2006-1327: MESH NETWORKS IN EMBEDDED COMPUTER SYSTEMS FOR TECHNOLOGY EDUCATION

C. Richard Helps, Brigham Young University

Craig Malquist, Brigham Young University

Craig Malquist is a Graduate Student in Information Technology at Brigham Young University. He has interests in instrumentation and sensor networks.

Mesh Networks in Embedded Computer Systems for Technology Education

Abstract

Embedded computer systems have advanced significantly in recent years. In the past these were usually low-cost devices with limited processing power. Computational capability was provided principally by independently operating 8-bit microcontrollers. These systems were programmed in assembly language or C code with no or minimal operating system. Now operating systems and communication with external systems are commonplace. With this evolution networking technologies, both conventional and experimental, have also become commonplace in embedded computer systems. Although standard networking technologies, such as Ethernet and TCP/IP are used other networking technologies are more appropriate for the constraints of embedded system applications. Amongst embedded systems with networks the field of low-power self-configuring mesh networks is becoming much more prevalent. They have applications to medical systems, games, family environments, natural habitats, traffic control, military battlefields and elsewhere.

This sub-discipline needs to be included in embedded computer system education. Including both theoretical and applied aspects in a technology course is challenging but possible.

This article summarizes the essential concepts and application domains of mesh networking and the challenges and opportunities of teaching this topic. It also describes practical methods of including both theoretical and applied elements in an embedded systems course for upper-division technology students.

Introduction

Embedded systems have grown over the past several years to take advantage of increasing processing power and memory in smaller and lower power-consumption packages. A major aspect of this growth has been the addition of operating systems and networking technologies to embedded systems. Some of the most widely adopted embedded systems are cell phones and PDAs. These as well as many other embedded applications, have met consumers' needs for communication and increased ability by combining 32-bit hardware together with full-blown operating systems in very small, mobile devices. Palm-size mobile systems with hundreds of megabytes of memory as well as wireless communication, high-resolution color displays and audio and video capabilities are common. In systems of this complexity it is not reasonable for developers to use assembly language or C to design and produce all the software for the application, the user interface, the hardware interface, the memory management and the communications subsystems. Sophisticated development environments and operating systems, together with libraries of functions are needed to enable modern software development approaches to be used to manage the size of the systems and the resources involved. This is especially true of consumer electronic devices, which change as rapidly as Moore's law would predict.

Simpler 8-bit microcontroller systems are still sold by the billion and are found in everything from cars to cordless telephones. These smaller systems typically have memory capacities

measured in hundreds of bytes up to a few kilobytes. These small systems have traditionally been developed using assembly language and C code, usually without an operating system. They have also been widely taught in Engineering, Engineering Technology and other technical college programs[10]. Even these simple systems are now being asked to provide networking capabilities[11] as developers create, and users demand, applications ranging from Bluetooth communications in cars and office equipment to sophisticated communications on a battlefield with dynamically changing communication requirements.

In short we are seeing an expansion of small 8-bit microcontroller system to include standard communication protocols as well as more powerful processors being developed in small low-power packages for use in consumer devices which require sophisticated communication capabilities and operating-system based software.

Specific Constraints of embedded systems

In all these new developments the constraints of embedded systems remain common. Namely:

