability. The performance of the network is not affected when the number of sensor is increased. Agent architectures that support adaptive network load balancing could do much of a redesign automatically.

2. Extensibility. Mobile agents can be programmed to carry task adaptive fusion processes that extend the capability of the system.

3. Stability. Mobile agents can be sent when the network connection is alive and return results when the connection is re-established.

Therefore, the performance is not much affected by the reliability of the network.

Though this new approach makes a bigger progress in the traditional distributed communication, it also increases the overhead to the system. Each sensor should have the ability to process the data it obtains and the transition of the agents (software) also consumes the bandwidth and energy of each node. In this way, it doesn't fit our system very well, because each sensor in our system should be as simple and energy efficient as possible. We find a more improved model than agent, Active Message in [37].

Active Message [37], [38]

Active Message proposed by [37], [38] is an asynchronous communication mechanism intended to expose the full hardware flexibility and performance of modern interconnection networks. It has the ability to overlap communication and computation and reduction communication overhead at the same time. It minimizes the software overhead in message passing nodes and utilizes the full capability of the hardware. The basic idea is each message contains at its head the address of a user level handler that is executed on message arrival with the message body as argument. The role of the handler is to get the message out of the network and into the computation ongoing on the processing node. The handler must execute and complete quickly. This corresponds closely to the hardware capabilities in most message passing multiprocessors where a privileged interrupt handler is executed on message arrival, and represents a useful restriction on message driven processors. Under Active Messages the network is viewed as a pipeline operating at a rate determined by the communication overhead and with latency related to the message length and the network depth. The sender launches the message into the network and continues computing; the receiver is notified or interrupted on message arrival and runs the handler. To keep the pipeline full, multiple communication operations can be initiated from a node, and computation proceeds while the messages travel through the network. To keep the communication overhead to a minimum, Active Messages are not buffered except as required for network transport. Much like a traditional pipeline, the sender blocks until the message can be injected into the network and the handler executes immediately on arrival. Tolerating communication latency has been raised as a fundamental architectural issue; this is not quite correct. The real architectural issue is to provide the ability to overlap communication and computation, which, in turn, requires low overhead asynchronous communication.

Active message is the best way to design dynamic network systems. The efficiency of this model includes the elimination of buffering beyond network transport requirements, the simple scheduling of nonsuspensive message handlers, and the arbitrary overlap of computation and communication. By drawing the distinction between message handlers and the primary computation, large grains of computation can be enabled by the arrival of multiple messages.

Our system employs the active message mechanism. The sensor clipped on the cow's ear is to detect virus and check the temperature and heartbeat of the cow. Once the application layer has the requirement for communication, the sensor sends its data to the system that groups the data into active message packets to be sent out. After its neighbors receive the active message, it behaves according to the header information in the active message, aggregating data, forwarding data, or dropping the message. Since active message is very small amount of control information, it doesn't produce lots of network traffic. In this way, data information can be processed locally and sending data back to base station only in emergency, so that there isn't much transition. The implementation of active message is efficient way to deal with the problems of energy efficiency and appropriates resource utilization.

4.3. Resource Discovery

Resource discovery is very important in design a distributed system. Jini's service discovery [39] is implemented base on TCP and UPD; it relies on mobile Java codes. It is not clear how these may be implemented using data-centric, ad-hoc sensor networks with services based on more generic mobile codes. Service Location Protocol (SLP) [40] is an IETF protocol for service discovery that is designed solely for IP-based networks. Bluetooth [41] devices have a range of 10 meter and can directly communicate with at most seven other Bluetooth devices in a piconet. Bluetooth Service Discovery Protocol (SDP) [42] allows devices to browse and retrieve services by matching service classes or device attributes. Only services within the range of the device are returned.

Both Jini's service discovery and SLP are not suitable for our sensor networks system because they are base on TCP/UDP or IP networks. Due to distance limitation of Bluetooth, its SDP can only retrieve service with range of 10 meters. The lookup service provided by [42] may retrieves services that could be multiple hops from the requesting node.