- Non-standard user interfaces without QWERTY keyboard and screen. Interfaces typically implemented through sensors and pushbuttons, although text-oriented devices (e.g. PDA) may also include alphabetic input mechanisms such as pens or on-screen keyboards. Sensor-oriented interfaces also require matching software access, which is typically different from standard computer peripherals. Device drivers are needed for these sensors.
- Mobile. IE relying on battery power and therefore having significant power consumption constraints. This is particularly prevalent in wireless communication applications where significant power must be transmitted or the receiver must listen for extended periods of time. This often means that the CPU must be run at a relatively low speed to conserve power. This also means that bandwidth can be severely constrained since bandwidth and power consumption are strongly related.
- There are usually strong price constraints especially for consumer devices. This translates into significant memory constraints and the use of flash memory rather than disk drives. Cost, however, may be relative. Doolin and Sitar[7] describe disposable mesh networked motes dropped into the path of advancing wildfires to monitor the progress and conditions of the burn. Although these particular, single-use motes may be expensive by consumer standards they are cheap compared to the damage they can help prevent or mitigate. Similar applications are described by Nikoletseas[14] and Chatzigiannakis[3].
- Real-time constraints: Devices interfacing with the real world often have to meet externally imposed real-time requirements. Quite often those deadlines are relatively slow, of the order of tens of milliseconds, nevertheless they are inflexible and therefore require special accommodation within the application or OS software.
- OS constraints: Due to the CPU speed, memory restraints, lack of fast writable memory (no HDD), and real-time requirements, normal operating systems cannot be used unless they are significantly modified.

As indicated above these systems also now frequently require modern networking capabilities. Sensor mesh networking is a growing sub-discipline which specifically addresses distributed embedded systems gathering data and communicating flexibly to larger systems. Within the

constraints of embedded systems, sensor mesh networks use their combined processing power to provide the sensing, processing and communication needs of the application[19]. Sensor mesh network technologies provide modern networking, with consideration for routing, data assurance and security, low power, flexibility and, as the name states, accommodation for sensors.

Sensor Mesh Networks

Mesh networking allows a group of devices to inter-communicate with each other in the network. They are mesh connected. IE nodes are typically in contact with several other nodes in the network, thus providing multiple communication paths through the network. Mesh networks are self-configuring and also self-healing in that nodes can be added or removed and the mesh will still be able to function. Mesh network nodes have the ability to pass information from one mote on to another mote in a multi-hop configuration. Through these characteristics a mesh network can grow out from the central location by finding nodes within radio range and then establishing communication with those nodes. Information that needs to go to the central location is passed through the mesh of nodes until it reaches its destination. There are obvious networking design issues that must be dealt with here, such as how to select a path through the mesh and how to ensure that the information does get through the mesh to its destination.

Sensor mesh networks have a similar architecture to regular mesh networks except that sensors are attached to the nodes and the information gathered from the sensors is sent to a destination, usually a database attached to the mesh network, for storage. Sensor mesh networks incorporate a variety of sensors to be able to gather information about their environment. There are various sensors that are used including "light, temperature, vibration, magnetic field, acoustic, and wind shear, [which] can be integrated on the mote depending on the application"[17].

Power becomes a dominant factor in designing sensor mesh networks. The nodes in a sensor mesh network are usually mobile and must therefore rely on stored power, such as batteries, or harvested power, such as solar cells. Both stored and harvested power technologies provide power in the milliwatt range or smaller and thus power considerations dominate system design. There are several devices on a node which can affect the available power and power consumption of a node. First, the battery is the main source for power on a node. Battery size and capacity will depend on the size of the mote, ranging from an AA battery for a Crossbow Micaz mote of a couple of cubic inches in size, to batteries integrated onto a $1mm^3$ smart dust mote[13]. Once the battery is depleted the mote is rendered useless until the battery is replaced. Research is being done on solar cells to help improve the life of the battery[17].

The sensors and the communication systems are not the only significant sources of power consumption. A sophisticated operating system running on a hardware base of a multi-Megahertz CPU and multi-megabyte memory system can also consume significant power. The standard power conservation technique in embedded systems is to have the hardware 'sleep' periodically; operations, including the clock, are suspended for a period until the system is woken by an external interrupt or a very low power timer triggering a wake-up interrupt. The OS must support, enable and activate this functionality and the networking system (hardware and software) must be designed in the expectation that other nodes may be asleep for a very large percentage of their operating time. This is particularly challenging in communication

environments. Receivers and transmitters must be awake at the same time for communication to take place.