A system architecture proposed by Lim in [42], which is specialized in sensor network systems enlightens us on our system design. The whole sensor network infrastructure is divided into three layers:

1. Application systems. For example, sensor information processing layer and collaborative signal processing

2. Configurable distributed systems that provide distributed services to the application systems

3. Sensor networking and physical device layer that routes messages through the ad-hoc sensor network.

The system adopts data-centric directed diffusion protocol to implement all the distributed services and for retrieve of data through dynamically changing ad-hoc sensor networks. Distributed services and applications use the publish/subscribe API provides by directed diffusion. To enable the ability to reconfigure sensor networking, configuration, and adaptation functionalities, the sensors may make use of three main classes of distributed services: Lookup service, composition service, and adaptation service. The lookup service enables new system and network services to be registered and made available to other sensor nodes. The composition service allows sensor nodes to be formed as clusters and provides the management of the cluster. The adaptation service allows sensor nodes and clusters to reconfigure dynamically to support node mobility, failure ad spontaneous deployment. These servers enable sensor nodes to form community in ad hoc networks, support self-configuration and adapt to real-time information changes and events. These servers may be replicated for higher availability, efficiency and robustness.

In mad cow disease monitoring and tracking system, base stations are installed in the cow barn, farming field and transporting truck. They have more powerful processing capacity and more energy sources comparing with the sensor nodes clipped on the cow ears. The base station may provide all the three services. A cluster of sensors can group to a network connected with one base station. In directed diffusion, initially, the query will flood in the network. Once the base station receives data from its networks, it'll reinforce the shortest path. The base station manages the mobility of the sensors.

4.4 Pervasive database management

The real time tracking information of each cow can be very big. So an appropriate data management method is necessary for the system. If event information is stored locally then queries must be flooded to all nodes (unless the user has prior knowledge about the location of the event). If event information is stored using data-centric storage [43], the query can be sent to the sensor network nodes associated with that event name.

TinyDB [44] is a query processing system for extracting information from a network of TinyOS sensors. TinyDB provides a metadata catalog to describe the kinds of sensor readings that are available in the sensor network. It uses a declarative query language to describe the data. This makes it easier to write applications, relative independent from the sensor network changes. TinyDB manages the underlying radio network by tracking neighbors, maintaining routing tables, and ensuring that every node in the network can efficiently and (relatively) reliably deliver its data to the user. TinyDB allows multiple queries to be run on the same set of motes at the same time. Queries can have different sample rates and access different sensor types, and TinyDB efficiently shares work between queries when possible. To adapt a new sensor node to the network, downloading the standard TinyDB code to new motes, and TinyDB does the rest. TinyDB motes share queries with each other: when a mote hears a network message for a query that it is not yet running, it automatically asks the sender of that data for a copy of the query, and begins running it. No programming or configuration of the new motes is required beyond installing TinyDB.

TinyDB is much easier to program and very easy to adapt new motes into the network. But TinyDB maintains a routing table, and monitors its neighbor to deliver data, which may cost a lot of energy, but provides robustness. Since in our system, the underlying OS isn't as same as TinyOS, it is impossible to apply the whole TinyDB to our system. The basic theoretic frame work of our system can imitate TinyDB, but the necessary coding modification is needed in real implementation.

In [45] divided the queries in an ALERT system into three categories:

1. Historical queries, which typically aggregate queries over historical data obtained from the device network.

2. Snapshot queries that concern the device network at a given point in time.

3. Long-running queries, which concern the device network over a time interval.

We may have all these queries in our mad cow disease monitoring and tracking system. For instance, we may want to see the temperature curve for a cow in its lifetime. Once a mad cow disease is detected, we may want to retrieve the current cow body parameters in that area. Also, we can query every node for body parameters in a certain time interval.

Other data management requirements are less universal across the three categories but yet must be addressed in order to support a comprehensive ubiquitous computing environment. For example, the issue of mobility raises a number of issues. First, the fact that the terminals (i.e. devices) are constantly moving, and often have limited storage capacity means that a ubiquitous computing system must be able to deliver data to and receive data from different and changing locations. This results in the need for various kinds of proxy solutions, where users are handed off from one proxy to another as they move. Protocols must be constructed in such a way as to be able to tolerate such handoffs without breaking. Mobility also raises the need for intelligent

data staging and pre-staging, so that data can be placed close to where the users will be when they need it (particularly in slow or unreliable communications situations).