Third, the types of sensors used will affect the power usage of the system. Sensor power consumption requirements can vary widely. While low power sensors will obviously be favored over higher-powered ones, choices are frequently constrained by the application. Once again the sensor is often put into a sleep state so that power is only consumed while readings are actually being taken. The analog to digital conversion chip that will allow the microprocessor to read the information from the sensors will also require power and thus a power efficient chip must be chosen. Conversion chips are frequently incorporated into microcontrollers. In these cases there is often an option to turn them on or off to conserve power.

A well known variant of the sensor mesh networking idea is that of "Smart Dust". The Smart Dust Project was run by Pister, Kahn and others at Berkeley and had a goal of creating a computerized 'mote' about 1 mm^3 with independent power supply and the capability to collect data and communicate it to other motes[13, 15]. Micro-Electro-Mechanical Systems (MEMS) techniques are used to incorporate sensors and computational capability on the same chip. The Smart Dust Project produced a number of interesting concepts and derivative projects but did not of itself become a standard. The terminology 'smart dust' is now sometimes used in a generic sense to refer to very small computerized motes.

Many applications are being proposed to use mesh networking and sensor mesh networks. Firefighting applications have already been mentioned. Firefighters use smart dust networks to aid in monitoring fires (both building fires and wildfires). Sensors can be distributed over the area concerned and then relay position, temperature and other information back to a central location to aid in tracking and extinguishing fires. Researchers have proposals to use smart dust to easily track changes in the environment[19], manufacturing plants and monitoring security[4]. Smart dust could also be used in hazardous environments (such as a chemical spill in a building or a radioactive environment like Chernobyl) where humans cannot enter but cheap sensors can be dispersed to gather information. Another application is internet access for small communities[2]. The military is interested in mesh networks for troop communications[5]. In military operations where it is difficult to deploy network access points in the field, mesh networks provide an easy way to establish communication with both members of the same division and commanders. Mesh networking provides an easy way to move troops between teams if redeployment or changes in deployment are needed. A member of one mesh simply leaves and joins the new mesh. Both mesh networks reconfigure, or self-heal, and return to full functionality. Telecommunications companies are also interested in mesh networking for increased reliability between local offices.

Standards (and lack thereof)

There are currently no IEEE approved standards for mesh networking or smart dust. A standard, IEEE 802.11s, is currently in development. The 802.11s standard for mesh networking seems mostly targeted at ISP distribution models, taking advantage of mesh networking's features of being self-configuring, self healing and passing traffic for distant nodes. Bilel Jamoussi of Nortel networks, who is working with the group, indicates that there is progress, but there is

some way to go, "By May 2006, I expect we'll have a single document…"[9]. The development of a standard is an important step as many interested players such as Intel, Nortel, Nokia and Motorola, who have an interest in the area, will be reluctant to release products until a standard is agreed. Some products are available but they are experimental in nature and suitable for design teams, researchers and others to use for exploration in anticipation of the standards approval and major release of products in the next few years.

An alternative approach to sensor mesh networking is represented by the Zigbee standard[18]. Zigbee is based on the IEEE 802.15.4 networking standard. The Zigbee Alliance states that one of their goals is to "create a specification defining mesh, peer-to-peer, and cluster tree network topologies with data security features and interoperable application profiles. The ZigBee specification provides a cost-effective, standards-based wireless networking solution that supports low data rates, low power consumption, security and reliability"[18].

Mesh Networking Principles

Mesh networking systems are designed around a few basic principles that should be included in a course on the topic. These principles include networking topologies and some routing concepts and algorithms.

There are many different network topologies that are used today. Common network topologies are star, ring, token ring, bus, and mesh. There are two types of mesh networks, a full mesh and a partial mesh network. In a full mesh network each node in the network is connected to all the other nodes in the network. In a partial mesh network a node will be connected to a few nodes in the network, but not all. All the nodes will be able to communicate with each other; however, the information may need to be passed through intermediate nodes to reach the final destination. All sensor mesh networks depend on a base mesh network in order to function. Mesh networks can be implemented as wired or wireless mesh networks.

Wired Mesh Networks

Wired mesh networks are of interest to telecommunication enterprises. They use multiple wired links between nodes, although only one link is active between a node and the node feeding it to prevent looping. The process of choosing which links are active is similar to the Spanning Tree algorithm that is currently used in many network topologies. If the network is to be connected to the 'outside world' an end node needs to have a path back to the core. If the network is only a local network, all devices need to be able to communicate with each other; routing will need to allow for all nodes to communicate. In Local Area Networks (LAN) each connection will require a Network Interface Card (NIC) at each end of the connection. Depending on how many nodes are in the network, full mesh networks may not be possible in practice as there would be too many NICs in each node. For a full-mesh network NICs_per_node = (nodes-1).

Wired mesh networks can also be implemented in Wide Area Networks (WAN) to connect central offices in telecommunication systems. They can provide superior connectivity relative to current systems. Many WAN networks (such as ISPs) currently use a SONET ring topology. This topology has all of the central offices connected in a ring. A SONET ring has two links

between each central office that are adjacent in the ring; only one of these links is active and traffic normally travels around the link in one direction. A SONET ring achieves reliability through the redundant second ring. If a link is cut the central offices will discover this and communicate it to other nodes on the ring. Traffic is then routed using the second link so that the two central offices next to the break can still communicate. Subscribers are connected to the Central Offices (CO) in a form of a Star topology with the central office as the central point of the network. Mesh networks attempt to improve on this inter-office SONET topology as indicated by Demartino[6],

> Mesh-based interoffice networks can provide greater flexibility, efficiency, and robustness than ring-based networks. There is considerably more flexibility in provisioning both working paths and protection paths through mesh networks, which results in less stranded capacity and greater protection efficiency. Also, compared to ring-based networks, mesh-based networks can provide protection against a wide range of failures, particularly node failures and multiple failures.

Rings can still be used; however, there are other options besides rerouting along the second link such as going through an adjacent central office. In a mesh, if a link breaks between the Source CO and the Destination CO there are multiple paths that the data can travel to reach the destination.

Wireless Mesh Networks

Wireless networks are becoming increasingly popular as easy ways to connect to a network with mobile devices such as laptops and PDAs. Currently wireless local area networks (WLANs) are deployed with a central Wireless Access Point (WAP) as a "hot spot" to which the devices in the area are connected (a star topology). This configuration is used in home and business environments with minor variations. There may be many devices connected to one access point thus sharing a collision domain and limiting throughput. Each access point is then connected through a wired link into the main network so that access to devices outside the local wireless network is possible.

Two problems with the star topology are lack of redundancy and limited range. If an access point fails users in the area will lose access to the network. There is also the problem of the distance that devices can be from the access point and still be connected to the network. Mesh networks can help with both of these problems. A mesh topology can enhance reliability by providing multiple routes for data to travel to reach a destination. By using the nodes as routers each node that is in the network will help enhance reliability. Mesh networks will also be able to help extend wireless networks by allowing multi-hop routing allowing nodes to transmit information through other nodes in the network.

> Currently deployments of 802.11-based networks are purely single hop, with the various mobile or client devices connecting to the access point (AP) via a direct wireless link. There is, however, great excitement surrounding the notion of *multi-hop, wireless 802.11- based mesh networks*, where the wired backbone is reachable only via *multiple wireless hops*[1].

By using multiple wireless hops new wireless nodes can be added to the network as long as they are in range of nodes currently in the mesh network.

Both wired and wireless mesh networks are affected by available bandwidth. The routing algorithms need to take into account bandwidth as routes are calculated. This is not a great concern for wired mesh networks; however, with the limited bandwidth available to wireless network devices great care must be taken so that one link is not overloaded with traffic. The routing algorithms in the wireless network need to be able to know when a node is added or taken away from a network and then recalculate the routes for the network. Networks try to configure so that each device takes the shortest and least congested route in the network.