In [46] the authors describe two data management projects, one is called DataRecharging project, which aims to provide data synchronization and dissemination of highly relevant data for mobile users based on the processing of sophisticated user profiles; the other is called Telegraph project, which is developing a dynamic dataflow processing engine to efficiently and adaptively process streams of data from a host of sources ranging from web sources to networks of sensors.

4.5. Power Management

The fundamental constraint on a networked sensor is its energy consumption, since it may be either impossible or not feasible to replace its energy source. So to increase its usable lifetime with restricted energy consumption is very important. In addition to power control designs on Micro sensors in hardware, cross-layer power management [47] is also a feasible way.

TinyOS manages power management through the interaction of three elements (see Figure 5). First, each service can be stopped through a call to its StdControl.stop command; components in charge of hardware peripherals can then switch them to a low-power state. Second, the HPLPowerManagement component puts the processor into the lowest-power mode compatible with the current hardware state, which it discovers by examining the processor's I/O pins and control registers. Third, the TinyOS timer service can function with the processor mostly in the extremely low power power-save mode.

TinyDB uses these features to support sensor network deployments that last for months. In this context, idle listening dominates energy consumption. Low-power listening reduces the cost of idle listening by increasing the cost of transmission. However, instead of low power listening, TinyDB uses communication scheduling. Using coarse-grained (millisecond) time synchronization, TinyDB motes coordinate to all turn on at the same time, sample data, forward it to the query root, and return to sleep.

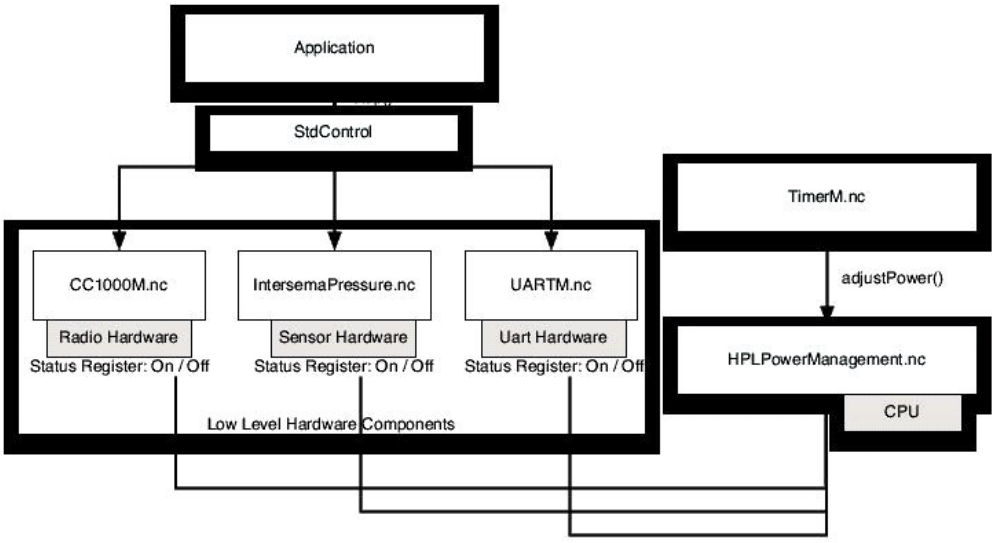


Figure 5: The TinyDB power management API. The application calls StdControl.stop to halt the low-level hardware. HPLPowerManagement.nc sees changes to the hardware status registers, which causes it to put the CPU into a low-power sleep state [38]

To further reduce the average power consumption of the network, low power listening can be combined with the periodic listening. Running both schemes simultaneously results in listening at reduced power for only a fraction of the time. But periodic listening may cause query failure when a query arrives within interval of two listening period.

These techniques attempt to minimize the energy usage at all levels of system operation. Thus, in addition to minimizing energy usage, the system lifetime can be increased by modifying the task allotment according to the available energy at network nodes. Some examples of such techniques can be found in [48, 49, 50, 51] for routing and data gathering. The first requirement for these techniques is to get the information about energy availability at the nodes. The remaining energy in a battery can be estimated from its discharge function and measured voltage supplied [51].

Also, in [52], the author proposed environmental energy harvesting framework (EEHF) toward energy harvesting to adaptively learn the energy properties of the environment and the renewal opportunity at each node through local measurements, make the information available in a succinct form for use in energy aware task assignment such as load balancing, leader elections for clustering. This is a way to allocate task with the spatial-temporal characteristics of energy availability.