With the abundance of 802.11x equipment that is already in use a wireless mesh network infrastructure is already in place in many areas. However, using 802.11b/g as a connection method will not work effectively. There is no way to share a link with another device and have both devices be able to have an equal portion of the capacity to transmit packets. If left as currently implemented the result will be that the closest node will take whatever bandwidth it needs and the devices behind it will get what is left over.

> The question of the capacity of the network cannot be properly addressed without discussing the issue of fairness. Note that a user node in Wireless Mesh Networks has to transmit relayed traffic as well as its own. Therefore, besides contention with other nodes for the same destination node, there is inevitable contention between its own and relayed traffic. This type of contention does not occur in fixed wireless local loops or wireless LANs in infrastructure mode where user nodes are always at one-hop distance from the base station or access point[8]

If fairness is not taken into account there will be a bottleneck at the wireless link connecting the base to the adjacent nodes. With fairness each node will get a portion of the bandwidth to transmit its information. As the mesh network grows larger the amount of bandwidth given to one node will decrease. A mesh network cannot grow forever as the needed bandwidth will run out at some point. This phenomenon has been researched and the loading curves and equations are known. Downstream nodes can be choked off as their communications converge on nodes further up the line, which are also handling their own traffic.

Routing

Each node in a wireless mesh network acts as a router because in the process of configuration it discovers who it is connected to, the current connection map, and the capacity of each link so that it can make a decision as to where to send the data it receives. The individual nodes will be routing the traffic of those downstream through itself up to the access point. Routing for mesh networks needs to come from the link layer as the network will change based on nodes that come on the network or leave the network.

> In radio networks, the link-layer determines how changes are detected. Periodic, network layer messaging to detect topology changes is either redundant or not possible. Bandwidth is at a premium in mobile applications. Protocols must be as lightweight as possible, preferably expending no more than 10 percent of total network traffic on overhead[5].

Because bandwidth is so limited in wireless networks routing needs to take as little of that bandwidth as possible. At the same time it is necessary to detect changes and adapt the network

to those changes so it is necessary to have the nodes find out about changes with as little information as possible. Changes also need to happen in a quick enough time that the network doesn't change before the current network routing has been implemented. In wired, static networks there are a few ways to determine routing including shortest-path and distance vector routing. In wireless mesh networks these distance algorithms can take up bandwidth and time, if the network changes before the calculation is finished the time and bandwidth spent calculating that algorithm will be lost.

Two characteristics that are required for a routing algorithm in a wireless mesh network are that it can build routes fast and if there are changes it can react quickly to them to get the network back up and running. One routing algorithm that accomplishes these characteristics is called Lightweight Multipath Routing (LMR). LMR is able to do the following:

- maintain multi-path routing which is loop-free at every instant
- operates correctly in the presence of arbitrary topological changes
- remain completely distributed, operating only with one-hop topological knowledge (i.e. it only knows who its adjacent routers are)
- does not involve a shortest-path computation, although the shortest-path option is usually (and often unknowingly) present at a router[5].

Corson[5] explains that the shortest-path option is often set by default in routers; however this option does use extra bandwidth as it is necessary to calculate what the shortest path to the destination is. Users need to be aware of this option and disable it if needed for the situation. In LMR routes are made randomly as a function of reply packet propagation. Through this packet propagation the routers learn who their neighbors are and whether they are incoming or outgoing with respect to a given destination.

Available Software and Operating Systems

A number of de-facto software environment standards are developing in the user community for sensor mesh networking. One of these is TinyOS[16] (http://www.tinyos.net).

> TinyOS is an open-source operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. TinyOS's component library includes network protocols, distributed services, sensor drivers, and data acquisition tools – all of which can be used as-is or be further refined for a custom application. TinyOS's event-driven execution model enables fine-grained power management yet allows the scheduling flexibility made necessary by the unpredictable nature of wireless communication and physical world interfaces.

The user interface for this operating system is similar to a command-line UNIX system. An outstanding feature of the OS is the memory footprint size. It can, and has been, installed on 8-bit microcontroller systems with tens of kilobytes of memory. Like almost all embedded systems the development environment is hosted on a much larger, separate workstation than the target system. Since the OS is specifically designed for sensor mesh networks the libraries include appropriate functionality to address the constraints of these systems.

The programming language commonly used with TinyOS is nesC[8] (http://nescc.sourceforge.net). nesC is a C language dialect, designed in conjunction with TinyOS. The authors' claim for nesC is as follows: (*ibid*)

> nesC's contribution is to support the special needs of this domain by exposing a programming model that incorporates event-driven execution, a flexible concurrency model, and component-oriented application design.

Although its origins lie in the C language nesC has much in common with object-oriented languages with constructs for *providing*, *using* and *implementing* components (objects). nesC is not a hard real-time language but its support for concurrency and user control allows a measure of real-time support under control of the programmer.

Teaching Sensor Mesh Networks: Objectives and outcomes

Teaching sensor mesh networking within the context of an embedded systems course depends on the background of the students and the objectives of the course. We have been teaching embedded systems for some years and are able to cover embedded systems principles in a single semester-long course. Prior to taking the course the students have had other classes in computer architecture, digital logic, object-oriented programming, assembly language programming, operating system and networking concepts (in a desktop-computing environment) and other computer-oriented topics. The main part of the embedded system course covers fundamental concepts and applications of small (8-bit) and larger (OS-based) embedded systems, real-time concepts and applications and includes class and laboratory work in interfacing sensors and actuators to embedded systems. Within this context we added a module to teach the above principles of sensor mesh networking, combined with a single lab experience. The objectives of the module were that students should

- Be cognizant of the evolution and need for sensor-mesh networking in embedded systems
- Be cognizant of the features and constraints of mesh networking systems. Including current and emerging standards.
- Be able to describe and work with the principles of implementation (networking, routing etc.)
- Be able to implement and test at least one working mesh networking system in a few configurations. (Lab assignment)

There is adequate educational material, summarized above, for class instruction. Class instruction can be enhanced through interesting applications (case studies) of the technology.

In order to implement the lab assignments a simple available sensor mesh networking system was obtained. There are a few of these systems available from different suppliers. The one selected was chosen because the price was reasonable (hundreds of dollars per mote), the hardware was available and the company had a number of demonstration applications which could be used as starting points for lab exercises. It is expected that within the next year there may be several newer competing products with equivalent capability and similar prices to choose from.

Mesh Networking Lab Experiences

A lab experience has been designed to help students experiment with and learn applications of the theoretical principles discussed above. The lab uses Micaz motes from Crossbow (http://www.xbow.com) which run TinyOS and NesC. They implement the Zigbee communications protocol. The motes are all identical, with CPU, power supply, and radio functionality but they can be enhanced with sensor modules or communication modules for linking to Ethernet systems. In order to be able to cover the basic principles of mesh networking within a very limited time extensive use is made of the demonstration code provided by the manufacturer. This gives students a good starting point and enables them to focus on the performance of the wireless sensor mesh network, rather than spending many hours discovering the hardware and software intricacies of the motes. If more class and lab time were available these explorations could be valuable learning experiences.

When setting up a mesh network one of the motes needs to have a communication module enhancement to allow information gathered by the mesh to be captured by a server. This mote is often referred to as the base station. The other motes can either have a sensor module attached or be a stand-alone mote in the network. In a functioning mesh network information is relayed to the base station through the mesh network. The base station mote with the communication enhancement then relays the information to the server.