It is very complex to implement energy harvesting in the network because the harvesting process itself cost energy, too. Idling listening adopted in TinyOS is a good mechanism to save energy and manage the power efficiently.

To further reduce the average power consumption of the network, low power listening can be combined with the periodic listening. Running both schemes simultaneously results in listening at reduced power for only a fraction of the time. But periodic listening may cause query failure when a query arrives within interval of two listening period. Virus detecting sensor should always awake to report abnormal detection, while other sensors can set into sleep mode if no readings are needed.

5. CONCLUSION

In this paper, we conducted a wide survey of the key mechanisms and techniques used in distributed sensor networks. We compared several task scheduling from FIFO to multi-task scheduling mechanism. Three techniques used in communication and invocation are compared. I pointed out that Active Message is more suitable for sensor network comparing client/server model and software agent. Lookup service is much better than the Jini, SLP from IETF, and Bluetooth SDP. In pervasive database management, TinyDB provides a query processing system for extracting information from a network of TinyOS. Because it relies on TinyOS supporting, it is not suitable for my system. Idle listening power manage is better than other mechanism. It is easy to implemented and efficient. Also, we analyzed the characteristics of mad cow disease monitoring and tracking system, and, finally, provided the system architecture and an instance of the distributed system.

REFERENCES

[1] Satyanarayanan, M. (2001) "Pervasive computing: Vision and challenges". *IEEE Personal Communication*.

[2] Intanagonwiwat, C., Govindan, R., and Estrin, D. (2000) "Directed diffusion: A scalable and robust communication paradigm for sensor networks". *Proceedings of the Mobicom'00*.

[3] Berkeley, U. (2001), "Smart buildings admit their faults". Website, http://coe.berkeley.edu/labnotes/1101.smartbuildings.html.

[4] Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., and Anderson, J. (2002) "Wireless sensor networks for habitat monitoring". *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*.

[5] Cerpa, A., Elson, J., D.Estrin, Girod, L., Hamilton, M., and Zhao, J. (2001) "Habitat monitoring: Application driver for wireless communications technology". *Proceedings of the ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*.

[6] (2001). Website, http://www.destronfearing.com/.

[7] Destron Technologies, "Pm beef uses digital angel products in usda process verified program". Website, http://www.geneticasysid.com.br/Artigo17.htm.

[8] Pottie, G. and Kaiser, W. (2000) "Wireless Integrated Network Sensors". *Communications of the ACM*, 43.

[9] Doumit, S. S. and Agrawal, D. P. (2002) "Self-organizing and energy-efficient network of sensors". *Proceedings of MILCOM 2002*, October, vol. 2, pp. 1245–1250.

[10] Srisathapornphat, C., Jaikaeo, C., and chung Shen, C. (2001) "Sensor information networking architecture and applications". *IEEE Personal Communications*, 8, 52–59.

[11] Liu, C. and Cao, G. (2009) "Minimizing the cost of mine selection via sensor networks". *in Proc. of IEEE INFOCOM*, April, pp. 2168–2176.

[12] Xu, N. (2002) "A survey of sensor network applications". *IEEE Communications Magazine*, 40.

[13] Cerpa, A., Elson, J., Estrin, D., and Girod, L. (2001) "Habitat monitoring: application driver for wireless communications technology". *In ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*.

[14] Mainwaring, A. M., Culler, D. E., Polastre, J., Szewczyk, R., and Anderson, J. (2002) "Wireless sensor networks for habitat monitoring". *WSNA*, pp. 88–97.

[15] Biagioni, E. S. and Bridges, K. W. (2002) "The application of remote sensor technology to assist the recovery of rare and endangered species". *International Journal of High Performance Computing Applications*, 16, 2002.

[16] Biagioni, E. S. and Sasaki, G. (2003) "Wireless sensor placement for reliable and efficient data collection". *Proceedings of the 36 th Annual Hawaii International Conference on System Sciences (HICSS03*, p. 2003.

[17] Wang, H., Elson, J., Girod, L., Estrin, D., and Yao, K. (2003) "Target classification and localization in habitat monitoring". *In ICASSP*.

[18] Baptista, A. M., M. Wilkin, et al. (1999). "Coastal and Estuarine Forest Systems: A multi-purpose infrastructure for the Columbia River." Earth System Monitor 9(3): 1-2, 4-5, 16.