In order to learn about the features of mesh networking a few configurations are explored. First, the students use two motes to establish communication between the base and a mote. This step is to ensure that communication is occurring between the two nodes of the mesh network. Second, a third mote is added in a configuration that forces the furthest mote to relay information through the closer mote to reach the base station. Adding the third mote will test the multi-hop capability of the mesh network. In this configuration the students can test how the mesh network can reconfigure or self-heal by swapping the location of the roaming motes so that the mote that was previously closest to the base now has to send its information through the previously farthest mote. Third, a fourth mote is added to the mesh network to allow the students to study how link utilization can affect paths through the network. The network is configured in a way that two motes are connected to the base station with one sending information quickly and the other slowly. The fourth mote is then added so that it has to communicate through the one of the other two motes. Students will be able to see which mote (the fast or the slow one) the furthest mote connects to. Fifth, a fifth mote is added to the network in a configuration as shown in figure 1. This configuration will further test path choices of the mesh network.

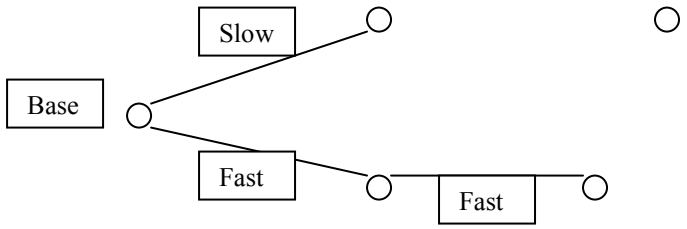

Figure 1: Configuration for a five-node sensor mesh network lab demonstration.

By completing this lab students should become familiar with the programming environment used by the Micaz motes including how to set up a computer to be used for programming. Students should also become familiar with "Surge View" which allows a user to see how the mesh network is configured and which motes are connected. Students should understand how the programs are structured, compiled, and then transferred to the motes. Students also should understand where settings for the motes are kept in the programs so that they can be changed for different applications and network configurations.

The questions the students should be able to answer by the time they have completed the lab are:

1. Are the motes able to communicate?
2. Does the mesh network exhibit multi-hop capabilities?
3. Is the mesh network self configuring?
4. Does the mesh network have the ability to self-heal or reconfigure itself?
5. How long does it take to self-heal or reconfigure?
6. In the 4-mote configuration what route did the mote furthest from the base take?
7. In the 5-mote configuration, which route did the mote furthest from the base take? Does it prefer the shortest path or the quickest path?

This instructional module has been taught on a trial basis to students. Student reaction was positive. They enjoyed discovering new technical areas within embedded systems that were on the leading edge of technology. Students seemed quite comfortable relating the material to existing networking systems with which they had some familiarity. The labs took an inordinate amount of effort because there were unexplained issues with the development system. This is expected to become easier rapidly as familiarity increases and improvements are made by the suppliers.

These lab experiences will be extended in future semesters to more extensive measurement of the system performance.

Conclusions

Sensor mesh networking is an important topic for embedded system development professionals and for students in the discipline. Those involved in embedded system development need to understand both the theory and applications of sensor mesh networking. Emerging standards and strong industry support, with multiple potential applications, ensure that this technology will become commonplace.

The principles of mesh networking are variations on existing technologies, adapted for the constraints of embedded systems. Development systems are available at reasonable prices and work well enough to illustrate the principles with reasonable amounts of set-up.

Teaching sensor mesh networks has a natural place within an embedded systems course or curriculum. The background of these students is very appropriate for this new sub-discipline.

The objectives of such a course should ensure that students can demonstrate understanding of the relationship between embedded systems and this new sub-discipline; demonstrate understanding of the key principles, theories and applications of the system itself; and can apply the principles in labs to solve problems.

Our limited experience has shown that students respond positively to this new topic area and it is our expectation that this will grow significantly in the next few years and embedded system courses and programs will be best served by incorporating this into the curriculum.

Bibliography

1. Acharya, A., Misra, A., Bansal, S. (2004). Design and analysis of a cooperative medium access scheme for wireless mesh networks. *Proceedings. First International Conference on Broadband Networks, 2004,* 621-631.

2. Bruno, Raffaele., Conti, Marco., and Gregori, Enrico, (2005) Mesh Networks: Commodity Multihop Ad Hoc Networks IEEE Communications Magazine, March 2005, pp 123-131

3. Chatzigiannakis, I., Nikoletseas, S. (2003). A sleep-awake protocol for information propagation in smart dust networks. *International Parallel and Distributed Processing Symposium, 2003, p. unknown.*

4. Chong, Chee-Yee, and Kumar, Srikanta P., (2003) Sensor Networks: Evolution, Opportunities, and Challenges, *Proceedings of the IEEE, Vol. 91, No. 8, August 2003*

5. Corson, S., Macker, J., Batsell, S. G. (1996). Architectural considerations for mobile mesh networking. *Military Communications Conference,* 1, 225-229.

6. DeMartino, K. (2003). An architecture for a seamless mesh network. *Proceedings on Fourth International Workshop on Design of Reliable Communications Networks,* 405-412.

7. Doolin, David M., Sitar, Nicholas. (2005) Wireless sensors for wildfire monitoring, Proceedings of SPIE Symposium on Smart Structures & Materials/ NDE 2005, San Diego, California, March 6-10, 2005.

8. Gay, David, Levis, Phil, von Behren, Rob, Welsh, Matt, Brewe,r Eric, and Culler, David (2003) The nesC Language: A Holistic Approach to Networked Embedded Systems, *Proceedings of Programming Language Design and Implementation (PLDI) 2003, June 2003*

9. Goth, G. (2005). Groups hope to avoid mesh standard mess. *IEEE Distributed Systems Online,* vol 6, no 9 (Sept 2005), 1-6.

10. Helps, C. Richard G., Phillips, David P.(1999) Teaching Embedded Systems Using Multiple Microcontrollers. *Conference Proceedings, ASEE 1999, Session 3547.*

11. Helps, C. Richard G.(2002) Teaching Embedded Systems: From Eight Bits to Operating Systems and Networks. *Conference Proceedings, ASEE 2002, Session 3647.*

12. Jun, J., Sichitiu, M. L. (2003). The nominal capacity of wireless mesh networks. *Wireless Communications, IEEE, 10,* 8-14.

13. Kahn, J.M., Katz, R.H., Pister, K.S.J. (1999) Next century Challenges: Mobile Networking for "Smart Dust" Proceedings of the 5th Annual ACM International Conference on Mobile Computing and Networking (MobiCom)

14. Nikoletseas, S., Chatzigiannakis, I., Euthimiou, H., Kinalis, A., Antoniou, A., Mylonas, G. (2004). Energy efficient protocols for sensing multiple events in Smart Dust Networks. *37th Annual Simulation Symposium, 2004, 15-24.*

15. Pister K. S. J., Kahn J. M., and Boser B. E. (1999) Smart dust: Wireless networks of millimeter-scale sensor nodes. In 1999 UCB Electronics Research Laboratory Research Summary

16. TinyOS (2006), What is TinyOS? Mission Statement. Accessed at http://www.tinyos.net/special/mission Jan 2006

17. Warneke, B., Scott, M., Leibowitz, B., Zhou, L., Bellew, C., Chediak, J., et al. (2002). An autonomous 16 mm^3 solar-powered node for distributed wireless sensor networks. *Sensors, 2002, Proceedings of IEEE, 1510 – 1515.*

18. Zigbee Alliance (2006) FAQ #3. Accessed at http://www.zigbee.org/en/about/faq.asp#8 Jan 2006.

19. Culler, David, Estrin, Deborah and Srivastava, Mani, (2004) Overview of Sensor Networks, *Published in IEEE Comnputer, August 2004*