[19] Akyildiz, L. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002) "A survey on sensor networks". *IEEE Communications Magazine*, 40, 102–114.

[20] Milenkovi, A., Otto, C., and Jovanov, E. (2006) "Wireless sensor networks for personal health monitoring: Issues and an implementation". *Computer Communications (Special issue: Wireless Sensor Networks: Performance, Reliability, Security, and Beyond*, 29, 2521–2533.

[21] Xu, N., Rangwala, S., Chintalapudi, K. K., Ganesan, D., Broad, A., Govindan, R., and Estrin, D. (2004) "A wireless sensor network for structural monitoring". *IN SENSYS*, pp. 13–24, ACM Press.

[22] James, P., Demmel, W., Kim, S., Kim, S., and Kim, S. (2005) "Wireless sensor networks for structural health monitoring". Tech. rep., in UC Berkeley Masters Thesis.

[23] Li, M. and Liu, Y. (2007) "Underground structure monitoring with wireless sensor networks". *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, New York, NY, USA, pp. 69–78, ACM.

[24] Gutierrez, J. A., Naeve, M., Callaway, E., Bourgeois, M., Mitter, V., and Heile, B. (2001) "Ieee 802.15.4: A developing standard for low-power low-cost wireless personal area networks". *In IEEE Network Magazine*, p. 5.

[25] Callaway, E., Gorday, P., Hester, L., Gutierrez, J. A., Naeve, M., Heile, B., and Bahl, V. (2002) "Home networking with ieee 802.15.4: A developing standard for low-rate wireless personal area networks". *IEEE Communications Magazine*, pp. 70–77.

[26] Srisathapornphat, C., Jaikaeo, C., and Shen, C.-C. (2000) "Sensor information networking architecture". *In 2000 International Workshop on Parallel Processing (ICPP 2000.*

[27] Song, H., Zhu, S., and Cao, G. (2008) Svats: "A sensor-network-based vehicle anti-theft system". *in Proc. of IEEE INFOCOM*, April, pp. 171–175.

[28] Shi, E. and Perrig, A. (2004) "Designing secure sensor networks". *Wireless Communication Magazine*, 11, 38–43.

[29] Akyildiz I.F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002) "A survey on sensor networks". *IEEE Communications Magazine*.

[30] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002) "Wireless sensor networks: a survey". *Computer Networks*, 38, 393–422.

[31] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. (2000) "System architecture directions for networked sensors". *Architectural Support for Programming Languages and Operating Systems*, pp. 93–104.

[32] Lu, C., Blum, B., Abdelzaher, T., Stankovic, J., and He, T. (2002) "Rap: A real-time communication architecture for large-scale wireless sensor networks". *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2002)*.

[33] Pandey, R., Kottalam, J., Ramin, Y., Wirjawan, I., and Koshy, J. "Peeros–system software for wireless sensor networks". Tech. rep.

[34] Jansen, P., Mullender, S., Hvinga, P., and Scholten, J. (2003) "Lightweigh edf scheduling with deadline inheritance". Technical report (tr-crit-03-23).

[35] Jeong, H.-J., Nam, C.-S., and Shin, D.-R. (2008) "Design and implementation of middleware in sensor networks using publish/subscribe model". *Proceedings of the IEEE International Workshop on Semantic Computing and Applications*, pp. 145–146.

[36] Qi, H., Iyengar, S., and Chakrabarty, K. (2001) "Multiresolution data integration using mobile agents in distributedsensor networks". *IEEE Transaction on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 31, 383–391.

[37] Eicken, V. T., Culler, D., Goldstein, S., and Schauser, K. (1992) "Active messages: a mechanism for integrated communication and computation". *Proceedings of the 19th Annual International Symposium on Computer Architecture*, pp. 256–266.

[38] Mainwaring, A. M. and Culler, D. E. (1999) "Design challenges of virtual networks: Fast, general-purpose communication". *Proceedings of the 1999 ACM Sigplan Symposium on Principles and Practise of Parallel Programming (PPoPP99)*, pp. 119–130.

[39] Arnold, K., Scheifler, R., and Waldo, J. (1999) "*The Jini Specification*". Addison Wesley.

[40] Guttman, E. (1999) "Service location protocol: Automatic discovery of ip network services". *IEEE Internet Computing*, 3, 71–80.

[41] Want, R., Hopper, A., Falco, V., and Gibbons, J. (1992) "The active badge location system". *ACM Transaction on Information Systems*.

[42] Lim, A. "Distributed services for information dissemination in self-organizing sensor networks". *Journal of Franklin Institute, Special Issue on Distributed Sensor Networks for Real-Time Systems with Adaptive Reconfiguration*.

[43] Ratnasamy, S., Estrin, D., Govindan, R., Karp, B., Shenker, S., Yin, L., and Yu, F. (2002) "Data-centric storage in sensornets". *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA)*.

[44] Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2005) "Tinydb: An acquisitional query processing system for sensor networks". *ACM Trans. Database Syst*, **30**, 2005.

[45] Bonnet, P., Gehrke, J. E., and Seshadri, P. (2000) "Querying the physical world". *IEEE Personal Communications*, 7, 10–15.

[46] Franklin, M. (2001) "Challenges in Ubiquitous Data Management Informatics: 10 Years Back, 10 Years Ahead". Springer-Verlag, *lncs #2000* edn.

[47] Bougard, B., Pollin, S., Catthoor, F., and Dehaene, W. (2006) "Cross-layer power management in wireless networks and consequences on system-level architecture". *Signal Processing, Special section: Advances in signal processing-assisted cross-layer designs*, 86, 1792–1803.

[48] Kalpakis, K., Dasgupta, K., and Namjoshi, P. (2002) "Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks". Technical report umbc-tr-02-13.

[49] Younis, M., Youssef, M., and Arisha, K. (2002) "Energy-aware routing in cluster-based sensor networks". *Proceedings of the 10th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS02)*.

[50] Shah, R. and Rabaey, J. (2002) "Energy aware routing for low energy ad hoc sensor networks". *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 350–355.

[51] Singh, S., Woo, M., and Raghavendra, C. S. (1998) "Power-aware routing in mobile ad hoc networks". *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, pp. 181–190.

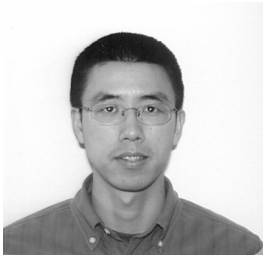
[52] Kansal, A. and Srivastava, M. B. (2003) "An environmental energy harvesting framework for sensor networks". *Proceedings of the 2003 international symposium on Low power electronics and design*.

Authors

Lei Zhang is an assistant professor of computer science, Frostburg State University, University System of Maryland. She received M.S and Ph. D of Computer Science from Auburn University in 2005 and 2008. Before she pursued her graduate study in U.S, she worked as an instructor at school of Electrical Engineering & Automation, Tianjin University, China. Her current research interests include Wireless Networks Protocols Design and Applications, Distributed Algorithm, Information Security, Data Mining and Human Computer Interaction. She is a member of IEEE/ACM. She has been a technical reviewer for numerous international journals and conferences.

Alvin S. Lim is an Associate Professor of Computer Sciences, Auburn University, Auburn, AL. He received his Ph. D of computer science from The University of Wisconsin-Madison. His research interests include self-organizing networks, wireless mobile networks, high performance networks, mobile computing and databases, reliable and dynamically reconfigurable distributed systems, complex software development, parallel processing, and performance measurements and analysis. He has been the technical reviewer and editor for numerous international journals and conferences.

Hui Song received his Ph.D. degree in Computer Science and Engineering from The Pennsylvania State University in 2007. Since then, he has been with the Department of Computer Science at Frostburg State University, where he is currently a tenure-track Assistant Professor of Computer Science. His research interests are in the areas of network and system security, wireless ad-hoc and sensor networks, and mobile computing. He received one best paper award for ACNS'08 and was a recipient of the best research assistant award of the Department of Computer Science and Engineering at the Pennsylvania State University in 2005. He has served as reviewers for numerous conferences and journals and is a member of ACM and IEEE.



Xinliang Zheng received his Ph.D. degree in Computer Science and Engineering from the University of South Carolina at Columbia in 2007. Since then, he has been with the Department of Computer Science at Frostburg State University, where he is currently a tenure-track Assistant Professor of Computer Science. His research interests are in the areas of networks and network security. He is a member of IEEE.